Федеральное государственное автономное

образовательное учреждение высшего образования

«Пермский государственный национальный исследовательский университет» (ПГНИУ)

Региональный институт непрерывного образования (РИНО ПГНИУ)

Цифровая кафедра

Выпускная аттестационная (квалификационная) работа

по курсу профессиональной переподготовки

«Анализ данных и машинное обучение»

**ОПРЕДЕЛЕНИЕ ЦЕНЫ НОУТБУКА**

**В ЗАВИСИМОСТИ ОТ ЕГО ХАРАКТЕРИСТИК**

Разработчики проекта:

Заякин Евгений Анатольевич,

Хрулева Яна Валерьевна

Пермь, 2024

**Оглавление**

[ПАСПОРТ ПРОЕКТА 3](#_heading=h.30j0zll)

[СОДЕРЖАНИЕ ПРОЕКТА 4](#_heading=h.1fob9te)

[Анализ проблемы исследования 4](#_heading=h.3znysh7)

[Исходные данные 7](#_heading=h.2et92p0)

[Реализация проекта 8](#_heading=h.tyjcwt)

[Этап 1. Подготовка данных к анализу 8](#_heading=h.3dy6vkm)

[Этап 2. Предварительный анализ данных 12](#_heading=h.1t3h5sf)

[Этап 3. Корреляционный анализ данных 21](#_heading=h.4d34og8)

[Этап 4. Моделирование 25](#_heading=h.2s8eyo1)

[Этап 5. Прогнозирование 29](#_heading=h.17dp8vu)

[Заключение 30](#_heading=h.3rdcrjn)

[Список использованных источников и литературы 31](#_heading=h.26in1rg)

[Приложения 33](#_heading=h.lnxbz9)

**ПАСПОРТ ПРОЕКТА**

**Название проекта:** Определение цены ноутбука в зависимости от его характеристик.

**Сведения об авторах:**Заякин Евгений Анатольевич, Хрулева Яна

Валерьевна

**Цель:** выполнить анализ данных о ноутбуках и построить модель зависимости цены ноутбука от различных факторов с помощью линейной регрессии, позволяющую делать прогнозы с хорошей точностью.

**Задачи:**

1. Выполнить анализ проблемы, обосновать ее актуальность.
2. Осуществить загрузку данных и подготовку их к анализу количественными методами, включая устранение пропущенных значений.
3. Выполнить предварительный анализ данных, в том числе выявление и обработку выбросов, проверку распределения данных на нормальность, корреляционный анализ.
4. Осуществить моделирование зависимости целевого признака (цены ноутбука) от факторных методом линейной регрессии, в том числе подобрать наилучшую модель, оценить ее качество и выполнить прогнозирование.
5. Выполнить интерпретацию полученных результатов и сделать выводы о достижении цели.

**Краткое описание проекта:**

Требуется проанализировать данные о ноутбуках и определить, возможно ли описать зависимость их цены от имеющихся характеристик (процессор, объем оперативной памяти, тип экрана и т.д.) методом линейной регрессии. Сделать выводы.

**Конкретные ожидаемые результаты:**

* Построенная регрессионная модель, описывающая зависимость цены ноутбука от его характеристик

**СОДЕРЖАНИЕ ПРОЕКТА**

**Анализ проблемы исследования**

Анализ зависимости цены ноутбука от его характеристик является важным направлением исследования потребительского рынка электроники. Понимание факторов, влияющих на ценообразование ноутбуков, позволяет как потребителям принимать более информированные решения о покупке, так и производителям оптимизировать свои стратегии ценообразования и разработки продуктов.

Индекс цены ноутбука, формирующийся под влиянием множества технических и экономических факторов, является ключевым параметром отображения сложившихся рыночных тенденций и предпочтений потребителей. Он характеризует фактическую ценность, которую покупатели готовы платить за определенные комбинации характеристик, зачастую не поддающиеся простой количественной оценке. Поэтому его изучение требует комплексного анализа.

Так, например, типичный современный ноутбук может обладать определенным объемом оперативной памяти (Ram), весом, наличием сенсорного экрана (Touchscreen), типом матрицы дисплея (Ips), разрешением экрана (ppi), процессором определенного бренда (Cpu brand), типом и объемом накопителя (HDD, SSD, Hybrid, Flash\_Storage), графическим процессором (Gpu brand) и операционной системой (os). Изменение любого из этих параметров может оказать существенное влияние на конечную стоимость устройства.

Рассматривая динамику рынка ноутбуков за последние годы, необходимо отметить ключевую тенденцию роста производительности и разнообразия конфигураций, а также влияние инновационных технологий на ценообразование. Появление новых процессоров, более быстрых типов памяти и накопителей, улучшенных дисплеев и графических карт непосредственно сказывается на стоимости конечного продукта.

Анализируя эволюцию рынка ноутбуков с точки зрения ценообразования, стоит указать на ряд факторов, оказавших значительное влияние на формирование цен:

− Развитие технологий производства и снижение стоимости компонентов.

− Изменение потребительских предпочтений и спроса на определенные характеристики (например, популярность SSD-накопителей).

− Конкуренция между производителями и стратегии ценообразования.

− Влияние экономической ситуации и валютных курсов.

Общее влияние данных факторов формирует сложную динамику ценообразования, которую необходимо учитывать при анализе.

Современный рынок ноутбуков характеризуется широким спектром моделей с различными характеристиками и ценовыми категориями. Более корректным представляется анализ текущего состояния рынка, опираясь на актуальные данные о технических характеристиках и ценах.

За определенный период времени, например, на основе предоставленных данных, можно выявить закономерности изменения цены в зависимости от конкретных параметров ноутбуков. При этом в динамике ценообразования можно выделить различные сегменты рынка и влияние отдельных характеристик на стоимость.

Формирование стоимости ноутбука – это сложный процесс, зависящий от совокупности его технических характеристик, бренда, маркетинговой стратегии и текущей экономической ситуации.

Анализ данной ситуации и определяет актуальность темы исследования. Таким образом, необходимо проанализировать данные о характеристиках ноутбуков и их ценах, чтобы определить, возможно ли описать зависимость цены ноутбука от его технических параметров с помощью факторных переменных.

**Цель:** выполнить анализ данных о характеристиках ноутбуков и построить модель зависимости цены ноутбука от различных факторов с помощью линейной регрессии, позволяющую делать прогнозы с высокой точностью.

**Задачи:**

* Выполнить анализ проблемы, обосновать ее актуальность.
* Осуществить загрузку данных и подготовку их к анализу количественными методами, включая устранение пропущенных значений.
* Выполнить предварительный анализ данных, в том числе выявление и обработку выбросов, проверку распределения данных на нормальность, корреляционный анализ.
* Осуществить моделирование зависимости целевого признака (цена ноутбука) от факторных (характеристики ноутбука) методом линейной регрессии, в том числе подобрать наилучшую модель, оценить ее качество и выполнить прогнозирование.
* Выполнить интерпретацию полученных результатов и сделать выводы о достижении цели.

**Исходные данные**

В настоящей работе анализируется набор данных, содержащий информацию о различных моделях ноутбуков. Данные представляют собой характеристики ноутбуков и их цены на момент сбора.

**Список колонок анализируемого набора данных:**

1. index – индекс записи.
2. Company – компания-производитель ноутбука.
3. TypeName – тип ноутбука (например, Ultrabook, Notebook).
4. Ram – объем оперативной памяти (в гигабайтах).
5. Weight – вес ноутбука (в килограммах).
6. Price – цена ноутбука (в условных единицах) (целевой показатель).
7. Touchscreen – наличие сенсорного экрана (1 – есть, 0 – нет).
8. Ips – наличие IPS-матрицы (1 – есть, 0 – нет).
9. ppi – плотность пикселей экрана.
10. Cpu brand – бренд процессора.
11. HDD – объем жесткого диска (в гигабайтах).
12. SSD – объем твердотельного накопителя (в гигабайтах).
13. Hybrid – наличие гибридного накопителя (в гигабайтах или 0-нет).
14. Flash\_Storage – объем flash-памяти (в гигабайтах).
15. Gpu brand – бренд графического процессора.
16. os – операционная система.

Необходимо проанализировать данные о характеристиках ноутбуков и определить, возможно ли описать зависимость цены ноутбука от имеющихся факторных переменных методом линейной регрессии.

Выдвинем гипотезу исследования: цена ноутбука зависит от его характеристик, таких как объем оперативной памяти, наличие сенсорного экрана и IPS-матрицы, бренд процессора и видеокарты, а также типы и объемы накопителей, и может быть предсказана с помощью этих показателей.

