**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

Курсовая РАБОТА

**по дисциплине «Автоматизация схемотехнического проектирования»**

**Тема**: **Анализ экспериментальных данных**

|  |  |  |
| --- | --- | --- |
| Студент гр. 1302 |  | Новиков Г.В. |
| Преподаватель |  | Боброва Ю.О. |

Санкт-Петербург

2025

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студент Новиков Г.В. | | |
| Группа 1302 | | |
| Тема работы: Анализ экспериментальных данных | | |
| Исходные данные:  Датасет, содержащий информацию о мобильных телефонах. | | |
| Содержание пояснительной записки:  «Содержание», «Введение», «График распределения параметров», «Медиана», «Графики распределения при медианном значении», «Среднее значение и стандартное отклонение», «Различия в классах», «Классификация», «Листинг», «Заключение», «Список использованных источников» | | |
| Предполагаемый объем пояснительной записки:  Не менее 15 страниц. | | |
| Дата выдачи задания: 03.02.2025 | | |
| Дата сдачи реферата: 24.03.2025 | | |
| Дата защиты реферата: 24.03.2025 | | |
| Студент гр. 1302 | |  | Новиков Г.В. | |
| Преподаватель | |  | Боброва Ю.О. | |

**Аннотация**

В данной курсовой работе проводится анализ экспериментальных данных, представленных в виде датасета, содержащего характеристики мобильных телефонов. Основной целью работы является исследование параметров, таких как цена, диагональ экрана, объем оперативной памяти и другие, с использованием различных статистических методов и визуализации данных.

**Summary**

In this course work, an analysis of experimental data is conducted, presented in the form of a dataset containing characteristics of mobile phones. The main goal of the work is to investigate parameters such as price, screen size, RAM capacity, and others, using various statistical methods and data visualization techniques.

СОДЕРЖАНИЕ

[Введение 5](#_Toc193647127)

[1. Графики рапредения параметров 7](#_Toc193647128)

[2. Медиана 14](#_Toc193647129)

[2.1. Медиана параметра n 14](#_Toc193647130)

[2.2. Медиана параметра m 14](#_Toc193647131)

[3. Графики распределения при медианном значении 15](#_Toc193647132)

[3.1. График распределения параметров 15](#_Toc193647133)

[3.2. Гистограмма распределения, скатерограмма и боксплот параметров 22](#_Toc193647134)

[4. Среднее значение и стандартное отклонение 26](#_Toc193647135)

[5. Различия в классах 27](#_Toc193647136)

[5.1. Различия в классах по параметру n 27](#_Toc193647137)

[5.2. Различия в классах по параметру m 27](#_Toc193647138)

[6. Классификация 29](#_Toc193647139)

[Листинг 31](#_Toc193647140)

[Заключение 40](#_Toc193647141)

[Список использованных источников 41](#_Toc193647142)

Введение

Датасет содержит 1512 строк с характеристиками мобильных телефонов:

1. phone\_name — Название модели телефона.
2. brand — Бренд или производитель телефона.
3. os — Операционная система, установленная на телефоне.
4. inches — Диагональ экрана телефона в дюймах.
5. resolution — Разрешение экрана в формате "ширина x высота" (например, "1080x1920").
6. battery — Емкость аккумулятора телефона в миллиампер-часах (мАч).
7. battery\_type — Тип аккумулятора (например, "Li-Ion" или "Li-Po").
8. ram(GB) — Объем оперативной памяти (RAM) в гигабайтах (ГБ).
9. announcement\_date — Дата анонса телефона в формате "год-месяц-день".
10. weight(g) — Вес телефона в граммах (г).
11. storage(GB) — Объем встроенной памяти телефона в гигабайтах (ГБ).
12. video\_720p — Поддержка записи видео с разрешением 720p (HD).
13. video\_1080p — Поддержка записи видео с разрешением 1080p (Full HD).
14. video\_4K — Поддержка записи видео с разрешением 4K (Ultra HD).
15. video\_8K — Поддержка записи видео с разрешением 8K.
16. video\_30fps — Поддержка записи видео с частотой 30 кадров в секунду (fps).
17. video\_60fps — Поддержка записи видео с частотой 60 кадров в секунду (fps).
18. video\_120fps — Поддержка записи видео с частотой 120 кадров в секунду (fps).
19. video\_240fps — Поддержка записи видео с частотой 240 кадров в секунду (fps).
20. video\_480fps — Поддержка записи видео с частотой 480 кадров в секунду (fps).
21. video\_960fps — Поддержка записи видео с частотой 960 кадров в секунду (fps).
22. price(USD) — Цена телефона в долларах США (USD)

Рассматриваемый параметр – price(USD).

Задачи:

1. Построить график распределения параметров. Разделить выборку по классам. Подписать оси, добавить легенду и сетку на график.

2. Рассчитать медиану параметра n: для выборки в целом, для каждого класса отдельно

3. Рассчитать медиану параметра m: для выборки в целом, для каждого класса отдельно

4. Построить график распределения параметров для объектов выше медианного значения в выборке и ниже. Подписать оси, добавить легенду и сетку на график.

5. Построить гистограмму распределения, скатерограмму и боксплот параметров для объектов выше медианного значения в выборке и ниже. Подписать оси, добавить легенду и сетку на график.

6. Рассчитать среднее значение и стандартное отклонение для параметров: для всей выборки и для каждого класса отдельно.

7. Статистически оценить различия в классах по параметру n- совпадают ли выборки по этому параметру или нет.

8. Статистически оценить различия в классах по параметру m - совпадают ли выборки по этому параметру или нет.

9. Классификация данных.

1. Графики рапредения параметров

Задание: построить график распределения параметров. Разделить выборку по классам. Подписать оси, добавить легенду и сетку на график.

Для начала построим гистограмму для рассматриваемого параметра.

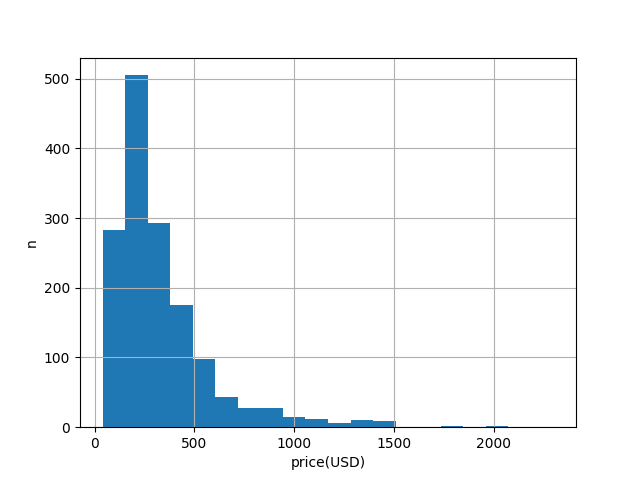


Рис. 1. Рассматриваемый параметр (price (USD))

Разделим выборку на 2 класса:

Класс 1: объекты со значением price(USD) больше или равным медиане.

Класс 2: объекты со значением price(USD) меньше медианы.

Медиана price(USD) равна 260.

Построим гистограммы для оставшихся параметров с разделением на классы. Для параметров brand, os, resolution, announcement\_date построены столбчатые диаграммы с 10 наиболее часто встречающимися значениями. Для параметра phone\_name график не был построен, так как для большинства строк значение уникально.

