**Министерство науки и высшего образования Российской Федерации**

**федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Российский экономический университет имени Г. В. Плеханова»**

Кафедра информатики

**Выпускная квалификационная работа**

**по программе профессиональной переподготовке**

на тему «Прогнозирование цен на мобильные устройства с использованием методов машинного обучения»

Выполнила:

Петросян Шушаник Ашотовна

15.11д-БИТЦ10/21б

3 курс, ВШКМиС

Преподаватель:

ст. преп. Савинова Виктория Михайловна

Москва 2024

**1.** **Понимание бизнес-целей**

**1.1.** **Понимание бизнеса**

В наши дни мы смело можем отследить тенденцию возрастания цен на гаджеты российском рынке. С каждым днем повышается вероятность покупки и продажи техники по параллельному импорту. Возрастает конкуренция среди магазинов мобильных устройств. Сеть магазинов терпит убытки и сокращение прибыли в связи с ростом конкуренции в секторе и ростом цен поставщиков в связи с ситуацией в мире и скачков на валютном рынке.

В связи с предпосылками возникают следующие задачи:

1. Оптимальное ценообразование;

2. Анализ рынка;

3. Изучение конкурентов и их действий;

4. Анализ отклонений факта от плана продаж, выручки, прибыли.

5. Совершенствование ассортимента товаров (расширения ассортимента востребованных категорий и уменьшение не востребованный);

Ожидаемые преимущества: увеличение прибыли за счет привлечения клиентов.

Критерий успеха – увеличение продаж в результате оптимальной постановки цен.

**1.2.** **Доступные ресурсы**

Для успешной реализации проекта необходимы следующие категории специалистов: аналитик данных, бизнес-аналитик, специалист по базам данных, руководитель проекта.

Заказчик располагает всем необходимым оборудованием для проведения анализа данных.

**1.3.**  **Риски**

· Несоблюдение сроков проекта

· Риск неплатежеспособности заказчика

· Риск нехватки и неполноты данных

· Риск несоответствия полученных результатов требованиям заказчика

**1.4.**  **Ограничения**

Ограничение сроков: 6 месяцев.

Ставки по сотрудникам:

Аналитик данных – 1 ставка;

Бизнес-аналитик – 1 ставка.

**1.5.** **Цели исследования данных**

Задачи анализа данных, решаемые в рамках проекта:

1. Построение визуализации рентабельности продаж товаров, характеристик потребителей, недельные колебания продаж, сравнительного анализа поставщиков, аналитики каналов сбыта.

2. Решение задачи прогнозирования продаж, выручки, прибыли с использованием моделей регрессии. В качестве моделей регрессии будут рассмотрены методы: линия тренда, метод ближайшего соседа, нейронная сеть

**1.6.Критерии успешности изучения данных.**

Метрики оценки точности и качества построенных моделей:

Для моделей регрессии: качество модели определяется с использованием коэффициента детерминации (R2), точность модели определяется на основании средней относительной ошибки (MAPE).

Границы значений метрик: R2 должен быть больше либо равен 0.8, MAPE не более 10%.

**2.** **Начальное изучение данных.**

**2.1.** **Сбор данных.**

**Внутренние данные** – Номер продукта, Цена, Скидка, Вес, Разрешние, Пиксели на дюйм, Процессорное ядро, Частота процессора, Внутренняя память, Оперативная память, Задняя камера, Фронтальная камера, Батарея, Толщина .<https://www.kaggle.com/datasets/mohannapd/mobile-price-prediction>