**Реализация проекта**

**Этап 1. Подготовка данных к анализу**

Загрузим данные через :

import pandas as pd

import warnings

from sklearn.preprocessing import MinMaxScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

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

import matplotlib.pyplot as plt

import seaborn as sns

warnings.filterwarnings("ignore")

# Чтение файла CSV

data = pd.read\_csv('laptop-price.csv')

# Вывод первых 10 строк

print("Первые 10 строк:")

print(data.head(10))

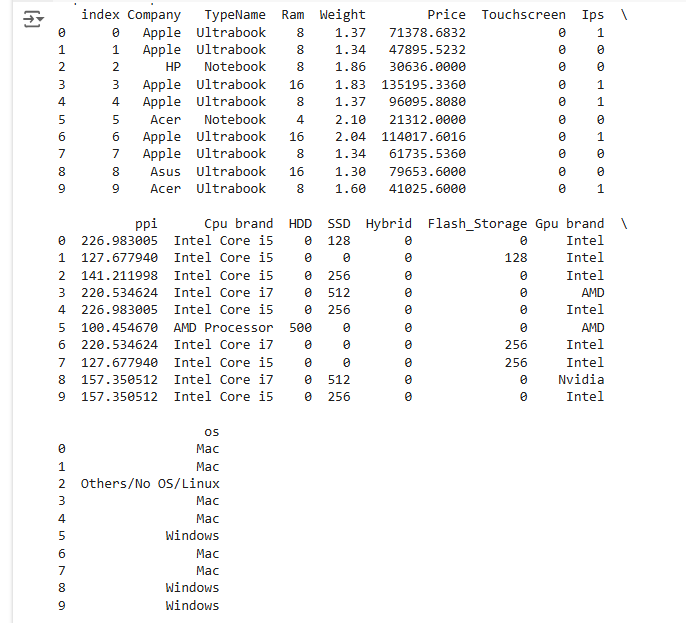


Рисунок 1. Исходный датафрейм

Убедимся, что все количественные столбцы имеют числовой тип. Если это не так, выполним преобразование типа столбца к числовому.

df.info()

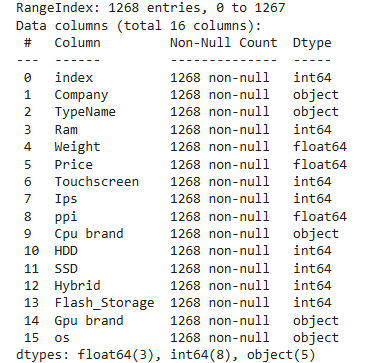
ы

Рисунок 2. Типы данных колонок

Показывает, что большинство количественных столбцов ('index', 'Ram', 'Weight', 'Price', 'Touchscreen', 'Ips', 'ppi', 'HDD', 'SSD', 'Hybrid', 'Flash\_Storage') уже имеют корректный числовой тип (int64 или float64). Но, как и ожидалось, столбцы 'Company', 'TypeName', 'Cpu brand', 'Gpu brand' и 'os' имеют тип 'object'.

Заменим категориальные данные на числовые. Мы использовали метод fit\_transform для кодирования категориальных переменных.

from sklearn.preprocessing import LabelEncoder

# Кодирование категориальных переменных

categorical\_features = ['Company', 'TypeName', 'Cpu brand', 'Gpu brand', 'os']

for feature in categorical\_features:

    le = LabelEncoder()

    data[feature] = le.fit\_transform(data[feature])

numeric\_features = ['Ram', 'Weight', 'ppi', 'HDD', 'SSD', 'Hybrid', 'Flash\_Storage', 'Price']

categorical\_features = ['Company', 'TypeName', 'Cpu brand', 'Gpu brand', 'os' , 'Touchscreen', 'Ips']

data.info()

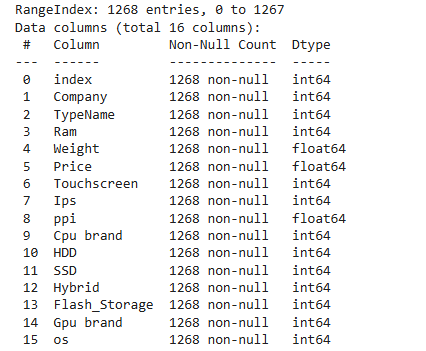


Рисунок 5. Новые типы данных колонок

Проверим данные на пропуски. Если в столбцах или строках очень много пропусков, такие признаки и объекты для анализа малоинформативны, их следует удалить.

data.isna()

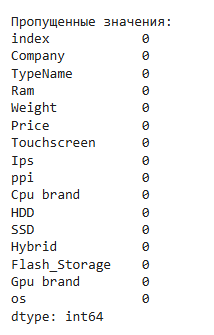


Рисунок 6. Количество пропусков по колонкам

Как видно, пропусков в колонках нет. Теперь посмотрим на количество пропусков в строках – проще всего это сделать таким же способом, каким смотрели пропуски в колонках. Для этого потребуется транспонировать датафрейм:

data.transpose().isna().sum()



Рисунок 7. Доля пропусков по строкам

Итак, результат первого этапа – это готовый к анализу набор данных в виде датафрейма.

**Этап 2. Предварительный анализ данных**

Вычислим описательные статистики по колонкам (среднее, моду, медиану, стандартное отклонение, квартили).

***Среднее арифметическое*** *равно сумме значений всех вариант выборки, деленной на объем выборки:*

.

Здесь *п* − объем выборки, а *xi* − варианты выборки.

***Модой*** называется значение признака, встречающееся в выборке наиболее часто. Условимся использовать для обозначения моды символы *Mo*.

В случае не сгруппированных данных для нахождения медианы необходимо ранжировать выборку, т. е. расположить данные в порядке их возрастания или убывания. Медианой будет являться значение признака, находящееся в середине ранжированного ряда. Медиана находится по формуле

Выборочная дисперсия находится по формуле *.*

Используется также другая формула для вычисления дисперсии: , где *.*

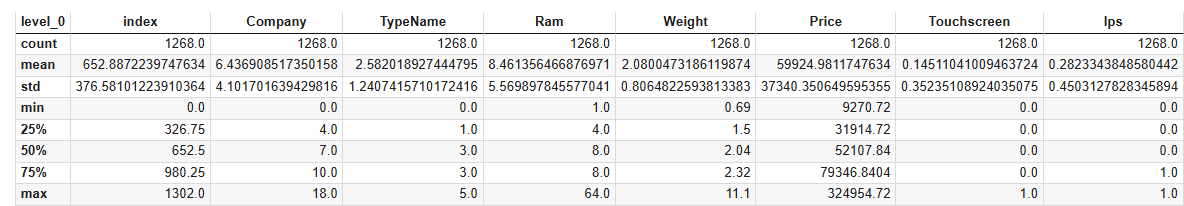
Дисперсия имеет размерность квадрата размерности случайной величины, что затрудняет ее интерпретацию и делает не очень наглядной. Для более наглядного описания рассеяния удобнее пользоваться характеристикой, размерность которой совпадает с размерностью исследуемого признака. С этой целью вводится понятие ***стандартного отклонения*** (или ***среднего квадратичного отклонения***).

***Стандартным отклонением*** называется положительный квадратный корень из дисперсии:

.

Стандартное отклонение имеет те же единицы измерения, что и результаты измерения исследуемого признака, и, таким образом, оно характеризует степень отклонения признака от среднего арифметического. Иными словами, оно показывает, как расположена основная часть вариант относительно среднего арифметического.

df.describe()



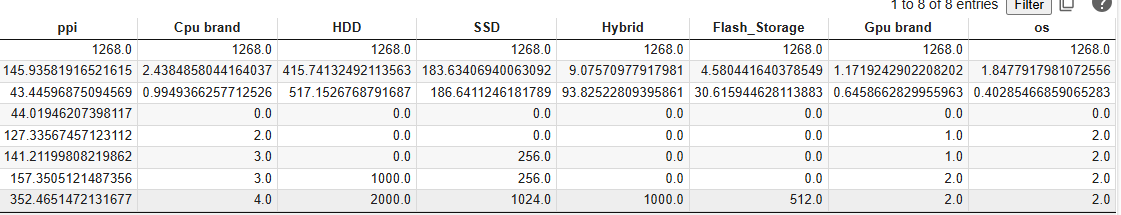


Рисунок 10. Описательные статистики по колонкам

Для наглядности попробуем изменить инженерный формат представления чисел. Функция ниже принимает на вход вещественное число и преобразует его в строку вида «целая часть.2 разряда дробной части»:

def beautify\_float(value):

return "{:.2f}".format(value)

Применим эту фукцию ко всем ячейкам датафрейма с описательными статистиками с помощью метода *applymap*():

df.describe().applymap(beautify\_float)

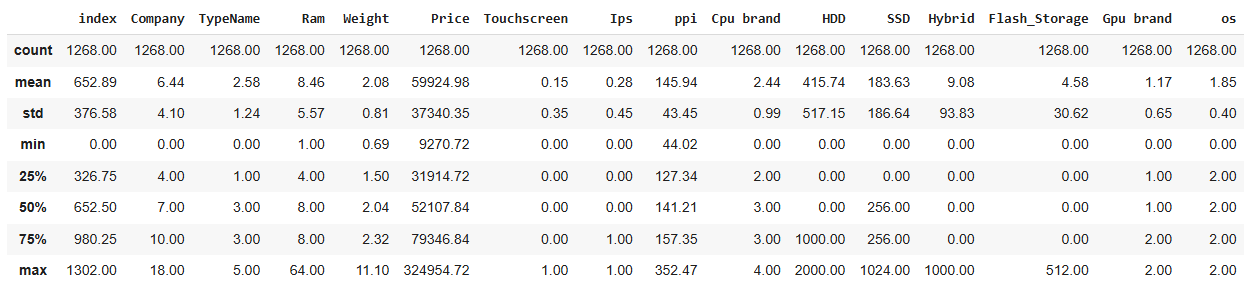


Рисунок 11. Описательные статистики по колонкам (в удобном формате)

Как мы можем заметить, максимальное значение колонки index не соответствует количеству строк. Это значит, что в колонке есть пропуски. Удалим ее, так как она нам мало чем может быть полезна.