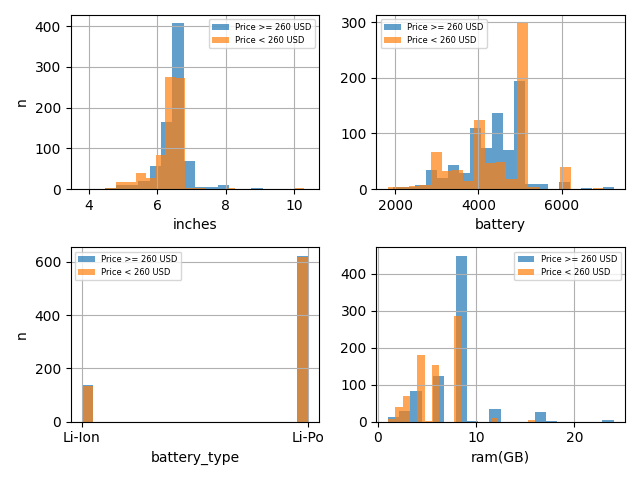


Рис. 2. Параметры

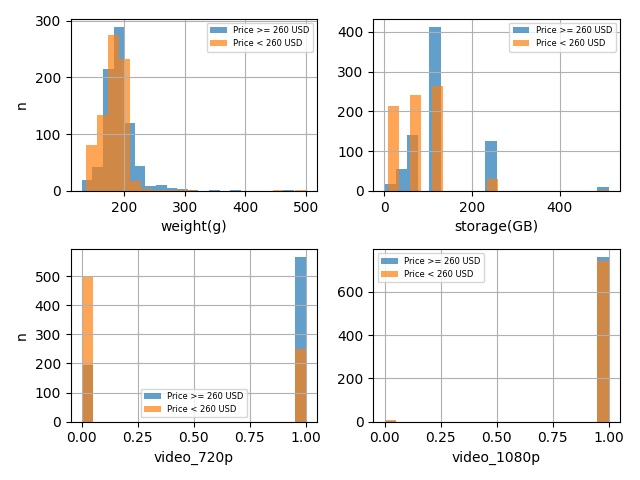


Рис. 3. Параметры

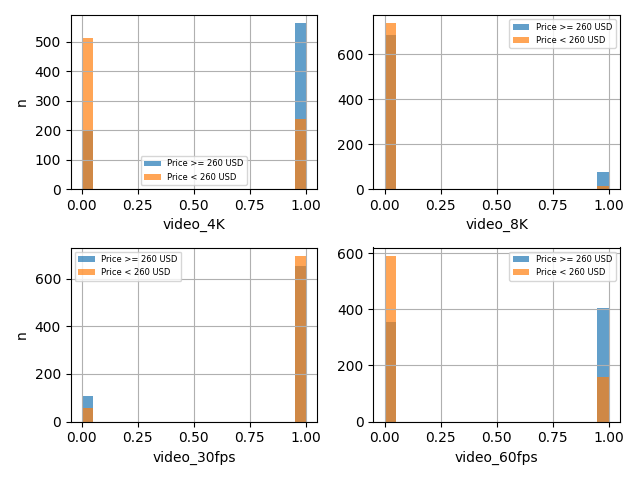


Рис. 4. Параметры

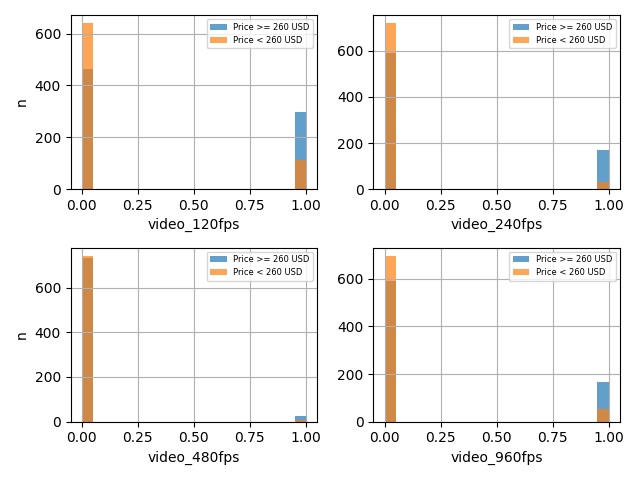


Рис. 5. Параметры

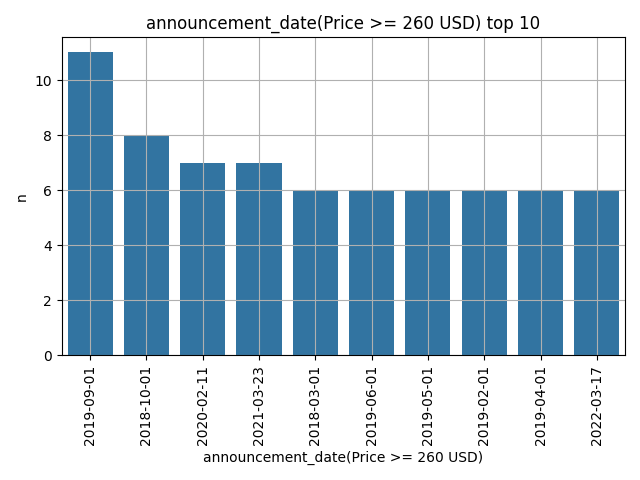


Рис. 6. Параметр

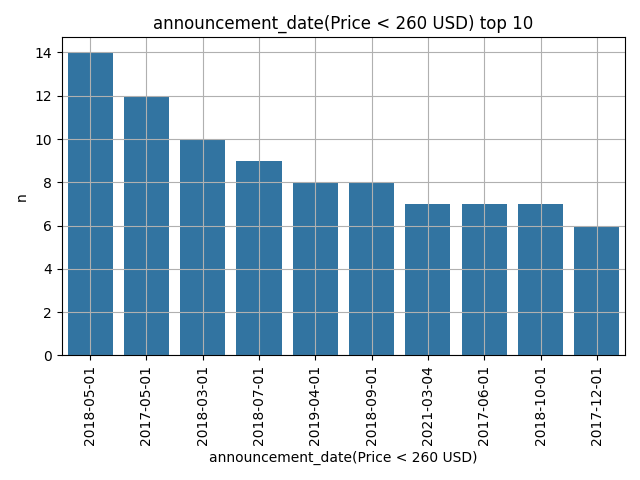


Рис. 7. Параметр

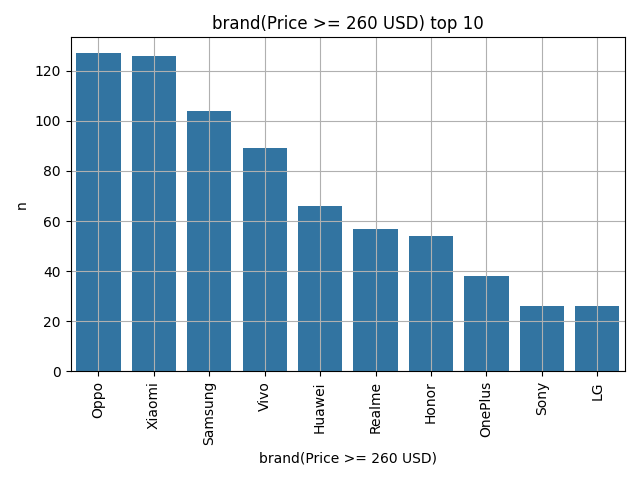


Рис. 8. Параметр

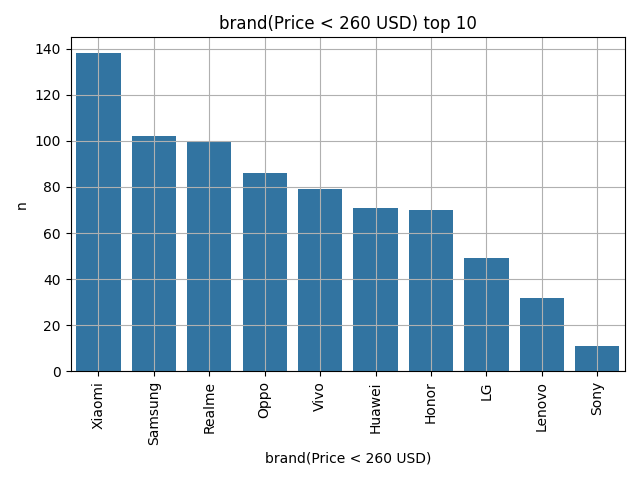


Рис. 9. Параметр

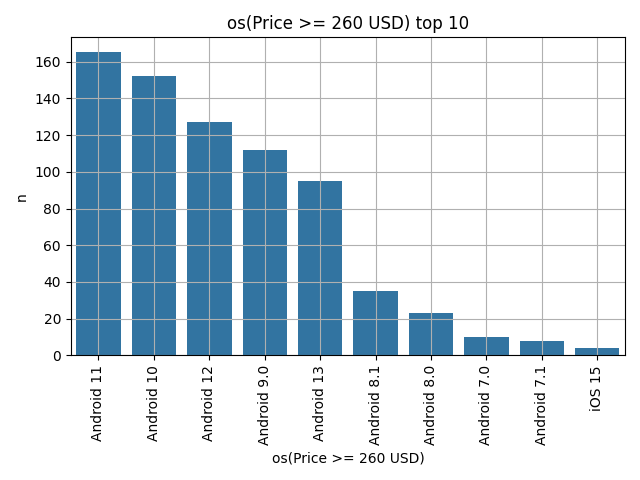


Рис. 10. Параметр

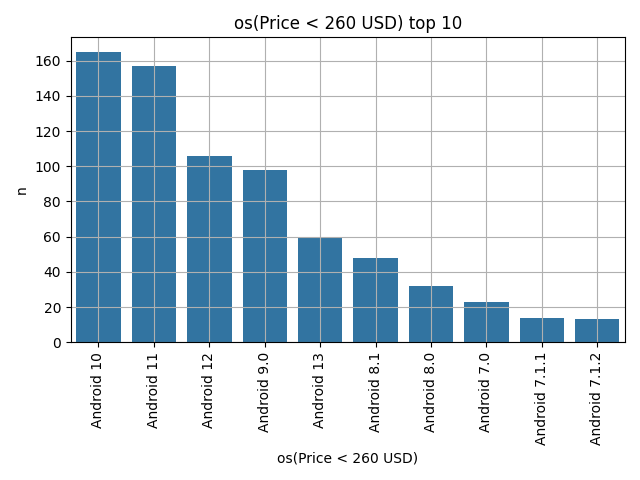


Рис. 11. Параметр

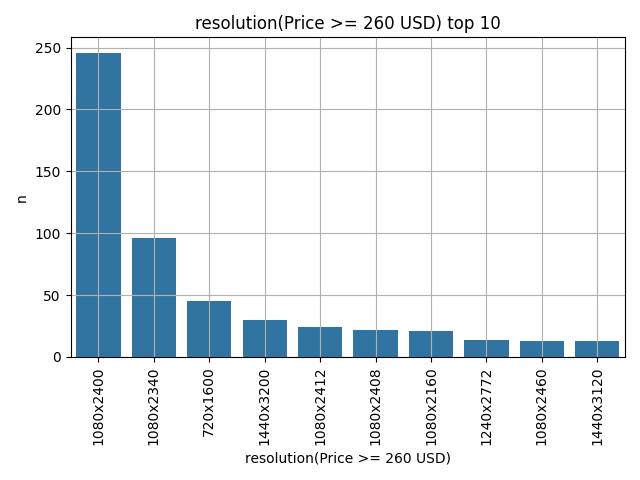


Рис. 12. Параметр

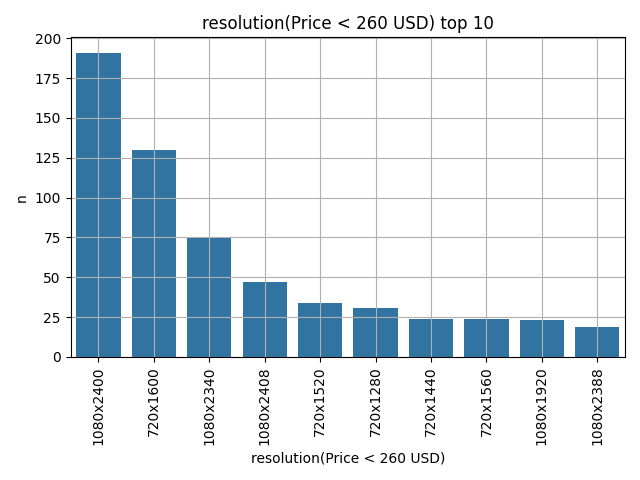


Рис. 13. Параметры

2. Медиана

2.1. Медиана параметра n

Задание: рассчитать медиану параметра n: для выборки в целом, для каждого класса отдельно.