**Внешние данные** –не требуются

**2.2.** **Описание данных**

Объем данных –17,7 кб

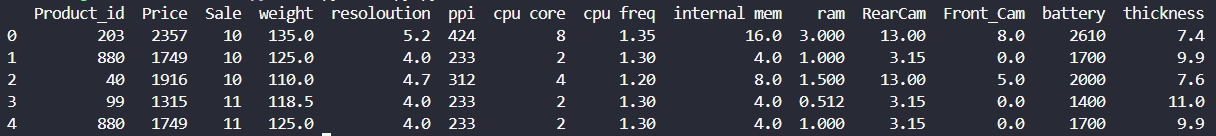
Типы, виды данных и схемы кодирования

| Наименование | Тип данных | Вид данных | Схема кодирования |
| --- | --- | --- | --- |
| Product\_id | Целое число | Дискретный | Целое число |
| Price | Целое число | Непрерывный | Целое число |
| Sale | Целое число | Непрерывный | Целое число |
| Weight | Число с плавающей точкой | Дискретный | Целое число |
| Resolоution | Число с плавающей точкой | Дискретный | Целое число |
| PPI | Целое число | Дискретный | Целое число |
| CPU core | Целое число | Дискретный | Целое число |
| CPU freq | Число с плавающей точкой | Дискретный | Целое число |
| Internal mem | Число с плавающей точкой | Дискретный | Целое число |
| RAM | Число с плавающей точкой | Дискретный | Целое число |
| RearCam | Число с плавающей точкой | Дискретный | Целое число |
| Front\_Cam | Число с плавающей точкой | Дискретный | Целое число |
| Battery | Целое число | Дискретный | Целое число |
| Thickness | Число с плавающей точкой | Дискретный | Целое число |

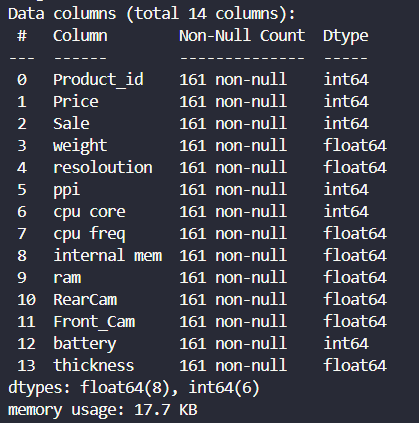
Формат данных – файл csv, разделитель – “,”.

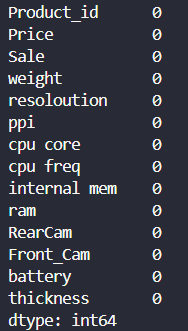
**2.3. Исследование данных**

Построение описательной статистики



Проверка пропущенных значений: пропущенные значения отсутствуют.