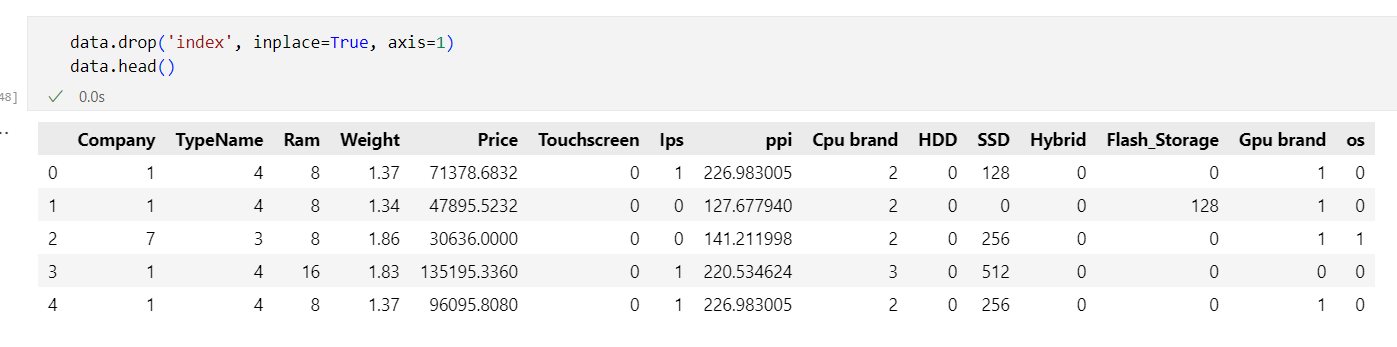


Рисунок 12. Датафрейм после удаления столбца

Для обеспечения корректной работы модели линейной регрессии и получения адекватных результатов, необходимо выполнить стандартизацию данных. Так как видно, что признаки в нашем наборе данных имеют значительный разброс в своих значениях.

from sklearn.preprocessing import StandardScaler

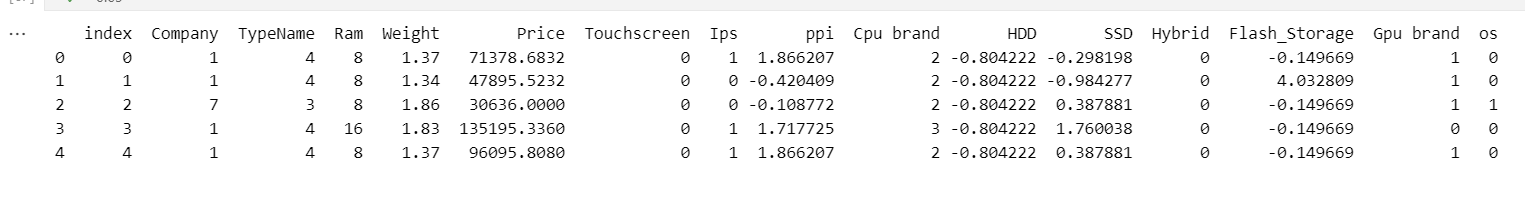
# Создаем экземпляр StandardScaler

scaler = StandardScaler()

# Стандартизируем данные

data[['ppi', 'HDD', 'SSD', 'Flash\_Storage']] = scaler.fit\_transform(data[['ppi', 'HDD', 'SSD', 'Flash\_Storage']])

print(data.head())



***Аномальными наблюдениями*** (*выбросами*, англ. *Outliers, Extreme values*) называют такие значения уровня временного ряда, которые значительно отличаются от остальных. При выявлении подобных «выбросов» возникают серьезные вопросы: являются ли отклоняющиеся данные действительно ошибками (например, регистрации) или это реальные значения и как получить адекватные оценки для параметров изучаемой совокупности.

Проверим данные на наличие выбросов, для этого можно использовать диаграмму «ящик с усами» (boxplot). Если выбросов мало, то следует их сгладить.

График ***«ящик с усами»,*** или ***«ящичковая диаграмма»***, или ***диаграмма размаха*** − график, используемый в описательной статистике и компактно изображающий одномерное [распределение вероятностей](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B2%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D0%B5%D0%B9). Такой вид диаграммы в удобной форме показывает медиану, нижний и верхний квартили, минимальное и максимальное значения выборки и [выбросы](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D0%B1%D1%80%D0%BE%D1%81_(%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0)).

Несколько таких ящиков можно нарисовать рядом друг с другом, чтобы визуально сравнивать одно распределение с другим, их можно рисовать горизонтально либо вертикально. Расстояния между различными частями ящика позволяют определить степень распространения (дисперсии) и асимметрии в данных и выявить выбросы.

Границами ящика служат первый и третий [квартили](https://ru.wikipedia.org/wiki/%D0%9A%D0%B2%D0%B0%D1%80%D1%82%D0%B8%D0%BB%D1%8C) (25-й и 75-й [процентили](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%BD%D1%82%D0%B8%D0%BB%D1%8C) соответственно), линия в середине ящика — медиана (50-й процентиль). Концы усов — края статистически значимой выборки (без [выбросов](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D0%B1%D1%80%D0%BE%D1%81_(%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0))) могут определяться несколькими способами. В общем виде эта формула имеет вид

*.*

*X*н — нижняя граница уса, *X*в — верхняя граница уса, *Q*1 — первый квартиль ,*Q*3 — третий квартиль, *k* — коэффициент, наиболее часто употребляемое значение которого равно 1,5. Данные, выходящие за границы усов ([выбросы](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D0%B1%D1%80%D0%BE%D1%81_(%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0))), отображаются на графике в виде точек, маленьких кружков или звёздочек. Иногда на графике отмечают среднее арифметическое и его [доверительный интервал](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%B2%D0%B5%D1%80%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D0%B2%D0%B0%D0%BB_%D0%B4%D0%BB%D1%8F_%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B6%D0%B8%D0%B4%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B9_%D0%B2%D1%8B%D0%B1%D0%BE%D1%80%D0%BA%D0%B8) («зарубка» на ящике). На рис. изображен график «ящик с усами».

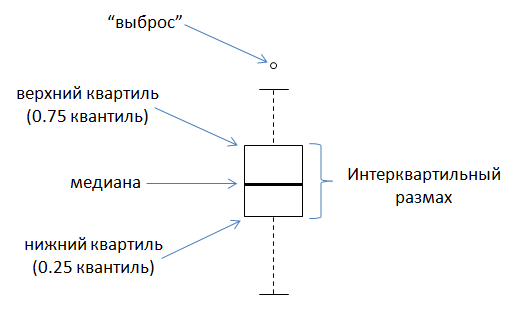


Рисунок 12. «Ящик с усами»

Посмотрим на диаграммы boxplot всех числовых колонок датафрейма:

df.boxplot();

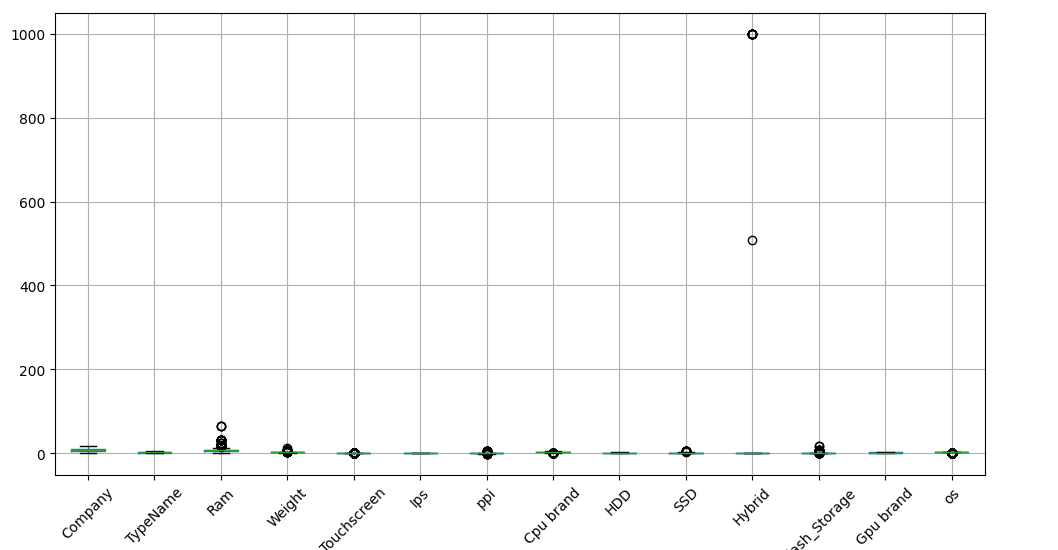


Рисунок 14. Диаграммы boxplot всех числовых колонок

Заметны явные выбросы в колонке Hybrid. Посмотрим на значения этой колонки, отсортировав строки от большего значения к меньшему

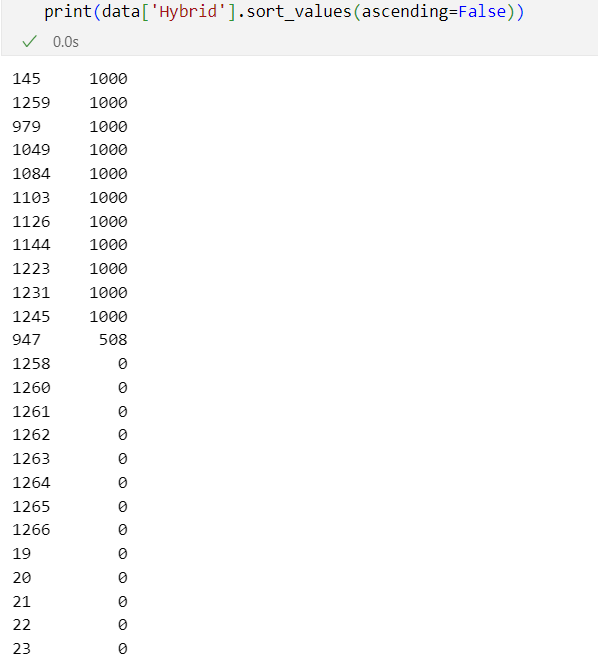


Рисунок 15. Значения колонки Hybrid по убыванию

Мы видим, что абсолютное большинство значений колонки, а именно 99%, составляют нули. Тогда имеет смысл удалить колонку.