Медиана была рассчитана для параметра inches.



Рис. 14. Медиана параметра inches.

2.2. Медиана параметра m

Задание: рассчитать медиану параметра m: для выборки в целом, для каждого класса отдельно.

Медиана была рассчитана для параметра ram(GB).



Рис. 15. Медиана параметра ram(GB).

3. Графики распределения при медианном значении

3.1. График распределения параметров

Задание: построить график распределения параметров для объектов выше медианного значения в выборке и ниже. Подписать оси, добавить легенду и сетку на график.

Данные разделены относительно медианы параметра ram(GB).

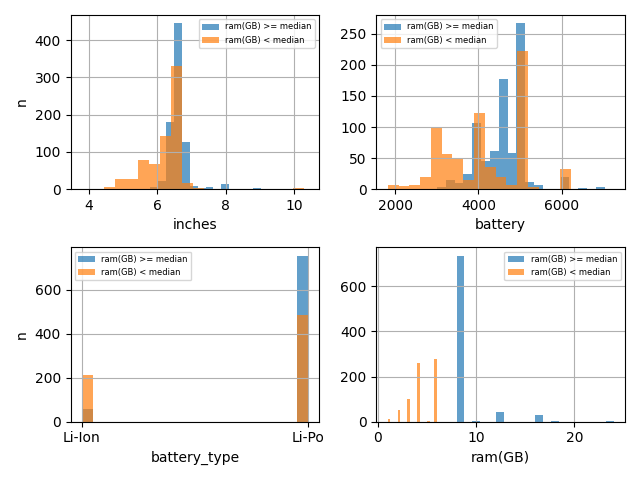


Рис. 16. Параметры

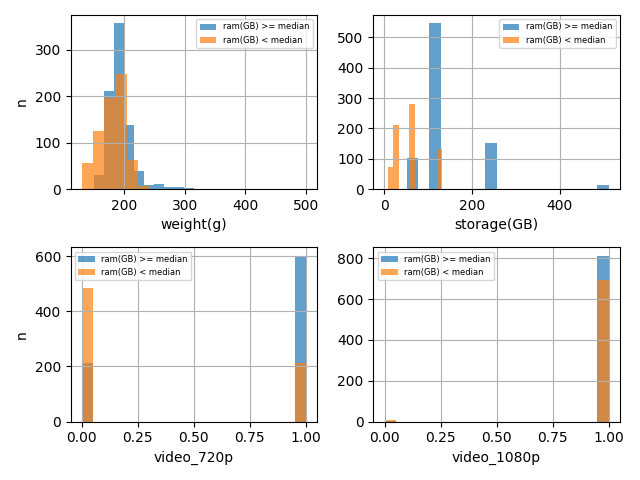


Рис. 17. Параметры

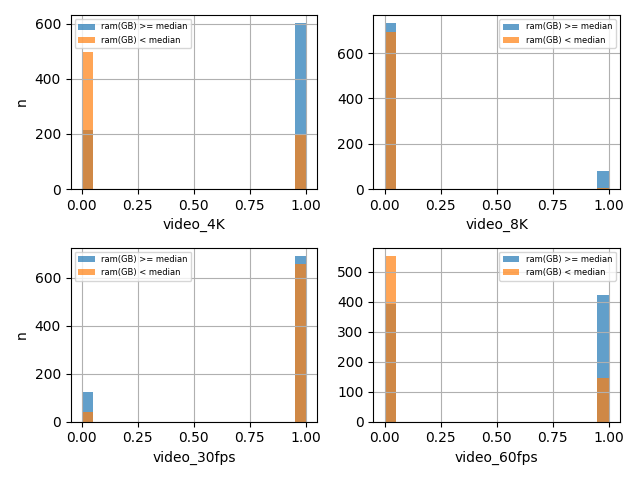


Рис. 18. Параметры

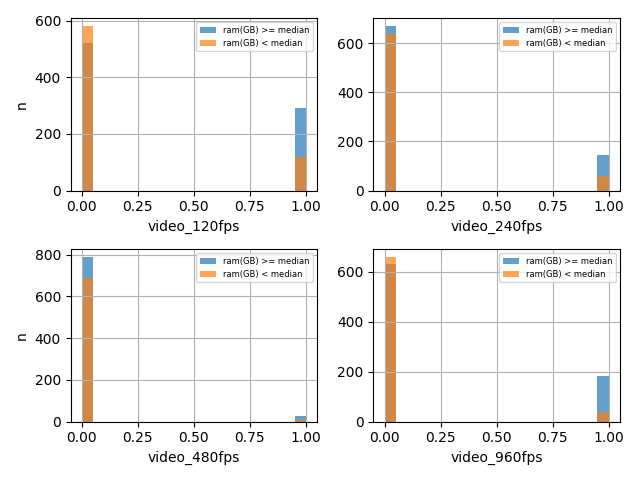


Рис. 19. Параметры

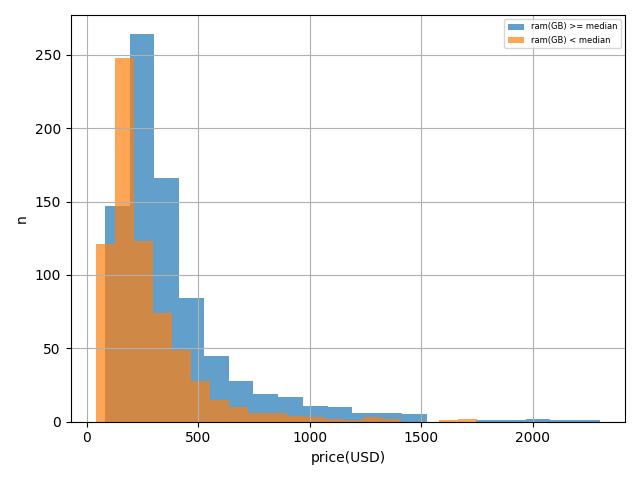


Рис. 20. Параметр price(USD)