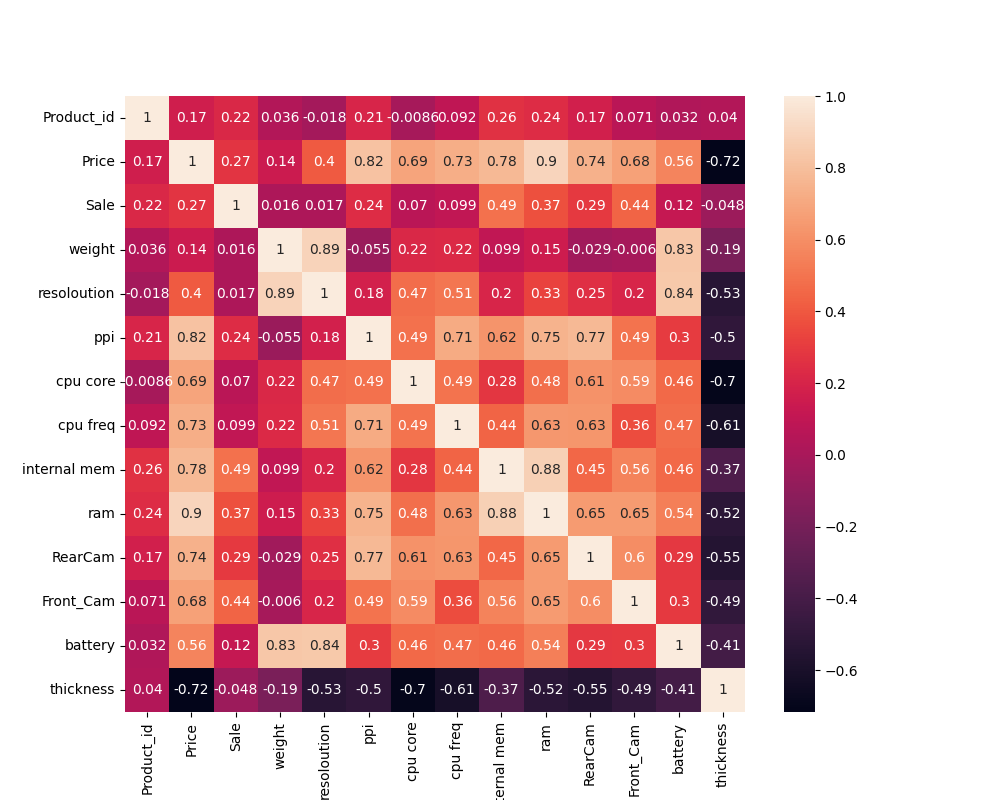


****

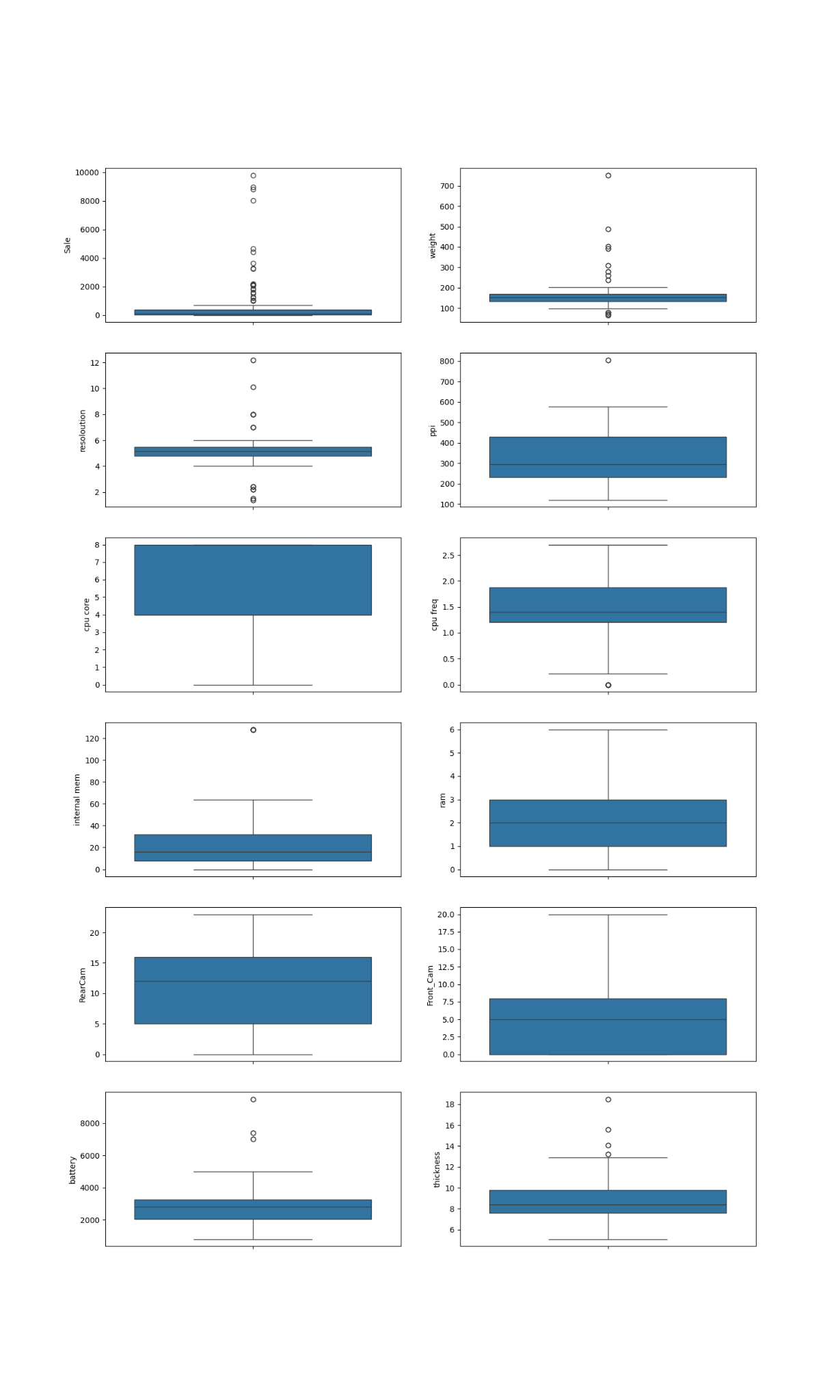
Проверка дубликатов: дубликаты отсутствуют

****

Построение матрицы корреляций данных:

****

Проверка на экстремальные выбросы.

****

Как можно заметить присутствуют выбросы, они будут очищены на этапе подготовки данных.