Снова взглянем на boxplot всех числовых колонок датафрейма

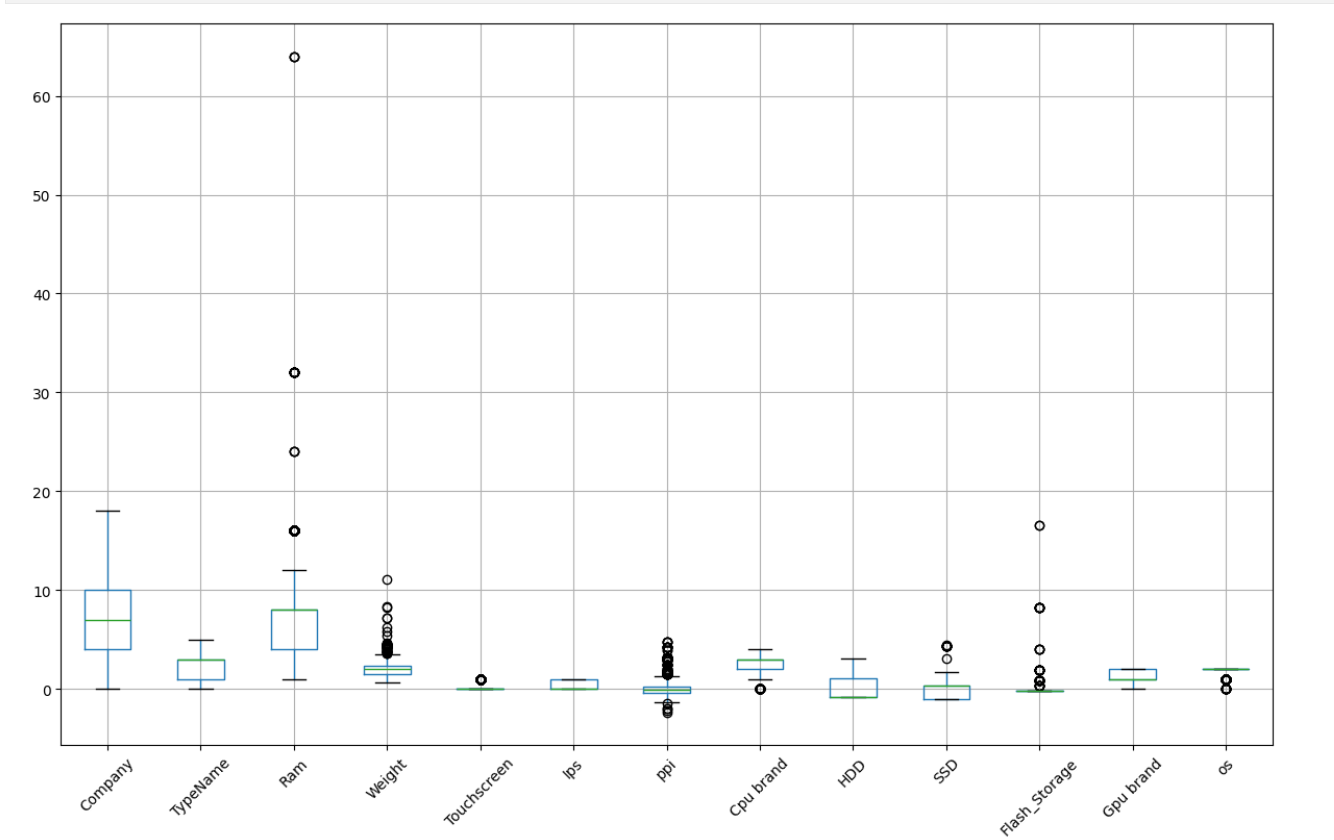
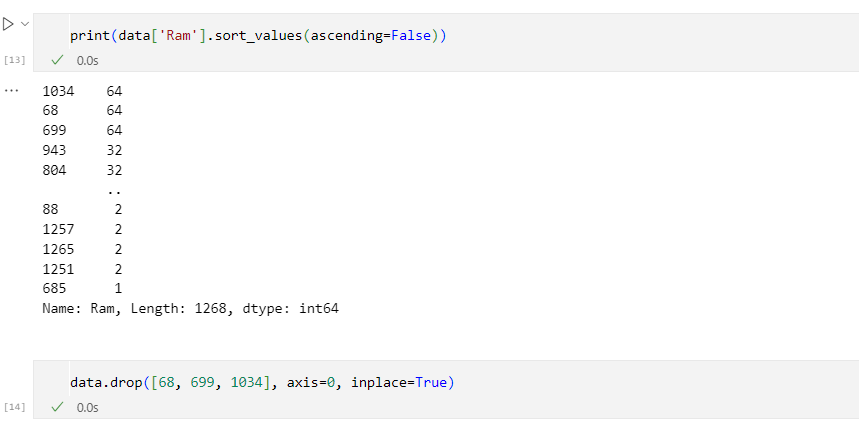


Рисунок 16. Диаграммы boxplot всех числовых колонок

Удалим еще отдельные строки с выбросами в колонке Ram



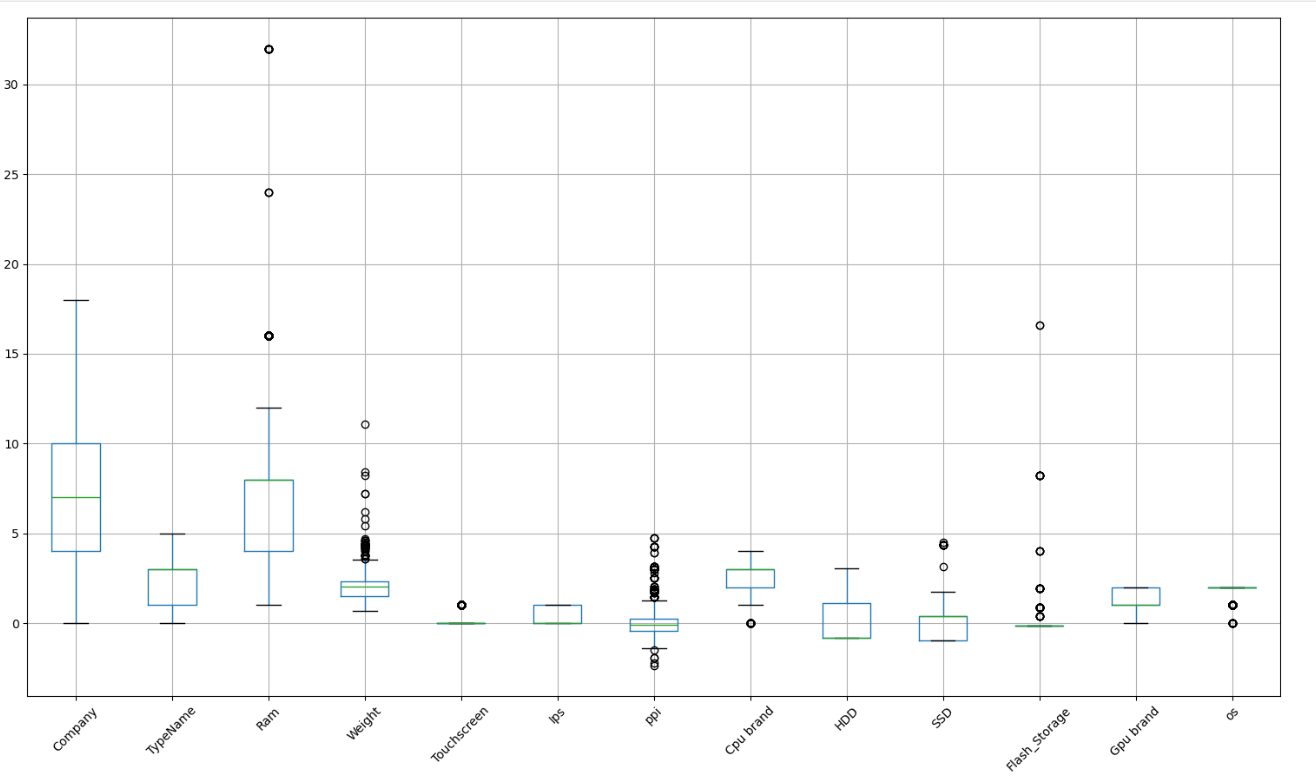


Рисунок 17. Диаграммы boxplot всех числовых колонок

Построим отдельные диаграммы для каждой колонки:

plt.figure(figsize=(12, 6))

for i, col in enumerate(numeric\_features):

    plt.subplot(2, 5, i + 1)

    sns.boxplot(x=data[col])

    plt.title(col)

plt.tight\_layout()

plt.show()