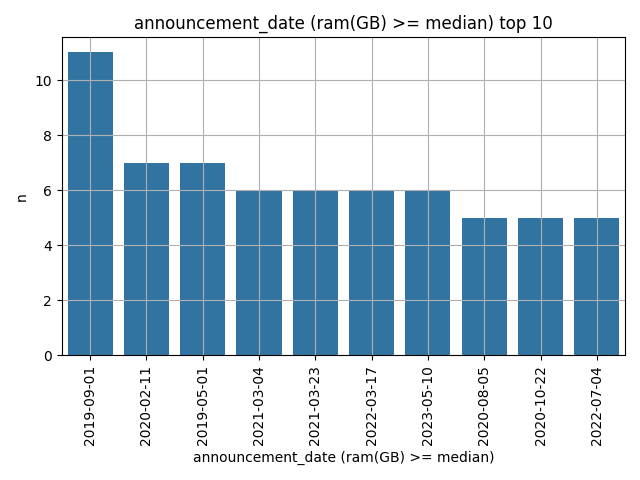


Рис. 21. Параметр

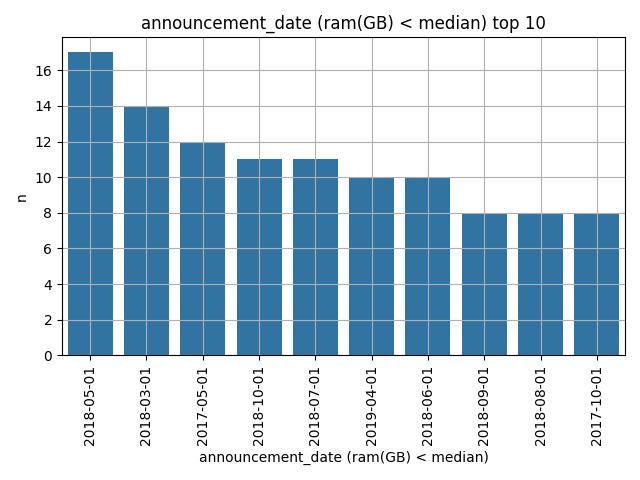


Рис. 22. Параметр

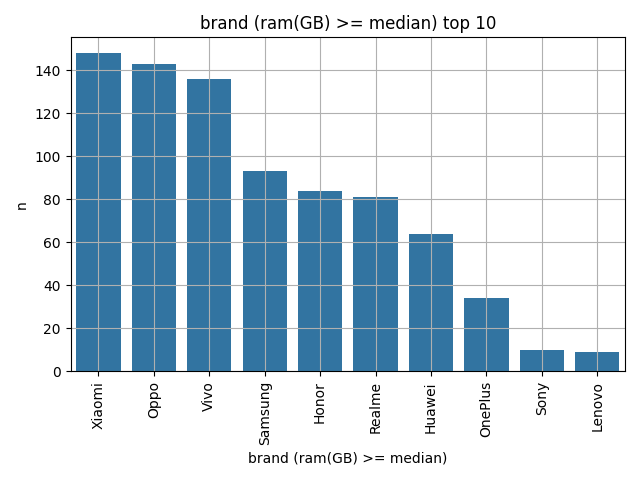


Рис. 23. Параметр

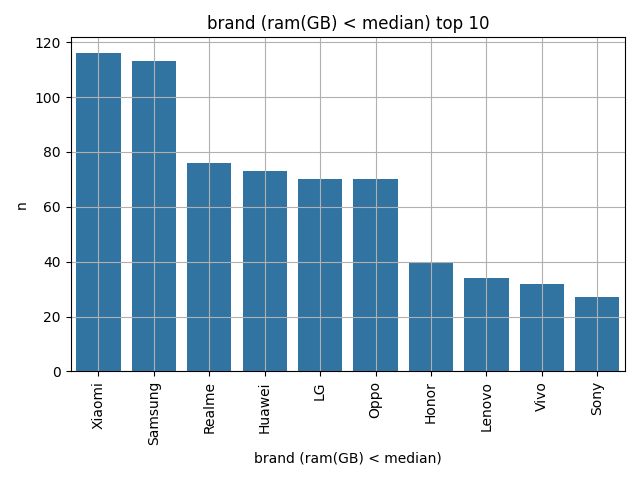


Рис. 24. Параметр

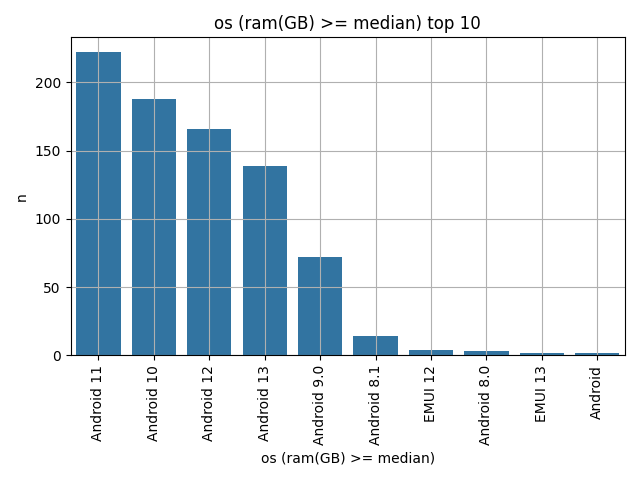


Рис. 25. Параметр

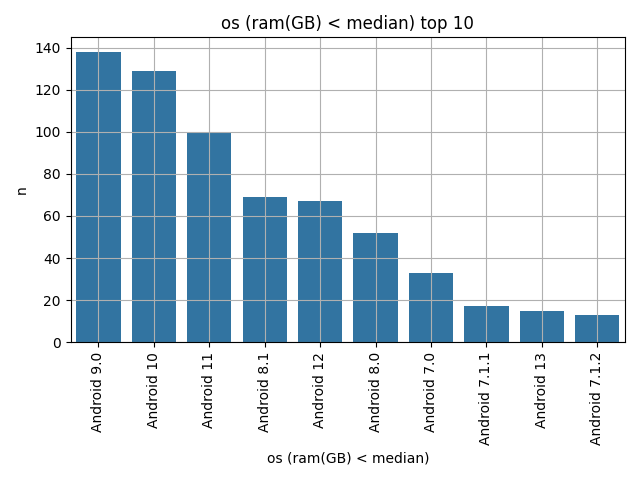


Рис. 26. Параметр

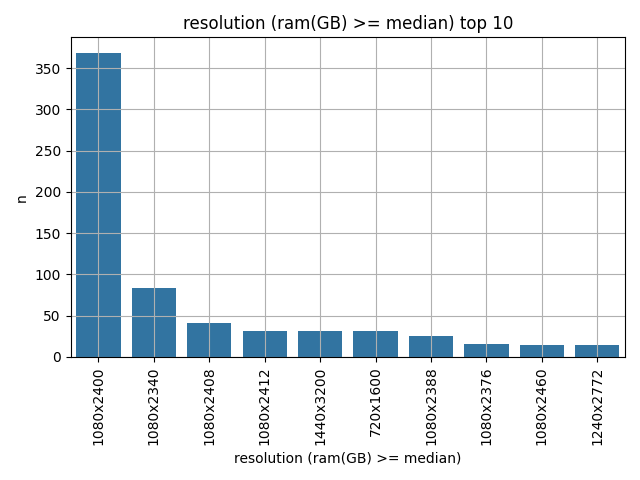


Рис. 27. Параметр

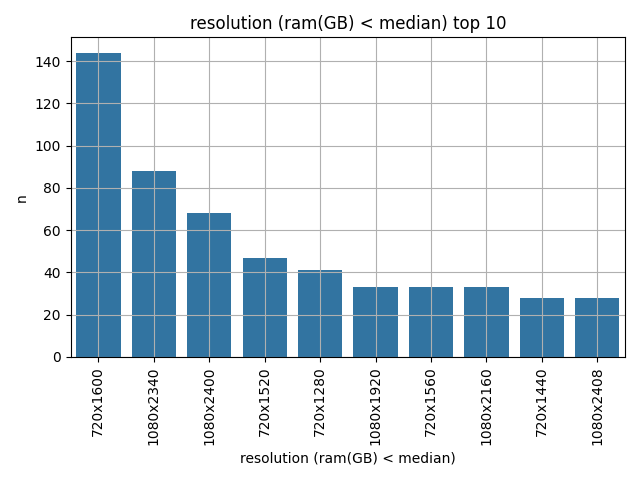


Рис. 28. Параметр

3.2. Гистограмма распределения, скатерограмма и боксплот параметров

Задание: построить гистограмму распределения, скатерограмму и боксплот параметров для объектов выше медианного значения в выборке и ниже. Подписать оси, добавить легенду и сетку на график.

Скатерограмма и боксплоты были построены по каждому столбцу относительно медианы столбца ram(GB).