Также при построении диаграмм рассеяния было обнаружено, что чем больше cpu freq и battery тем больше цена

**3.** **Подготовка данных**

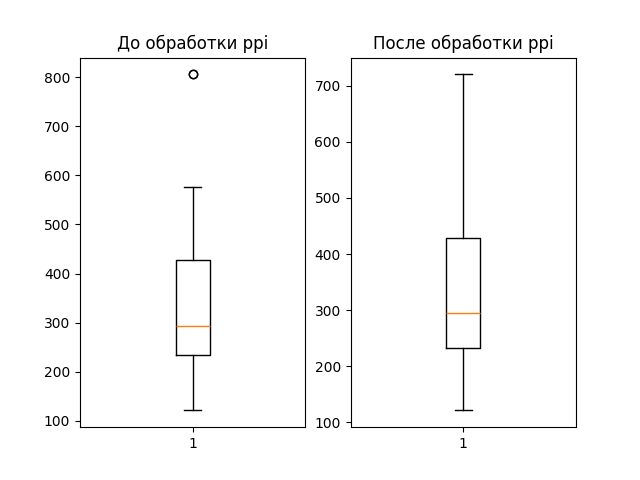
Был проведен анализ на наличие выбросов на основе ящиков с усами, представленных в прошлой главе и вот решение:

* Product\_id, cpu core, ram, RearCam, Front\_cam: В этих признаках выбросов нет.
* Sale было принято не очищать от выбросов
* Weight: Необходимо заменить значения, меньшие нижней границы, на нижнюю границу, а значения, большие верхней границы, на верхнюю границу.
* Resolution: Необходимо заменить значения, меньшие нижней границы, на нижнюю границу, а значения, большие верхней границы, на второе по величине значение среди выбросов.
* ppi: Необходимо заменить выбросы на верхнюю границу.
* cpu freq: Необходимо заменить значение 0 на 0.5, потому что минимальная частота процессора должна быть не менее 0.5.
* internal mem. battery: Выбросы не нужно удалять, так как это могут быть реальные значения.

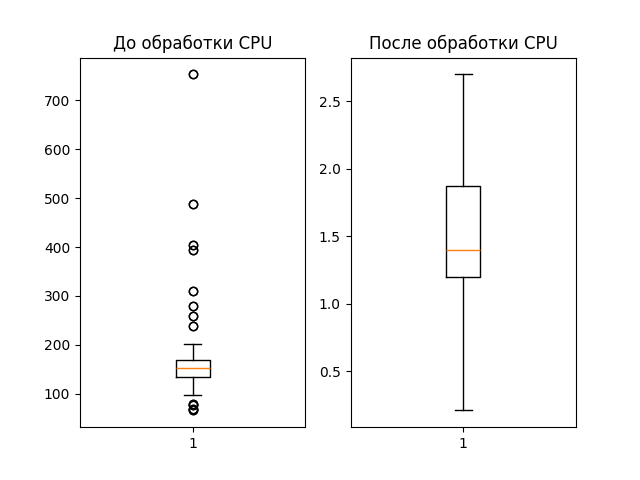
Признак "Product\_id" отбрасываем, на результат он влияют очень слабо, что видно по корреляционной матрице.

Данные не требуют очистки, так как не выявлено дубликатов, пропусков, противоречий и экстремальных значений.