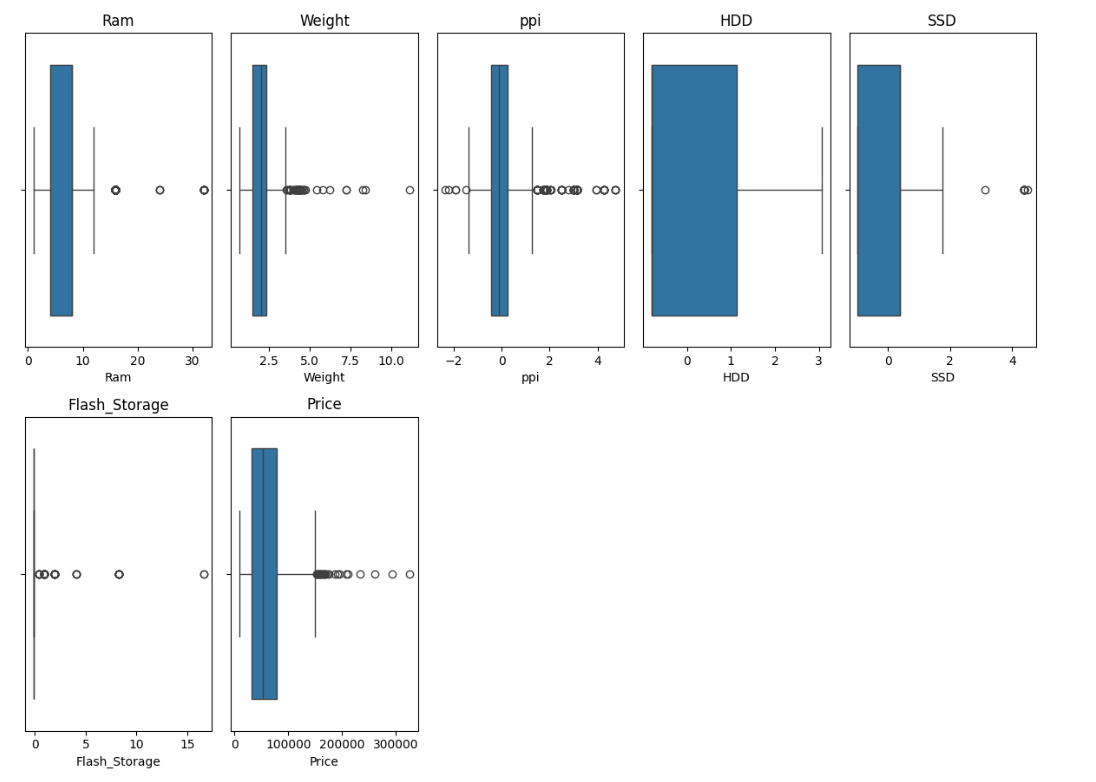
­ 

Рисунок 17. Визуализация выбросов на отдельных диаграммах boxplot

Как видим, необходимо обработать выбросы в колонках Weight и ppi

data.Weight.replace(data.Weight.max(), data.Weight.quantile(0.99), inplace=True)

data.ppi.replace(data.ppi.max(), data.ppi.quantile(0.99), inplace=True)

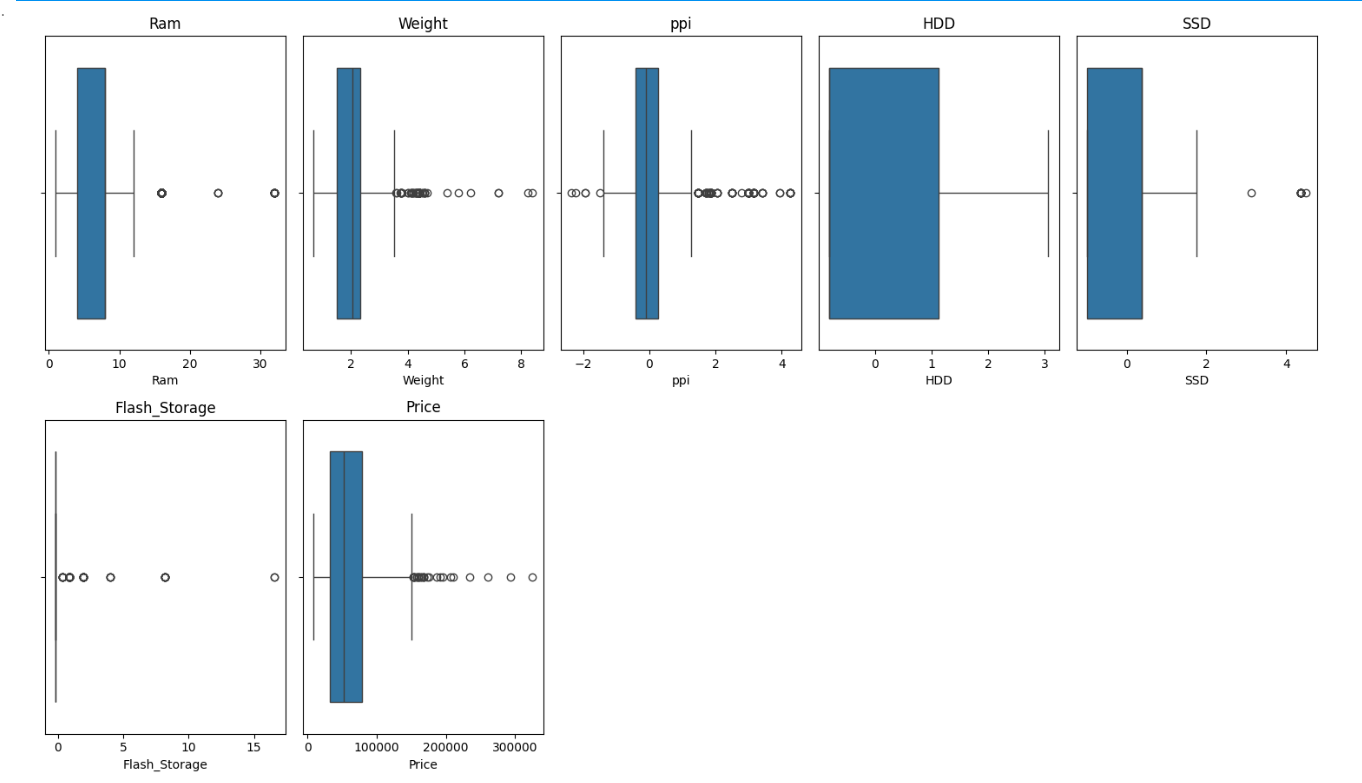


Рисунок 18. Визуализация выбросов на отдельных диаграммах boxplot после обработки

Проверим данные на нормальность распределения двумя способами:

* 1. Построим гистограмму и сделать предположение о том, являются ли данные нормально распределенными.
  2. Выполним статистический тест на нормальность и убедимся, что выдвинутое ранее предположение о нормальности верно или ошибочно.

***Гистограмма****,* представляющая собой совокупность примыкающих друг к другу прямоугольников, основание каждого из которых равно ширине интервала группировки, а площадь − частости этого интервала.

Гистограмма строится в декартовой (прямоугольной) системе координат следующим образом. По оси абсцисс откладываются отрезки, отображающие интервалы группировки, а затем на каждом из них строится прямоугольник, площадь которого равна частости данного интервала. В случае если все интервалы группировки имеют одинаковую ширину, высоты прямоугольников пропорциональны соответствующим частостям.

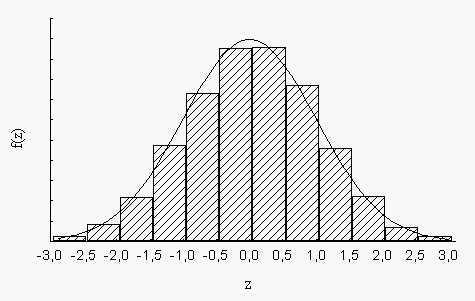


Рисунок 19. Гистограмма

Сначала построим гистограммы для всех числовых колонок:

plt.figure(figsize=(15, 10))

i = 1

for col in data.columns[:-1]:

  plt.subplot(4, 4, i)

  plt.hist(data[col], bins=30, color="blue", linewidth=1, edgecolor="black")

  plt.title(col)

  i += 1

plt.tight\_layout()

plt.show()