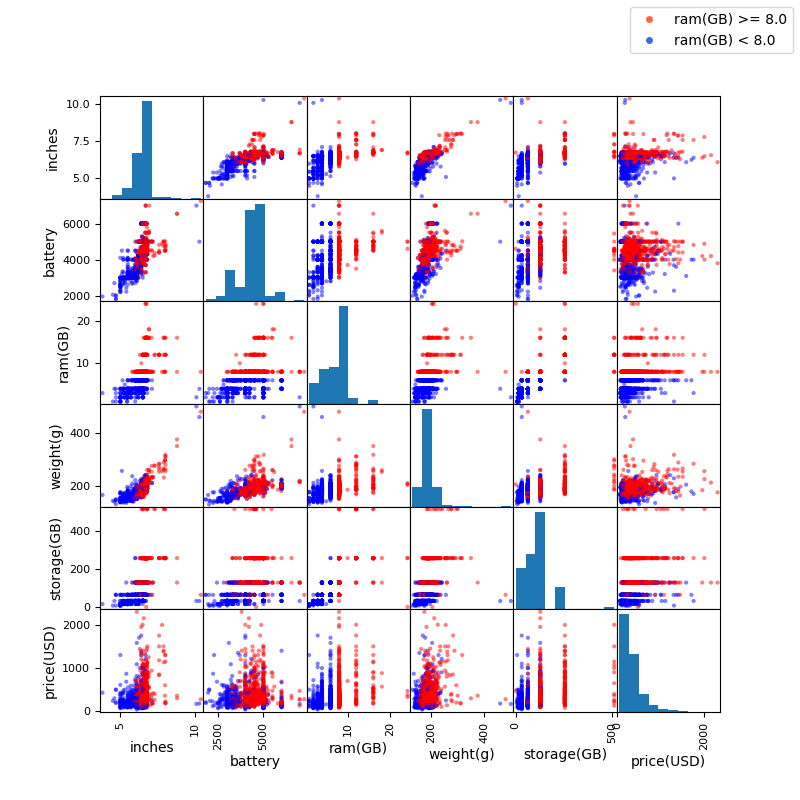


Рис. 29. Скатерограммы

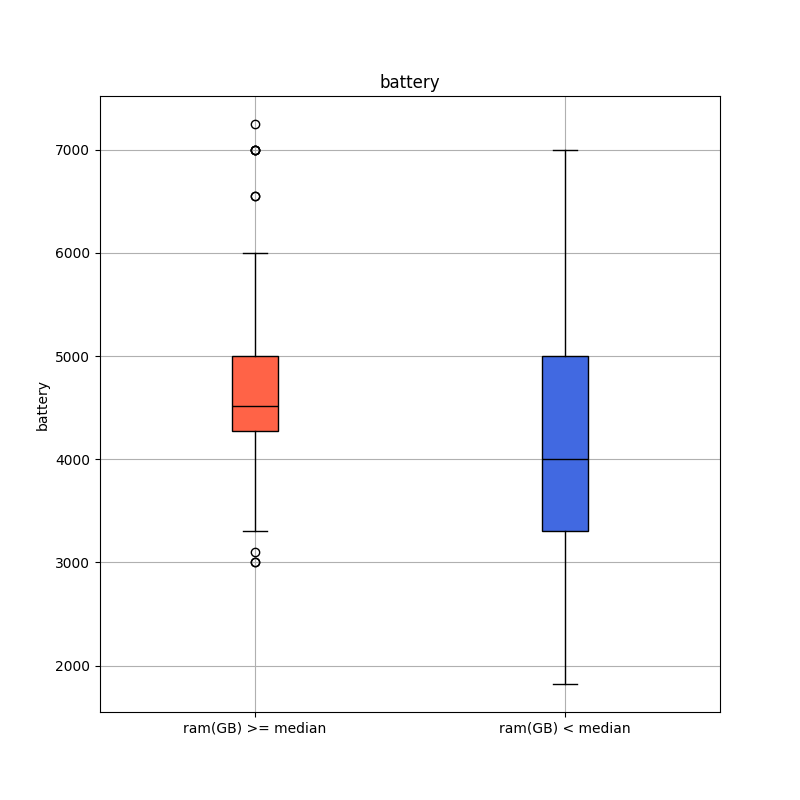


Рис. 30. Боксплот

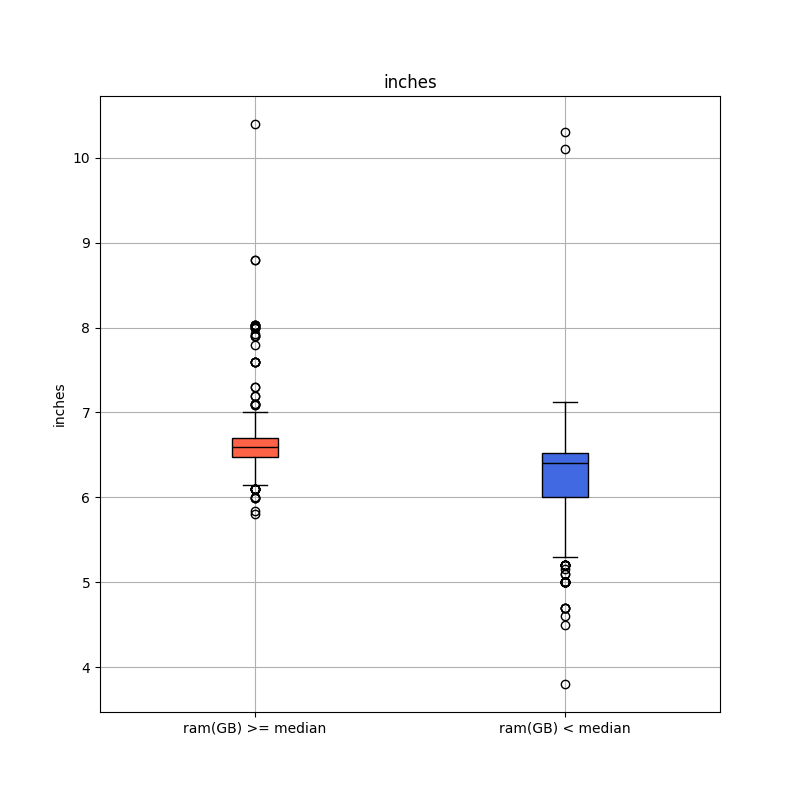


Рис. 31. Боксплот

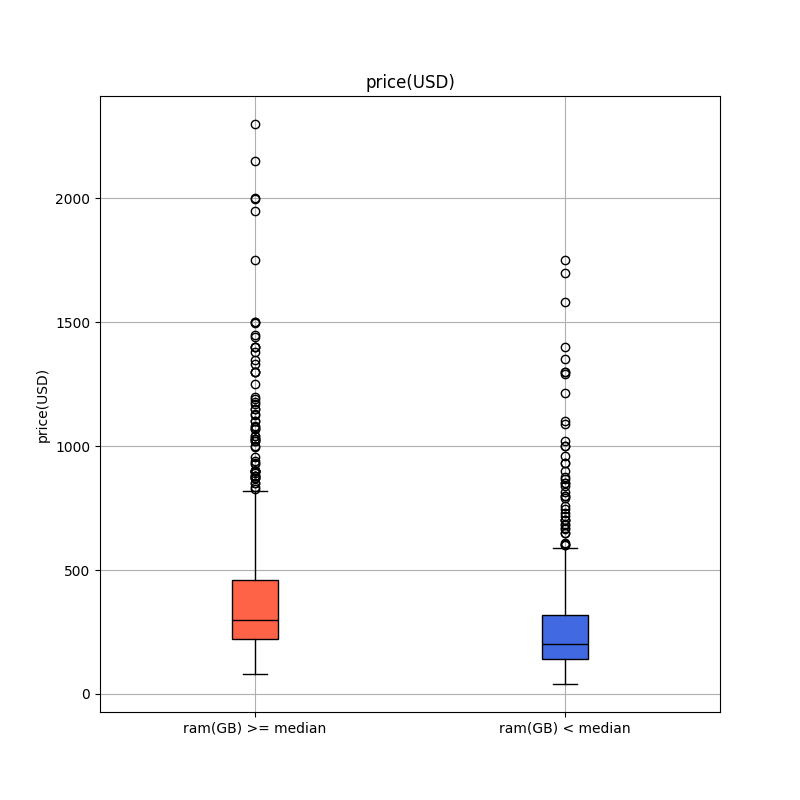


Рис. 32. Боксплот

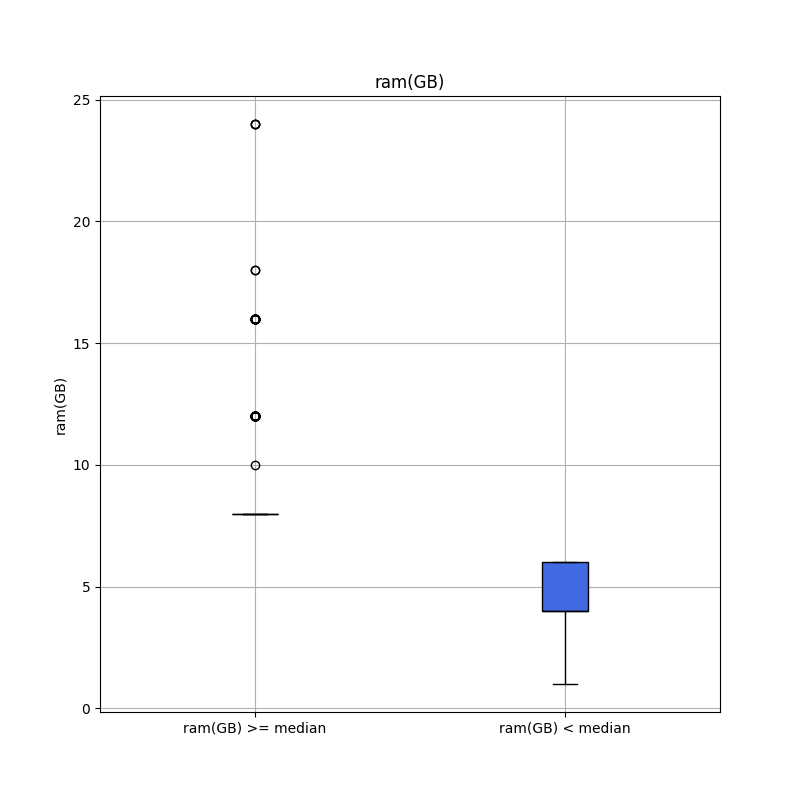


Рис. 33. Боксплот

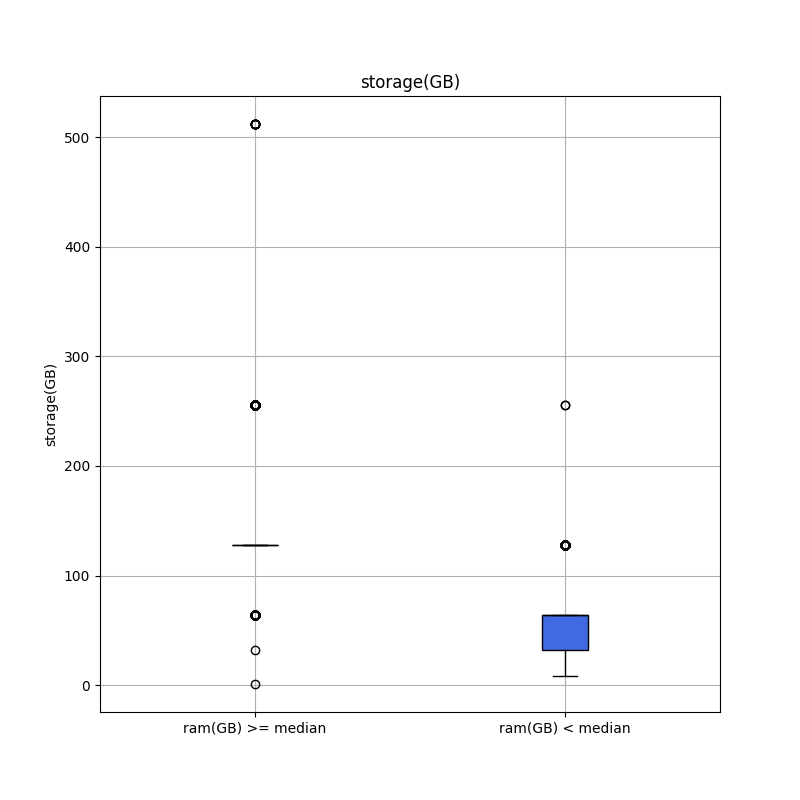


Рис. 34. Боксплот

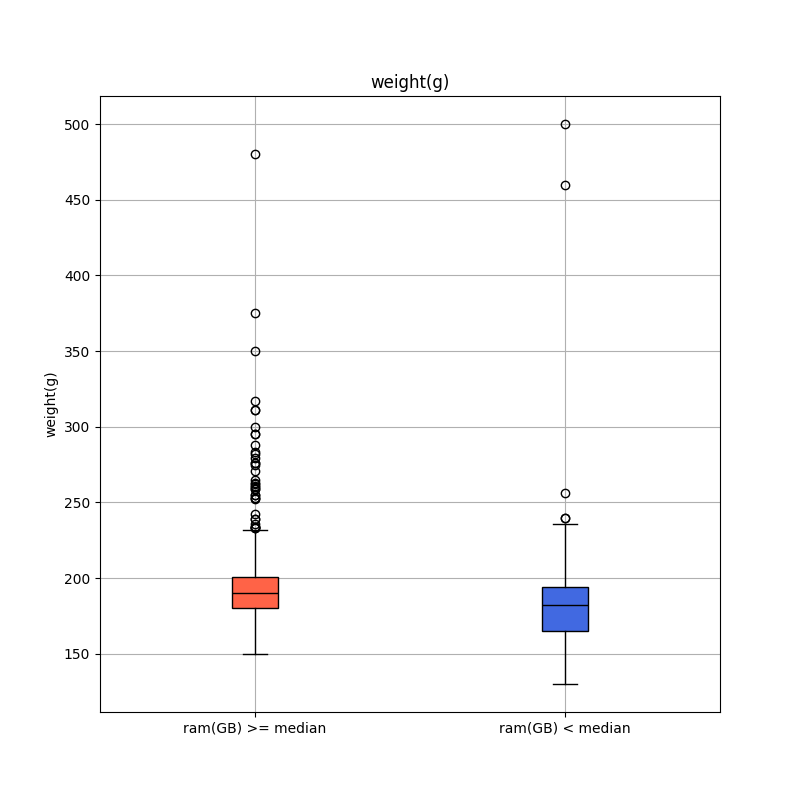


Рис. 35. Боксплот

4. Среднее значение и стандартное отклонение

Задание: рассчитать среднее значение и стандартное отклонение для параметров: для всей выборки и для каждого класса отдельно.

Данные разделены относительно медианы параметра ram(GB).

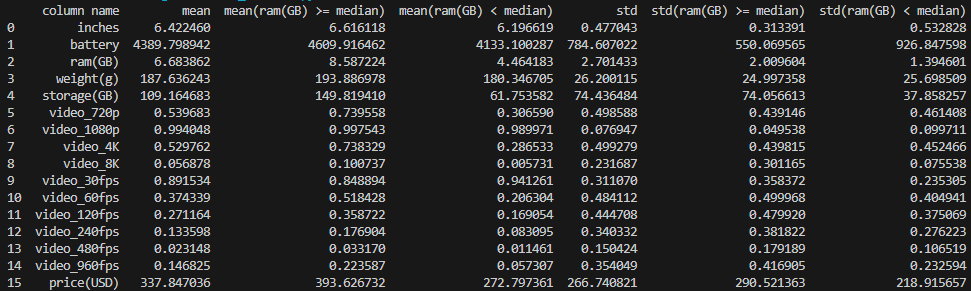


Рис. 36. Средние значения и стандартные отклонения

5. Различия в классах

5.1. Различия в классах по параметру n

Задание: статистически оценить различия в классах по параметру n- совпадают ли выборки по этому параметру или нет.

Данные разделены на классы по параметру price(USD).

Выбран параметр ram(GB).

Был проведен тест Шапиро-Уилка на принадлежность каждой выборки нормальному распределению. Выборки не распределены нормально.