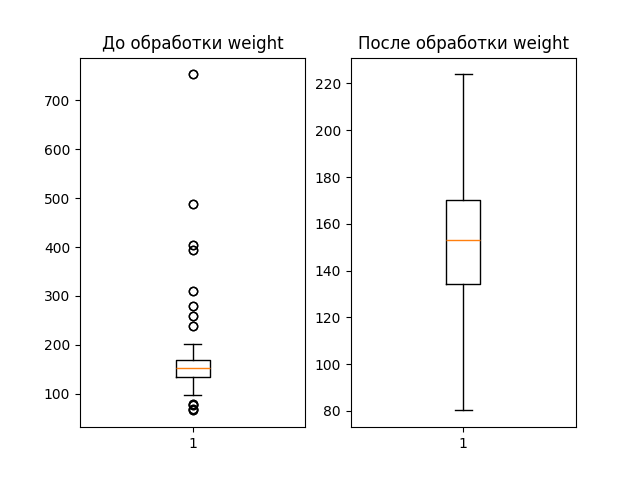
1. Очищение выбросов ppi



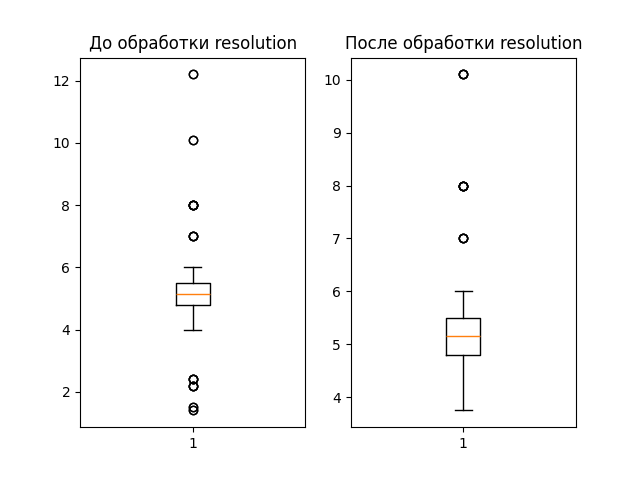
1. Обработка выбросов CPU



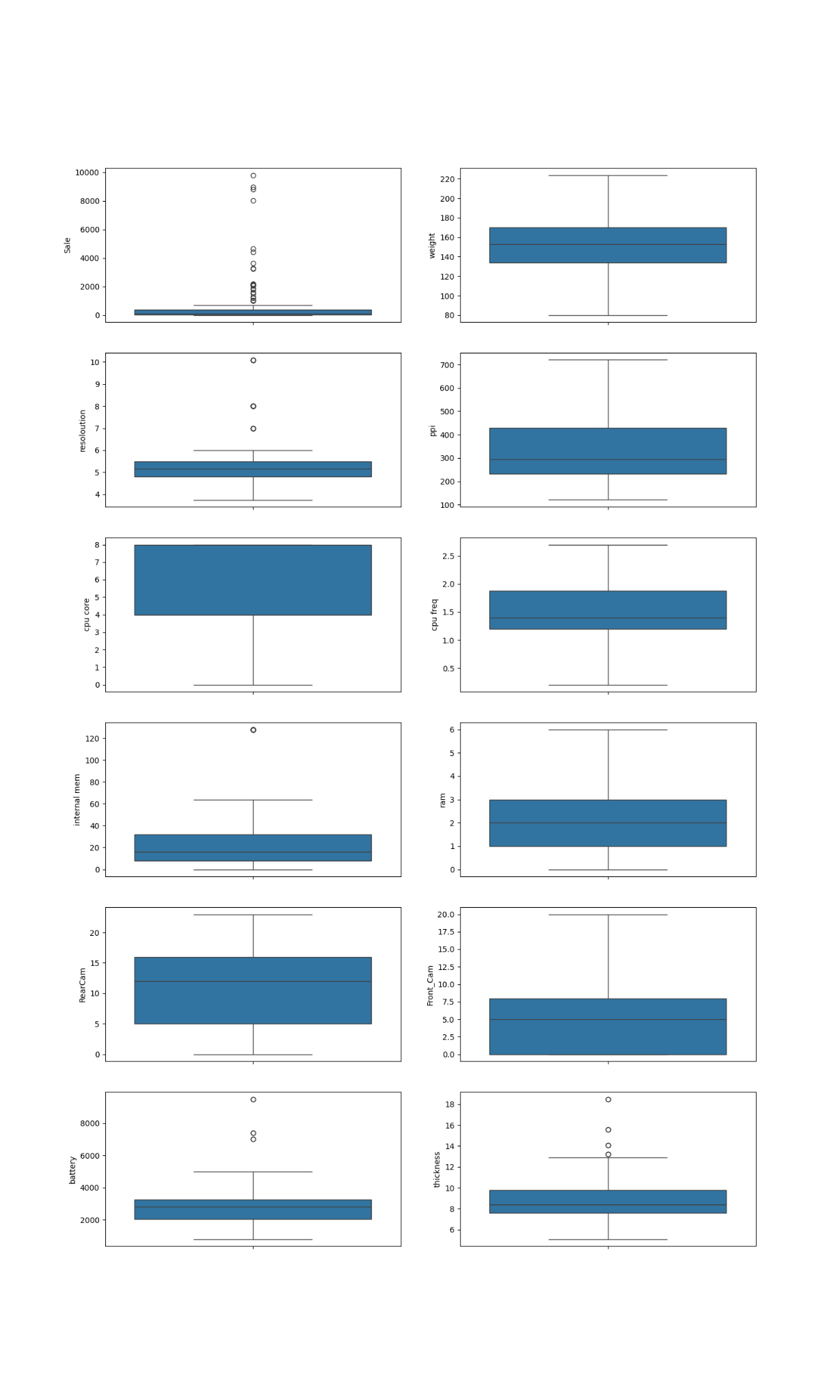
1. Обработка weight



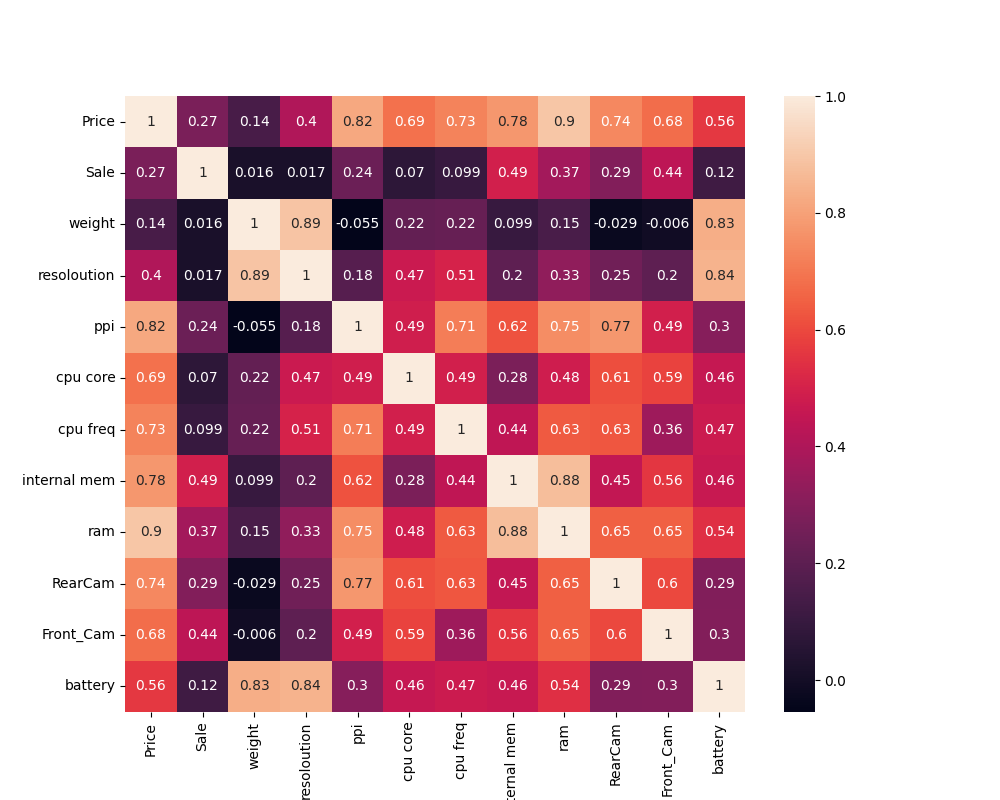
1. Обработка resolution



Общая картина после очистки выбросов:



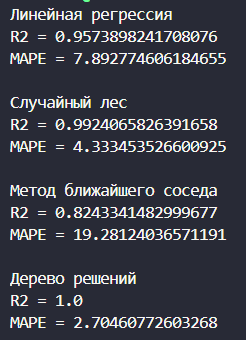
Корреляция некоторых данных увеличилась и теперь схема более наглядная, за счёт отсутствия колонок, слабо связанных с другими:

****

Разбиваем на тестовую и обучающую выборки.