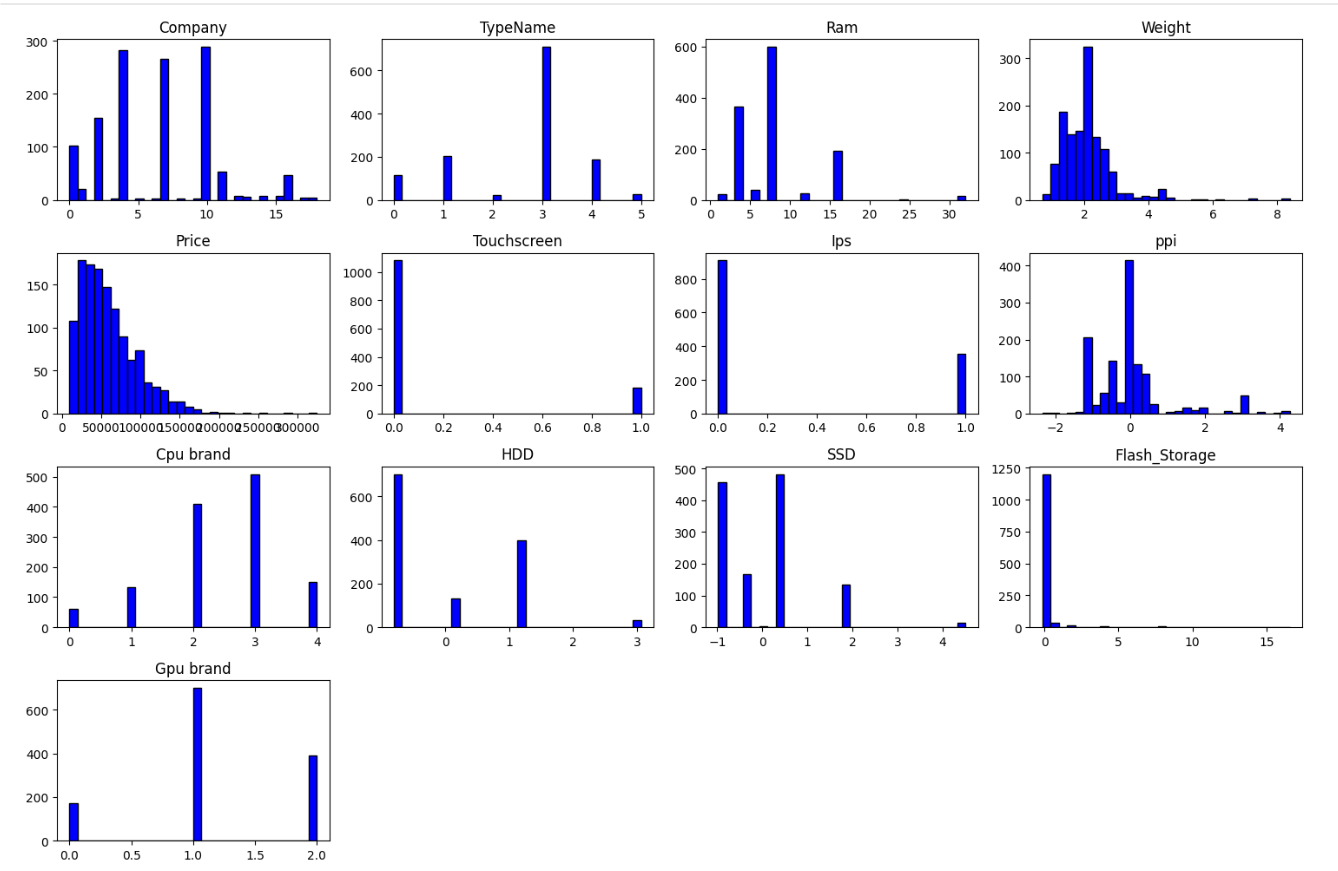


Рисунок 20. Гистограммы для всех колонок

Визуально не видно, что тут есть нормально распределенные колонки, поэтому применим статистические тесты (критерии согласия).

***Критерии согласия з***аключаются в проверке предположения о том, что результаты наблюдений могут быть описаны с помощью определенного закона распределения (в нашем случае нормального распределения). **П**ри малом числе измерений часто используется критерий Шапиро−Уилка.

На формальном языке проверяется гипотеза:

H0: наши данные согласуются с нормальным распределением.

Если p-value меньше заданного уровня значимости (обычно 0,05 или 0,01), то основная гипотезу отвергается.

Используем тест Шапиро-Уилка:

import scipy.stats as stats

for col in data.columns[:-1]:

  print(col, stats.shapiro(data[col]))

Результаты теста для колонок:

Company ShapiroResult(statistic=0.9358294018458737, pvalue=6.039747097644461e-23)

TypeName ShapiroResult(statistic=0.8137337368840596, pvalue=6.398477055304799e-36)

Ram ShapiroResult(statistic=0.6636466716181386, pvalue=1.9517495308394863e-44)

Weight ShapiroResult(statistic=0.798139713833142, pvalue=5.046510959624227e-37)

Price ShapiroResult(statistic=0.8927076336046738, pvalue=8.023163742358289e-29)

Touchscreen ShapiroResult(statistic=0.4190978608640723, pvalue=3.7811385307990555e-53)

Ips ShapiroResult(statistic=0.5637002928699115, pvalue=1.74829616863615e-48)

ppi ShapiroResult(statistic=0.784916913636655, pvalue=6.609226818436786e-38)

Cpu brand ShapiroResult(statistic=0.8863794650518356, pvalue=1.603589091829425e-29)

HDD ShapiroResult(statistic=0.7216855459983138, pvalue=1.3254414172299102e-41)

SSD ShapiroResult(statistic=0.7992402102601479, pvalue=6.005482125238716e-37)

Hybrid ShapiroResult(statistic=0.06824798623348849, pvalue=2.4721661598901404e-61)

Flash\_Storage ShapiroResult(statistic=0.1340142078780907, pvalue=5.018357165763597e-60)

Gpu brand ShapiroResult(statistic=0.7846680488049865, pvalue=6.367360986311375e-38)

Как видим, p-value имеет очень маленькие значения, что указывает на то, что они не следуют нормальному распределению.

**Этап 3. Корреляционный анализ данных**

***Корреляционный анализ*** – это совокупность методов оценивания степени тесноты статистической связи между анализируемыми переменными.

Выполним корреляционный анализ данных с помощью матрицы корреляции. В случае нормальности всех данных следует использовать коэффициент корреляции Пирсона, в противном случае – ранговые коэффициенты корреляции Спирмена.

**Парный коэффициент** корреляции характеризует взаимосвязь двух переменных на фоне действия остальных показателей и является самым распространенным показателем тесноты связи при статистическом анализе данных.

Парный коэффициент корреляции между количественными случайными переменными и носит название *выборочного коэффициента корреляции* Пирсона (*sample correlation coefficient*) (или просто коэффициента корреляции) и находится по формуле

где и  — *выборочные дисперсии (sample variances*) переменных  и , а — *выборочная ковариация* или выборочный ковариационный момент, и соответствующие *средние (means)* определяются по формулам



Коэффициент корреляции обладает следующими свойствами:

1. Принимает значения от –1 до +1.
2. Если , то связь между переменными  и  считается сильной. Если , то связь слабая.
3. Если , то корреляционное поле наблюдений представляет собой совокупность точек, которые можно расположить на одной прямой. Знак «+» свидетельствует о прямой линейной зависимости между переменными и , а знак «—» − об обратной линейной зависимости.
4. При  линейная корреляционная связь отсутствует.

Метод ранговой корреляции Спирмена позволяет определить тесноту (силу) и направление корреляционной связи между двумя признаками (как количественными, так и качественными). Коэффициент ранговой корреляции имеет границы изменения от –1 до +1. Полное совпадение рангов означает максимально тесную прямую связь, полная противоположность рангов – максимально тесную обратную связь. Формула расчета ***коэффициента корреляции рангов Ч. Спирмена:***

где  – ранг  в выборке .

Матрицу корреляции отобразим с помощью диаграммы «тепловая карта» (heatmap).

import seaborn as sns

import matplotlib.pyplot as plt

# Вычисление матрицы корреляции

correlation\_matrix = data.corr()

# Создание heatmap

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

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm')

plt.title('Матрица корреляции')

plt.show()

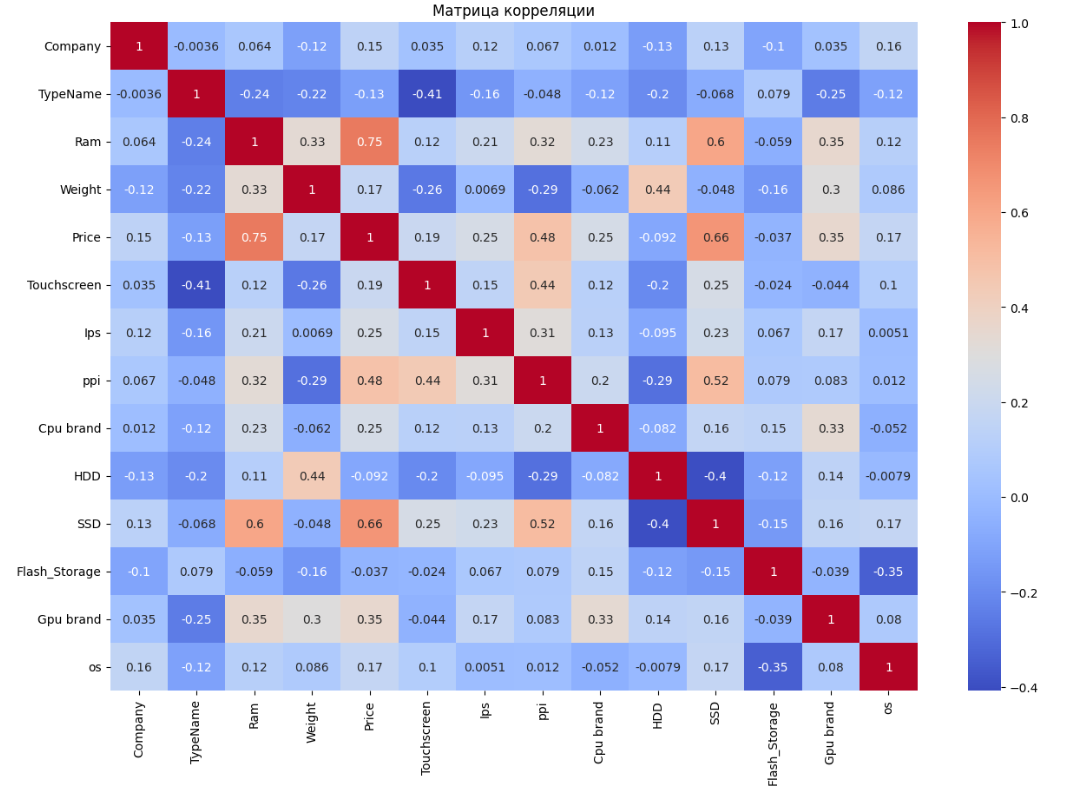


Рисунок 21. Тепловая карта матрицы корреляции

Целевой переменной в данном анализе выступает цена ноутбука (Price). В наборе данных присутствует множество факторов, которые могут потенциально влиять на стоимость устройства. На основе результатов корреляционного анализа будем отбирать факторные признаки, которые предположительно оказывают наибольшее влияние на целевую переменную, а также как можно меньше зависят друг от друга.