Так как выборки не распределены нормально, был использован непараметрический тест Манна-Уитни, не требующий предположений о форме распределения данных.

Для выбранного параметра выборки не совпадают.

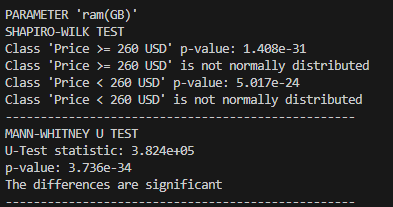


Рис. 37. Для параметра ram(GB).

5.2. Различия в классах по параметру m

Задание: статистически оценить различия в классах по параметру m - совпадают ли выборки по этому параметру или нет.

Данные разделены на классы по параметру price(USD).

Выбран параметр inches.

Был проведен тест Шапиро-Уилка на принадлежность каждой выборки нормальному распределению. Выборки не распределены нормально.

Так как выборки не распределены нормально, был использован непараметрический тест Манна-Уитни, не требующий предположений о форме распределения данных.

Для выбранного параметра выборки не совпадают.

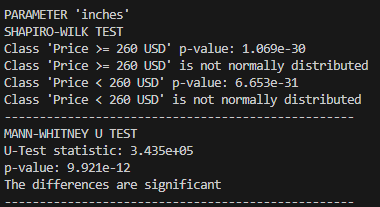


Рис. 38. Для параметра inches.

6. Классификация

Задание: классификация данных.

Был использован алгоритм LogisticRegression.

Данные разделены на классы по параметру price(USD), также данные разделены на обучающую и тестовую выборки в соотношении 7 к 3. Удалены параметры с данными, представленными текстом.