**4. Моделирование**

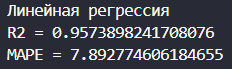
Результаты моделей с данными, которые не прошли нормализацию и стандартизацию

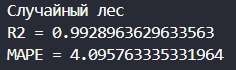
****

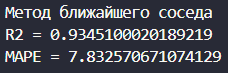
Вот методы которые были использованы для улучшения показателей R2 и MAPE:

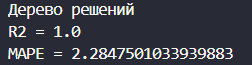
* Дерево решений – стандартизация
* Случайный лес – нормализация и стандартизация
* Метод ближайшего соседа – нормализация

На значения линейной регрессии данные изменения не оказывали значительного влияния

****

****

****

****

Предсказание случайного леса:

Actual Values Predicted Values

0 2859 2884.74

1 3316 3118.10

2 754 765.91

3 1396 1600.34

4 3260 3121.75

5 2006 1942.74

6 3287 3130.84

7 2654 2611.02

8 1921 1857.25

9 2580 2506.58

10 2714 2565.63

11 1511 1540.74

12 1831 1905.93

13 1238 1347.96

14 2409 2503.19

15 2276 2337.07

16 1676 1908.38

17 791 828.78

18 1396 1600.34

19 3211 3340.60

Actual Values Predicted Values

20 1522 1511.70

21 1950 1728.31

22 2392 2396.54

23 2001 2062.62

24 1810 1916.34

25 1984 1884.83

26 2137 2268.79

27 628 673.59

28 1831 1908.94

29 2258 2242.90

30 2977 2962.31

31 3658 3402.98

32 2124 2173.90

33 1810 1916.34

34 2343 2432.83

35 1843 1726.26

36 2466 2564.20

37 1347 1514.36

38 1302 1339.34

39 2536 2606.86

40 1646 1621.4

1. **Оценка результатов**

На основе представленных показателей, дерево решений и случайный лес демонстрируют наилучшие результаты. . Однако, перед тем, как применить их на практике, необходимо внимательно изучить риски переобучения и оценить их способность справляться с новыми данными.

1. **Внедрение**

Дальнейшее использование этих моделей достаточно субъективно и зависит от конкретных бизнес-задач, либо возможно объединение всех результатов для извлечения выгоды из данных. Однако эти данные могут быть полезны для прогнозирования роста цен.

**Приложение 1. Код программы Python для проведенного анализа**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import MinMaxScaler

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

from sklearn.neighbors import KNeighborsRegressor

from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import mean\_absolute\_percentage\_error

df=pd.read\_csv('./Cellphone.csv')

# Матрица корреляций

plt.figure(*figsize*=(10,8))

sns.heatmap(df.corr(),*annot*=True)

plt.savefig('matrix.png')

# Удаление ненужных колонки

df = df.drop(["Product\_id"], *axis*=1)

df = df.drop(["thickness"], *axis*=1)

# Много усов 1

nrows=6

ncols=2

fig,axes=plt.subplots(*nrows*=nrows,*ncols*=ncols,*figsize*=(15,25))

for i , col in enumerate(df.drop(['Price'],*axis*=1).columns):

sns.boxplot(df[col],*ax*=axes[i//ncols,i%ncols])

plt.savefig('usi1.png')

# Обзор

df.head()

# Проверка пустых значений

df.isnull().sum()

# Проверка дублирующихся значений

df.duplicated().sum()

# Матрица корреляций

plt.figure(*figsize*=(10,8))

sns.heatmap(df.corr(),*annot*=True)

plt.savefig('matrix.png')

# Обработка ppi

n1,n2=np.percentile(df['ppi'],[25,75])

iqr=n2-n1

lower\_bound=n1-(1.5\*iqr)

upper\_bound=n2+(1.5\*iqr)

plt.subplot(1,2,1)

plt.boxplot(df['ppi'])

plt.title('До обработки ppi')

df['ppi']= np.where(df['ppi']>upper\_bound, upper\_bound,df['ppi'])

plt.subplot(1,2,2)

plt.boxplot(df['ppi'])

plt.title('После обработки ppi')

plt.savefig('ppi.png')

# Обработка cpu

n1,n2=np.percentile(df['cpu freq'],[25,75])

iqr=n2-n1

lower\_bound=n1-(1.5\*iqr)

upper\_bound=n2+(1.5\*iqr)

plt.subplot(1,2,1)

plt.boxplot(df['weight'])

plt.title('До обработки CPU')

df['cpu freq']= np.where(df['cpu freq']< lower\_bound,0.5,df['cpu freq'])

plt.subplot(1,2,2)

plt.boxplot(df['cpu freq'])

plt.title('После обработки CPU')

plt.savefig('CPU.png')