Из матрицы корреляции видно, что наиболее сильную связь с ценой имеют следующие признаки:

* Объем оперативной памяти (Ram): коэффициент корреляции 0.75 (сильная положительная связь);
* Объем SSD-накопителя (SSD): коэффициент корреляции 0.66 (сильная положительная связь);
* Число пикселей на дюйм (ppi): коэффициент корреляции 0.48 (умеренная положительная связь);
* Бренд видеокарты (Gpu brand): коэффициент корреляции 0.35 (умеренная положительная связь);
* Наличие IPS-матрицы (Ips): коэффициент корреляции 0.25 (слабая положительная связь);
* Вес ноутбука (Weight): коэффициент корреляции 0.17 (слабая положительная связь).

Для моделирования выберем следующие признаки, которые показывают высокую корреляцию с ценой и при этом относительно независимы друг от друга:

* Ram (объем оперативной памяти);
* SSD (объем SSD-накопителя);
* ppi (плотность пикселей);
* Gpu brand (бренд видеокарты);
* Ips(наличие IPS-матрицы)
* Cpu brand(бренд процессора)

Таким образом, факторные признаки для построения модели определяются следующим образом:

X\_cols = ["Ram", "SSD", "ppi", "Gpu brand", 'Ips', 'Cpu brand']

Целевой переменной остается:

y\_col = "Price"

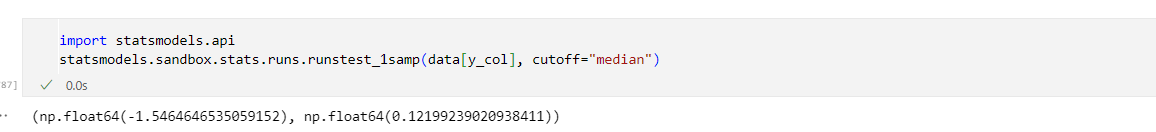
Проверим целевой признак на упорядоченность (если данные упорядочены, следует их перемешать), используя статистический тест Вальда-Волфовитца. Подключим соответствующий модуль:

import statsmodels.api

Нулевой гипотезой в тесте Вальда-Волфовитца является «H0: выборка случайна».

statsmodels.sandbox.stats.runs.runstest\_1samp(df[y\_col], cutoff="median")

Результат:



Отрицательное значение (-1.546) указывает, что количество серий меньше, чем ожидалось бы при случайном распределении. Это может означать, что значения склонны группироваться.

p-value(0.122) — это вероятность того, что наблюдаемое отклонение произошло случайно, если нулевая гипотеза (случайное распределение) верна. Так как 0.122 > 0.05, то мы не можем отвергнуть нулевую гипотезу.

Таким образом, лучше перемешать данные, чтобы улучшить процесс обучения модели.

Таким образом, на втором этапе была выдвинута гипотеза о том, что цена ноутбука зависит от объема оперативной памяти, объема SSD-накопителя, плотности пикселей дисплея (ppi), бренда видеокарты и процессора, а также наличия IPS-матрицы, и может быть смоделирована с их помощью с применением модели линейной регрессии.

**Этап 4. Моделирование**

Разобьём данные на обучающую и тестовую выборки, перемешав данные, используя соотношение 8 к 2:

# Разделение данных на обучающую и тестовую выборки

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

На обучающей выборке построим модель множественной линейной регрессии, используя в качестве целевой и факторных переменных отобранные ранее столбцы.

Подключим функцию для построения модели линейной регрессии:

from sklearn.linear\_model import LinearRegression

Обучим модель на обучающей выборке:

from sklearn.linear\_model import LinearRegression

# Обучение модели

model = LinearRegression().fit(X\_train, y\_train)

Подключим модуль с оценками качества моделей:

import sklearn.metrics as metrics

*определяет долю дисперсии зависимой переменной, объясненную эмпирической функцией регрессии.* По этой причине коэффициент детерминации является основной мерой качества подгонки регрессионной модели к данным с помощью выборочного уравнения регрессии.

Вычислим коэффициент детерминации R2 модели на обучающей выборке. Убедимся, что модель значима по критерию Фишера. Если коэффициент детерминации модели низок (меньше 0,7) и (или) модель незначима, то попробуем другие комбинации факторных признаков из отобранных ранее. Выберем такой набор признаков, при котором R2 модели будет наиболее высок и модель при этом окажется значимой.

Получим модельные значения на обучающей выборке:

y\_model = model.predict(X\_test)

Вычислим R2 на обучающей выборке:

R2 = metrics.r2\_score(y\_test, y\_model)

Полученное значение R2: 0.696174451168791 Вывод: на 69,6% наша целевая переменная описана с помощью объясняющих.

Проверим гипотезу о том, что модель статистически значима (ее коэффициент детерминации R2 значимо не равен 0), для этого используем критерий Фишера (F-тест):

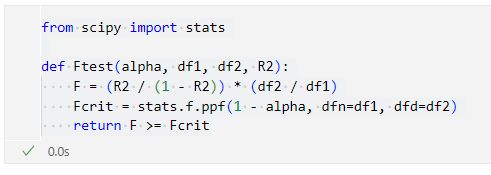
H0: модель незначима

H1: модель значима

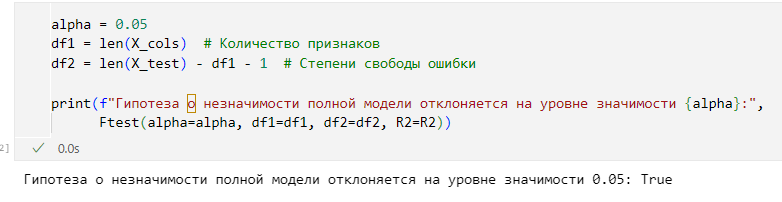
Статистика критерия (F-статистика) вычисляется по формуле

где df1 – число коэффициентов в модели за вычетом 1, df2 – число наблюдений (объем выборки) за вычетом числа коэффициентов модели.

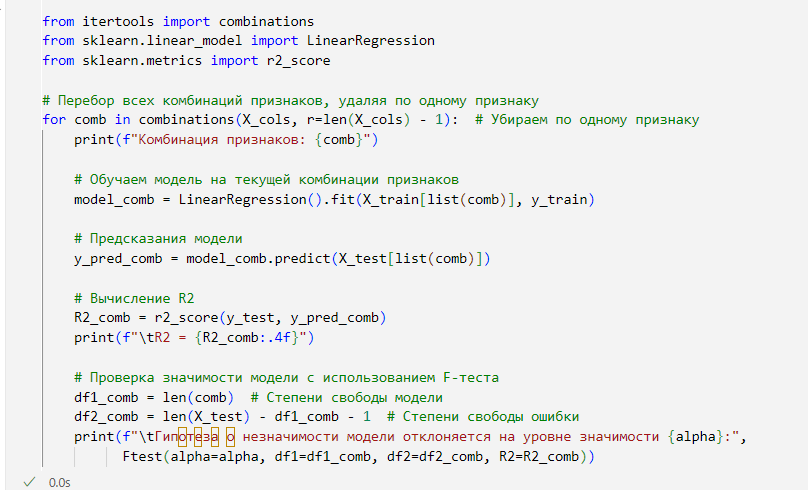
Критическое значение критерия Fкрит определяется как квантиль распределения Фишера уровня с числом степеней свободы df1, df2. Если F ≥ Fкрит, значит на уровне значимости можно отклонить гипотезу H0. Напишем функцию, выполняющую F-тест (возвращает True, если гипотеза о незначимости модели отклоняется, иначе False).



Выполним тест, примем уровень значимости 5%:



Итак, коэффициент детерминации достаточновысокий, модель значима по критерию Фишера. Переберем несколько комбинаций факторных признаков в поисках такой, при которой R2 будет еще выше.



Комбинация признаков: ('Ram', 'SSD', 'ppi', 'Gpu brand', 'Ips')

R2 = 0.6964

Гипотеза о незначимости модели отклоняется на уровне значимости 0.05: True

Комбинация признаков: ('Ram', 'SSD', 'ppi', 'Gpu brand', 'Cpu brand')

R2 = 0.6955

Гипотеза о незначимости модели отклоняется на уровне значимости 0.05: True

Комбинация признаков: ('Ram', 'SSD', 'ppi', 'Ips', 'Cpu brand')

R2 = 0.6834

Гипотеза о незначимости модели отклоняется на уровне значимости 0.05: True

Комбинация признаков: ('Ram', 'SSD', 'Gpu brand', 'Ips', 'Cpu brand')

R2 = 0.6664

Гипотеза о незначимости модели отклоняется на уровне значимости 0.05: True

Комбинация признаков: ('Ram', 'ppi', 'Gpu brand', 'Ips', 'Cpu brand')

R2 = 0.6760

Гипотеза о незначимости модели отклоняется на уровне значимости 0.05: True

Комбинация признаков: ('SSD', 'ppi', 'Gpu brand', 'Ips', 'Cpu brand')

R2 = 0.5459

Гипотеза о незначимости модели отклоняется на уровне значимости 0.05: True

Как можно заметить, уменьшение числа факторов только снижает качество модели.