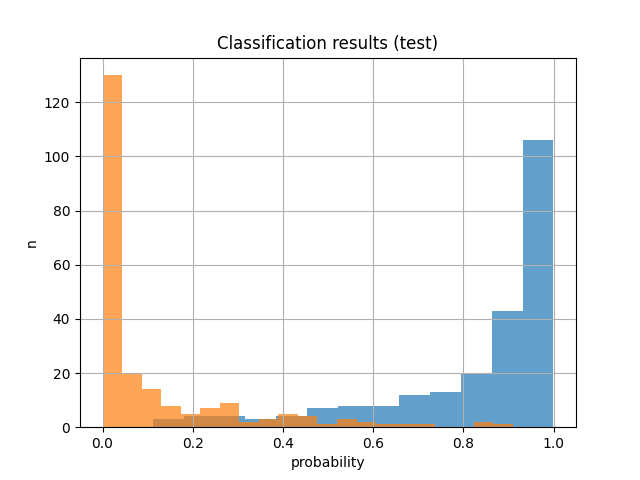


Рис. 39. Результаты классификации. Тестовая выборка

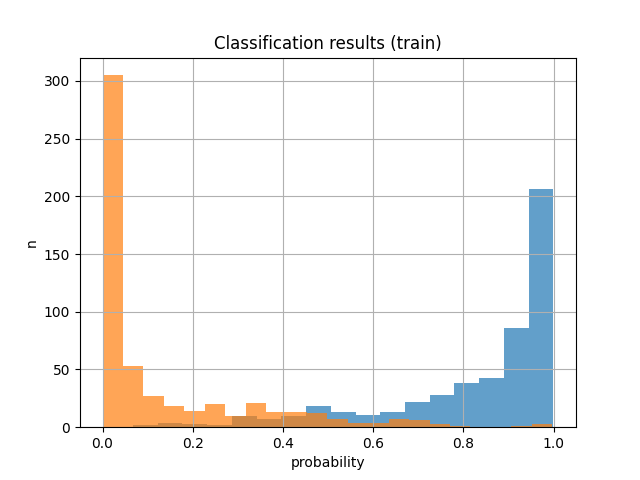


Рис. 40. Результаты классификации. Обучающая выборка

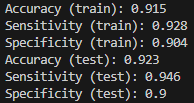


Рис. 41. Результаты классификации

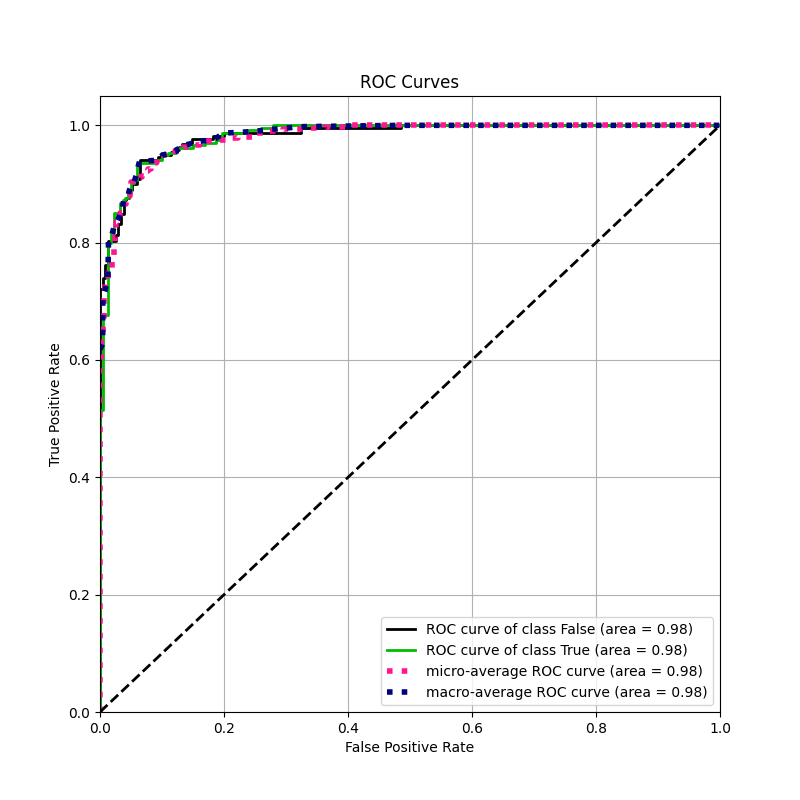


Рис. 42. ROC-кривая

Листинг

constants.py:

PRICE\_THRESHOLD = 260

LABEL\_POSITIVE = "Price >= " + str(PRICE\_THRESHOLD) + " USD"

LABEL\_NEGATIVE = "Price < " + str(PRICE\_THRESHOLD) + " USD"

IMAGE\_PATH = "images/"

read\_data.py:

import pandas as pd

def read\_data(bool\_to\_int: bool = True):

df = pd.read\_csv("cleaned\_all\_phones.csv")

column\_names = df.columns

if bool\_to\_int:

for i in range(len(column\_names)):

if df[column\_names[i]].dtype == bool:

df[column\_names[i]] = df[column\_names[i]].astype(int)

col\_investigated = df["price(USD)"]

return df, column\_names, col\_investigated

1\_plots.py:

import matplotlib.pyplot as plt

from plot\_data import \*

from read\_data import read\_data

from constants import \*

if \_\_name\_\_ == "\_\_main\_\_":

df, column\_names, col\_investigated = read\_data()

column\_names\_for\_barplots = ["brand", "os", "resolution", "announcement\_date"]

column\_names\_not\_for\_plot = ["phone\_name"]

plt.hist(col\_investigated, bins=20)

plt.xlabel(col\_investigated.name)

plt.ylabel('n')

plt.grid(True)

plt.savefig(IMAGE\_PATH + "hist\_" + col\_investigated.name)

plt.close("all")

print(df)

print("Price median:", df["price(USD)"].median())

positive = col\_investigated >= PRICE\_THRESHOLD

df\_for\_hists = df.drop(columns=["price(USD)", \*column\_names\_for\_barplots, \*column\_names\_not\_for\_plot])

build\_hists(df\_for\_hists, positive)

for i in range(len(column\_names\_for\_barplots)):

label = column\_names\_for\_barplots[i] + '(' + LABEL\_POSITIVE + ')'

df\_col\_positive = df[positive][column\_names\_for\_barplots[i]]

build\_bar\_plot(df\_col\_positive.value\_counts().head(10), label)

label = column\_names\_for\_barplots[i] + '(' + LABEL\_NEGATIVE + ')'

df\_col\_negative = df[positive == False][column\_names\_for\_barplots[i]]

build\_bar\_plot(df\_col\_negative.value\_counts().head(10), label)

2+3\_median.py:

from read\_data import read\_data

from constants import \*

if \_\_name\_\_ == "\_\_main\_\_":

df, column\_names, col\_investigated = read\_data()

positive = col\_investigated >= PRICE\_THRESHOLD

# col\_name = "inches"

col\_name = "ram(GB)"

print(col\_name + " median:", df[col\_name].median())

print(col\_name + " (class '" + LABEL\_POSITIVE + "') median:", df[positive][col\_name].median())

print(col\_name + " (class '" + LABEL\_NEGATIVE + "') median:", df[positive == False][col\_name].median())

4\_above\_below\_median\_hist.py:

from read\_data import read\_data

from constants import \*

from plot\_data import \*

if \_\_name\_\_ == "\_\_main\_\_":

df, column\_names, \_ = read\_data()

col\_name = "ram(GB)"

col = df[col\_name]

median = col.median()

column\_names\_for\_barplots = ["brand", "os", "resolution", "announcement\_date"]

column\_names\_not\_for\_plot = ["phone\_name"]

label\_c0 = col\_name + " >= median"

label\_c1 = col\_name + " < median"

above = col >= median

df\_for\_hists = df.drop(columns=["price(USD)", \*column\_names\_for\_barplots, \*column\_names\_not\_for\_plot])

build\_hists(df\_for\_hists, above, label\_c0, label\_c1)

for i in range(len(column\_names\_for\_barplots)):

label = column\_names\_for\_barplots[i] + " (" + label\_c0 + ")"

df\_col\_above = df[above][column\_names\_for\_barplots[i]]

build\_bar\_plot(df\_col\_above.value\_counts().head(10), label)

label = column\_names\_for\_barplots[i] + " (" + label\_c1 + ")"

df\_col\_below = df[above == False][column\_names\_for\_barplots[i]]

build\_bar\_plot(df\_col\_below.value\_counts().head(10), label)

plt.hist(df["price(USD)"][above], bins=20, alpha=0.7, label=label\_c0)

plt.hist(df["price(USD)"][above == False], bins=20, alpha=0.7, label=label\_c1)

plt.xlabel("price(USD)")

plt.ylabel("n")

plt.legend(prop={"size": 6})

plt.grid(True)

plt.tight\_layout()

path = IMAGE\_PATH + "hist\_price(USD)\_" + label\_c0 + "\_" + label\_c1

path = path.replace(">=", "bigger").replace("<", "smaller")

plt.savefig(path)

plt.close()

5\_above\_below\_median\_box.py:

from matplotlib.lines import Line2D

from read\_data import read\_data

from constants import \*

from plot\_data import \*

if \_\_name\_\_ == "\_\_main\_\_":

df, column\_names, \_ = read\_data(False)

col\_name = "ram(GB)"

col = df[col\_name]

median = col.median()

label\_above = col\_name + " >= median"

label\_below = col\_name + " < median"

labels = [label\_above, label\_below]

above = col >= median

colors = ['tomato', 'royalblue']

legend\_elements = \

[Line2D([0], [0], marker='o', color='w', markerfacecolor='tomato', markersize=6, label=f'{col\_name} >= {median}'),

Line2D([0], [0], marker='o', color='w', markerfacecolor='royalblue', markersize=6, label=f'{col\_name} < {median}')]

for name in df.columns:

if df[name].dtype not in ['object', 'bool']:

data = [df[above][name], df[above == False][name]]

fig, ax = plt.subplots(figsize=(8, 8))

ax.set\_ylabel(name)

bp = ax.boxplot(data, patch\_artist=True, tick\_labels=labels)

for patch, color in zip(bp['boxes'], colors):

patch.set\_facecolor(color)

for median in bp['medians']:

median.set\_color('black')

plt.title(name)

plt.grid()

plt.savefig(IMAGE\_PATH + f'boxplot\_{name}.png')

# plt.show()

5\_above\_below\_median\_scatter.py:

from matplotlib.lines import Line2D

from read\_data import read\_data

from constants import \*

from plot\_data import \*

if \_\_name\_\_ == "\_\_main\_\_":

df, column\_names, \_ = read\_data(False)

col\_name = "ram(GB)"

col = df[col\_name]

median = col.median()

column\_names\_for\_scatter = [df.columns[df.dtypes != 'bool']]

legend\_elements = \

[Line2D([0], [0], marker='o', color='w', markerfacecolor='tomato', markersize=6, label=f'{col\_name} >= {median}'),

Line2D([0], [0], marker='o', color='w', markerfacecolor='royalblue', markersize=6, label=f'{col\_name} < {median}')]

colors = df[col\_name].apply(lambda x: 'red' if x >= median else 'blue')

for name in column\_names\_for\_scatter:

pd.plotting.scatter\_matrix(df[name], color=colors, figsize=(8, 8))

plt.figlegend(handles=legend\_elements, loc='upper right', fontsize=10)

plt.savefig(IMAGE\_PATH + 'scatter\_matrix.png')

plt.show()

6\_above\_below\_median\_mean\_std.py:

from read\_data import read\_data

from constants import \*

from plot\_data import \*

if \_\_name\_\_ == "\_\_main\_\_":

df, column\_names, \_ = read\_data(False)

col\_name = "ram(GB)"

col = df[col\_name]

median = col.median()

label\_c0 = col\_name + " >= median"

label\_c1 = col\_name + " < median"

above = col >= median

table\_data = []

for name in column\_names:

if df[name].dtype not in ['object']:

mean = df[name].mean()

mean\_above = df[above][name].mean()

mean\_below = df[above == False][name].mean()

std = df[name].std()

std\_above = df[above][name].std()

std\_below = df[above == False][name].std()

table\_data.append([name, mean, mean\_above, mean\_below, std, std\_above, std\_below])

label\_c0 = '(' + label\_c0 + ')'

label\_c1 = '(' + label\_c1 + ')'

header = ['column name', 'mean', 'mean' + label\_c0, 'mean' + label\_c1, 'std', 'std' + label\_c0, 'std' + label\_c1]

df\_table = pd.DataFrame(table\_data, columns=header)

print(df\_table)