# weight

n1,n2=np.percentile(df['weight'],[25,75])

iqr=n2-n1

lower\_bound=n1-(1.5\*iqr)

upper\_bound=n2+(1.5\*iqr)

plt.subplot(1,2,1)

plt.boxplot(df['weight'])

plt.title('До обработки wight')

df['weight']= np.where(df['weight']>upper\_bound, upper\_bound,np.where(df['weight']< lower\_bound,lower\_bound,df['weight']))

plt.subplot(1,2,2)

plt.boxplot(df['weight'])

plt.title('После обработки wight')

plt.savefig('weight.png')

# resolution

n1,n2=np.percentile(df['resoloution'],[25,75])

iqr=n2-n1

lower\_bound=n1-(1.5\*iqr)

upper\_bound=n2+(1.5\*iqr)

plt.subplot(1,2,1)

plt.boxplot(df['resoloution'])

plt.title('До обработки resolution')

df['resoloution']= np.where(df['resoloution']>=10.1, 10.1,np.where(df['resoloution']< lower\_bound,lower\_bound,df['resoloution']))

plt.subplot(1,2,2)

plt.boxplot(df['resoloution'])

plt.title('После обработки resolution')

plt.savefig('resolution.png')

# Много усов 2

nrows=6

ncols=2

fig,axes=plt.subplots(*nrows*=nrows,*ncols*=ncols,*figsize*=(15,25))

for i , col in enumerate(df.drop(['Price'],*axis*=1).columns):

sns.boxplot(df[col],*ax*=axes[i//ncols,i%ncols])

plt.savefig('usi2.png')

# Разделение на тестовую и обучающую выборки

y = df.Price

x = df.drop("Price", *axis*=1)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, *test\_size* = 0.25, *random\_state*=1)

# Нормализация

minmax\_scaler = MinMaxScaler()

minmax\_scaler.fit(x\_train)

x\_train = pd.DataFrame(minmax\_scaler.transform(x\_train), *columns*=x\_train.columns)

x\_test = pd.DataFrame(minmax\_scaler.transform(x\_test), *columns*=x\_train.columns)

# Стандартизация

standard\_scaler = StandardScaler()

standard\_scaler.fit(x\_train)

x\_train = pd.DataFrame(standard\_scaler.transform(x\_train), *columns*=x\_train.columns)

x\_test = pd.DataFrame(standard\_scaler.transform(x\_test), *columns*=x\_train.columns)

# Преобразование данных в массивы Numpy

x\_train = np.array(x\_train)

y\_train = np.array(y\_train)

x\_test = np.array(x\_test)

y\_test = np.array(y\_test)

# Линейная регрессия

print("Линейная регрессия")

Model = LinearRegression()

Model.fit(x\_train, y\_train)

R2 = Model.score(x\_train, y\_train)

ypredict = Model.predict(x\_test)

mape = mean\_absolute\_percentage\_error(y\_test, ypredict)\*100

print('R2 =', R2)

print('MAPE =', mape, *end*="\n\n")

# Случайный лес

print("Случайный лес")

Model = RandomForestRegressor()

Model.fit(x\_train,y\_train)

R2 = Model.score(x\_train, y\_train)

ypredict=Model.predict(x\_test)

mape = mean\_absolute\_percentage\_error(y\_test, ypredict)\*100

print('R2 =', R2)

print('MAPE =', mape, *end*="\n\n")

# Предсказания случайного леса

res=pd.DataFrame({'Actual Values':y\_test,

'Predicted Values':ypredict})

print(res)

# Метод ближайшего соседа

print("Метод ближайшего соседа")

Model = KNeighborsRegressor()

Model.fit(x\_train, y\_train)

ypredict = Model.predict(x\_test)

R2 = Model.score(x\_train, y\_train)

mape = mean\_absolute\_percentage\_error(y\_test, ypredict)\*100

print('R2 =', R2)

print('MAPE =', mape, *end*="\n\n")

# Дерево решений

print("Дерево решений")

Model = DecisionTreeRegressor()

Model.fit(x\_train, y\_train)

ypredict = Model.predict(x\_test)

R2 = Model.score(x\_train, y\_train)

mape = mean\_absolute\_percentage\_error(y\_test, ypredict)\*100

print('R2 =', R2)

print('MAPE =', mape, *end*="\n\n")