7\_classes\_differences.py:

import pandas as pd

from scipy import stats

from scipy.stats import mannwhitneyu

from read\_data import read\_data

from constants import \*

from plot\_data import \*

if \_\_name\_\_ == "\_\_main\_\_":

df, column\_names, col\_investigated = read\_data()

# col\_name = "ram(GB)"

col\_name = "inches"

label\_c0 = LABEL\_POSITIVE

label\_c1 = LABEL\_NEGATIVE

positive = col\_investigated >= PRICE\_THRESHOLD

col\_c0 = df[positive][col\_name]

col\_c1 = df[positive == False][col\_name]

print("PARAMETER '" + col\_name + "'")

stat, p = stats.shapiro(col\_c0)

print("SHAPIRO-WILK TEST")

print("Class '" + label\_c0 + "' p-value: " + "{:.3e}".format(p))

print("Class '" + label\_c0 + "' is " + ("" if p >= 0.05 else "not") + " normally distributed")

stat, p = stats.shapiro(col\_c1)

print("Class '" + label\_c1 + "' p-value: " + "{:.3e}".format(p))

print("Class '" + label\_c1 + "' is " + ("" if p >= 0.05 else "not") + " normally distributed")

print("-" \* 50)

print("MANN-WHITNEY U TEST")

u\_stat, p\_value = mannwhitneyu(col\_c0, col\_c1)

print("U-Test statistic: {:.3e}".format(u\_stat))

print("p-value: {:.3e}".format(p\_value))

if p\_value < 0.05:

print('The differences are significant')

else:

print('The differences are not significant')

print("-" \* 50)

8\_classification.py:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LogisticRegression

import scikitplot as skplt

from read\_data import read\_data

from constants import \*

from plot\_data import \*

def prepare\_data(c0: pd.DataFrame, c1: pd.DataFrame):

Y0 = np.zeros((c0.shape[0], 1), dtype=bool)

Y1 = np.ones((c1.shape[0], 1), dtype=bool)

X = np.vstack((c0, c1))

Y = np.vstack((Y0, Y1)).ravel()

# перемешиваем данные

rng = np.random.default\_rng()

arr = np.arange(X.shape[0])

rng.shuffle(arr)

X = X[arr]

Y = Y[arr]

return X, Y

if \_\_name\_\_ == "\_\_main\_\_":

df, \_, col\_investigated = read\_data(False)

column\_names\_dropped = []

for name in df.columns:

if df[name].dtype == 'object':

column\_names\_dropped.append(name)

df = df.drop(columns=column\_names\_dropped)

# print(df)

print(df.columns)

label\_c0 = LABEL\_POSITIVE

label\_c1 = LABEL\_NEGATIVE

positive = col\_investigated >= PRICE\_THRESHOLD

c0 = df[positive]

c1 = df[positive == False]

X, Y = prepare\_data(c0, c1)

N = X.shape[0]

# разделяем данные на 2 подвыборки

trainCount = round(0.7\*N)

Xtrain = X[0:trainCount]

Xtest = X[trainCount:N+1]

Ytrain = Y[0:trainCount]

Ytest = Y[trainCount:N+1]

# col\_name\_0 = 'inches'

# col\_name\_1 = 'battery'

# plt.scatter(c0[col\_name\_0], c0[col\_name\_1], marker=".", alpha=0.7)

# plt.scatter(c1[col\_name\_0], c1[col\_name\_1], marker=".", alpha=0.7)

# plt.title('Scatter')

# plt.xlabel(col\_name\_0)

# plt.ylabel(col\_name\_1)

# plt.show()

clf = LogisticRegression(random\_state=13, solver='saga').fit(Xtrain, Ytrain)

Pred\_test = clf.predict(Xtest) # Predict class labels for samples in X.

Pred\_test\_proba = clf.predict\_proba(Xtest) # Probability estimates

Pred\_train = clf.predict(Xtrain)

Pred\_train\_proba = clf.predict\_proba(Xtrain)

acc\_train = clf.score(Xtrain, Ytrain)

acc\_test = clf.score(Xtest, Ytest)

plt.hist(Pred\_train\_proba[Ytrain, 1], bins='auto', alpha=0.7)

plt.hist(Pred\_train\_proba[~Ytrain, 1], bins='auto', alpha=0.7) # т.к массив свероятностями имеет два столбца, мы берем один - первый

plt.title("Classification results (train)")

plt.xlabel("probability")

plt.ylabel('n')

plt.grid(True)

plt.savefig(IMAGE\_PATH + 'classification\_train' + '.png')

plt.show()

plt.hist(Pred\_test\_proba[Ytest,1], bins='auto', alpha=0.7)

plt.hist(Pred\_test\_proba[~Ytest,1], bins='auto', alpha=0.7) # т.к массив свероятностями имеет два столбца, мы берем один - первый

plt.title("Classification results (test)")

plt.xlabel("probability")

plt.ylabel('n')

plt.grid(True)

plt.savefig(IMAGE\_PATH + 'classification\_test' + '.png')

plt.show()

sensitivity\_test = 0

specificity\_test = 0

for i in range(len(Pred\_test)):

if Pred\_test[i]==True and Ytest[i]==True:

sensitivity\_test += 1

if Pred\_test[i]==False and Ytest[i]==False:

specificity\_test += 1

sensitivity\_test /= len(Pred\_test[Pred\_test==True])

specificity\_test /= len(Pred\_test[Pred\_test==False])

sensitivity\_train = 0

specificity\_train = 0

for i in range(len(Pred\_train)):

if Pred\_train[i]==True and Ytrain[i]==True:

sensitivity\_train += 1

if Pred\_train[i]==False and Ytrain[i]==False:

specificity\_train += 1

sensitivity\_train /= len(Pred\_train[Pred\_train==True])

specificity\_train /= len(Pred\_train[Pred\_train==False])

print('Accuracy (train):', round(acc\_train, 3))

print('Sensitivity (train):', round(sensitivity\_train, 3))

print('Specificity (train):', round(specificity\_train, 3))

print('Accuracy (test):', round(acc\_test, 3))

print('Sensitivity (test):', round(sensitivity\_test, 3))

print('Specificity (test):', round(specificity\_test, 3))

skplt.metrics.plot\_roc\_curve(Ytest, Pred\_test\_proba, figsize = (8, 8))

plt.grid(True)

plt.savefig(IMAGE\_PATH + 'RFC\_roc\_curve\_test' + '.png')

plt.show()

plot\_data.py:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from constants import \*

def add\_hist(axs, data: pd.DataFrame, positive, xlabel, ylabel, label\_c0=LABEL\_POSITIVE, label\_c1=LABEL\_NEGATIVE):

axs.hist(data[positive], bins=20, alpha=0.7, label=label\_c0)

axs.hist(data[positive == False], bins=20, alpha=0.7, label=label\_c1)

axs.set\_xlabel(xlabel)

axs.set\_ylabel(ylabel)

axs.legend(prop={"size": 6})

axs.grid(True)

def build\_hists\_2x2(data: pd.DataFrame, positive, fig\_n: int, label\_c0=LABEL\_POSITIVE, label\_c1=LABEL\_NEGATIVE):

if data.shape[1] != 4:

raise AssertionError("len(data) != 4")

column\_names = data.columns

fig, axs = plt.subplots(2, 2)

for i in range(4):

x = i // 2

y = i % 2

col = data[column\_names[i]]

xlabel = column\_names

Заключение

Гистограммы и графики распределения показали, что параметры, такие как цена, объем оперативной памяти и диагональ экрана, имеют различное распределение в зависимости от класса (выше или ниже медианного значения). Это указывает на наличие различий между классами. Медианы для параметров (например, диагональ экрана и объем оперативной памяти) различаются между классами, что подтверждает наличие различий в характеристиках телефонов с разной ценой. Тест Манна-Уитни показал, что выборки по параметрам (например, объем оперативной памяти и диагональ экрана) статистически значимо различаются между классами. Это означает, что данные параметры могут быть полезны для классификации телефонов по цене. Использование алгоритма логистической регрессии позволило достичь приемлемой точности классификации. ROC-кривая показала, что модель хорошо справляется с разделением классов, что подтверждает ее эффективность.

Список использованных источников

1. Датасет «Phone Prices», содержащий данные о смартфонах URL: <https://www.kaggle.com/datasets/berkayeserr/phone-prices>
2. Документация pandas URL: <https://pandas.pydata.org/docs/>
3. Документация Scikit-learn URL: [https://scikit-learn.org](https://scikit-learn.org/)
4. Документация Matplotlib URL: [https://matplotlib.org](https://matplotlib.org/)
5. Документация SciPy URL: [https://scipy.org](https://scipy.org/)
6. Документация Seaborn URL: [https://seaborn.pydata.org](https://seaborn.pydata.org/)
7. Документация Scikit-plot URL: [https://scikit-plot.readthedocs.io](https://scikit-plot.readthedocs.io/)