Дадим интерпретацию коэффициентам модели.

print(f"Коэффициенты модели: {model.coef\_}, свободный член: {model.intercept\_}")

Результат:

Коэффициенты модели: [ 3591.24604099 10159.21600974 5498.49531587 6093.30815858

1455.75171359 742.91600437], свободный член: 20152.512287827987

Полученные коэффициенты модели и свободный член можно интерпретировать следующим образом:

* Свободный член (20152.51) - при нулевых значениях всех факторных переменных ('Ram', 'SSD', 'ppi', 'Gpu brand', 'Ips', 'Cpu brand') цена ноутбука будет равна примерно 20152.5.

Коэффициент при 'Ram' (3591.25) - при увеличении объема оперативной памяти ('Ram') на одну единицу, цена ноутбука увеличивается на примерно 3591.25, при условии, что все остальные факторные переменные остаются неизменными.

* Коэффициент при 'SSD' (10159.22) - при увеличении объема твердотельного накопителя ('SSD') на одну единицу, цена ноутбука увеличивается на примерно 10159.22, при условии, что все остальные факторные переменные остаются неизменными.
* Коэффициент при 'ppi' (5498.50) - при увеличении плотности пикселей на дюйм ('ppi') на одну единицу, цена ноутбука увеличивается на примерно 5498.50, при условии, что все остальные факторные переменные остаются неизменными.
* Коэффициент при 'Gpu brand' (6093.31) - при изменении бренда графического процессора ('Gpu brand') на одну категорию, цена ноутбука увеличивается на примерно 6093.31, при условии, что все остальные факторные переменные остаются неизменными.
* Коэффициент при 'Ips' (1455.75) - при наличии или отсутствии IPS-дисплея ('Ips'), цена ноутбука увеличивается или уменьшается на примерно 1455.75, при условии, что все остальные факторные переменные остаются неизменными.
* Коэффициент при 'Cpu brand' (742.92) - при изменении бренда центрального процессора ('Cpu brand') на одну категорию, цена ноутбука увеличивается на примерно 742.92, при условии, что все остальные факторные переменные остаются неизменными.

Эти коэффициенты показывают влияние соответствующих переменных на цену ноутбука. Чем выше коэффициент, тем большее влияние имеет соответствующий фактор на цену.

Построим диаграмму рассеяния, отображающую зависимость цены от объема оперативной памяти:

import matplotlib.pyplot as plt

sorted\_indices = X\_test.sort\_values(by='Ram').index

X\_test\_sorted = X\_test.loc[sorted\_indices]

y\_model\_sorted = model.predict(X\_test\_sorted)

# Построение графика

plt.figure(figsize=(10, 6))

plt.scatter(X\_test['Ram'], y\_test, color="blue", label="Реальные данные")

plt.plot(X\_test\_sorted['Ram'], y\_model\_sorted, color="red", label="Модель линейной регрессии")

plt.xlabel('Ram')

plt.ylabel('Price')

plt.title('Зависимость цены ноутбука от объема оперативной памяти')

plt.legend()

y\_pred = model.predict(X\_test)

R2 = metrics.r2\_score(y\_test, y\_pred)

print(R2)

plt.show()

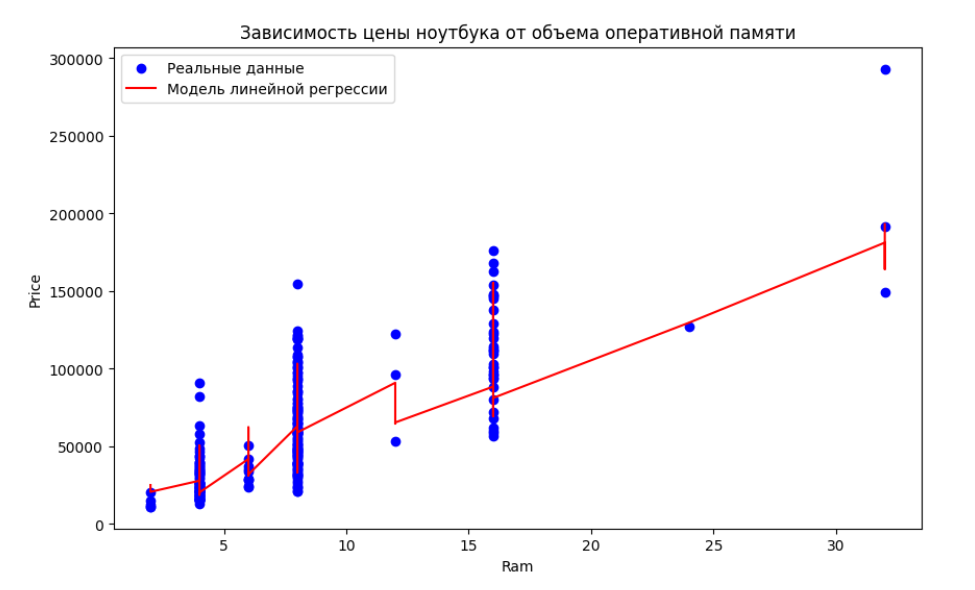


Рисунок 22. Диаграмма рассеяния с наложенной линией регрессии

Модель линейной регрессии является эффективным инструментом для изучения линейной связи между целевой переменной и факторными. Однако, в реальных данных часто встречаются более сложные зависимости, которые нельзя адекватно описать линейной моделью. В таких случаях модель линейной регрессии может не обеспечивать желаемой точности прогнозирования.

Для описания более сложных зависимостей между переменными можно использовать модель полиномиальной регрессии. Эта модель основана на предположении, что существует полиномиальная связь между целевой переменной и факторными. Она позволяет нам изучить влияние отдельных факторов на целевую переменную и интерпретировать коэффициенты модели.

Модель полиномиальной регрессии имеет ряд преимуществ перед линейной моделью. Она позволяет описывать более сложные зависимости между переменными, включая нелинейные и нелокальные зависимости. Кроме того, модель полиномиальной регрессии может быть более точной, чем линейная модель, особенно в случаях, когда данные имеют полиномиальную структуру.

Однако, для модели полиномиальной регрессии необходимо выбрать оптимальную степень полинома. Этот параметр влияет на точность модели. Если выбрать слишком низкую степень полинома, модель может быть недостаточно точной, в то время как чрезмерно высокая степень полинома может привести к переобучению модели. Поэтому, выбор оптимальной степени полинома является важным шагом в построении модели полиномиальной регрессии.

Была выбрана модель полинома 3 степени, чтобы достичь компромисса между точностью и риском переобучить модель

Обучим модель на обучающей выборке:

import matplotlib.pyplot as plt

import numpy as np

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import PolynomialFeatures

# Создание модели полиномиальной регрессии

model = Pipeline([

    ('poly\_features', PolynomialFeatures(degree=3)),

    ('linear\_regression', LinearRegression())

])

# Обучение модели

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_train)

Проведем тестирование на обучающей выборке:

y\_pred = model.predict(X\_train)

r2 = r2\_score(y\_train, y\_pred)

print("Коэффициент детерминации на обучающей выборке:", r2)

Результат:

Коэффициент детерминации на обучающей выборке: 0.7544660268791125

**Этап 5. Прогнозирование**

Выполним прогнозирование целевой переменной на тестовой выборке и вычислим коэффициент детерминации R2.

y\_pred = model.predict(X\_test)

r2 = r2\_score(y\_test, y\_pred)

print("Коэффициент детерминации на тестовой выборке:", r2)

Результат:

Коэффициент детерминации на тестовой выборке: 0.696174451168791

Выполним прогнозирование целевой переменной на тестовой выборке для модели полиномиальной регрессии и вычислим коэффициент детерминации R2.

y\_pred = model.predict(X\_test)

r2 = r2\_score(y\_test, y\_pred)

print("Коэффициент детерминации на тестовой выборке:", r2)

Результат:

Коэффициент детерминации на тестовой выборке: 0.7258338636780695

**Заключение**

На основе проведенного регрессионного анализа с использованием модели полиномиальной регрессии и линейной регрессии можно сделать следующие выводы:

* Выдвинутая гипотеза о зависимости цены ноутбука от объема оперативной памяти ('Ram'), объема твердотельного накопителя ('SSD'), плотности пикселей ('ppi') и марки графического процессора ('Gpu brand') подтвердилась. Обе модели показали, что эти факторы имеют значительное влияние на цену ноутбука.
* Модель полиномиальной регрессии продемонстрировала достаточно высокую точность в прогнозировании цены ноутбуков на тестовой выборке, с коэффициентом детерминации R^2 равным 0.73
* Несмотря на то, что R^2 не является абсолютно высоким, модель все равно может быть полезной для прогнозирования и анализа цен ноутбуков, учитывая наличие нелинейных зависимостей в данных.
* Модель линейной регрессии также показала хорошую точность, но несколько ниже, чем модель полиномиальной регрессии, с коэффициентом детерминации R^2 равным 0.70
* Далее, можно рассмотреть другие методы регрессии или оптимизировать параметры модели полиномиальной регрессии для улучшения ее точности.

Построенные модели полиномиальной регрессии и линейной регрессии могут быть использованы для прогнозирования цен ноутбуков с учетом различных факторов, что может быть полезным при анализе рынка и принятии решений в сфере продаж ноутбуков.

* Таким образом, мы достигли цели, то есть провели анализ данных о ноутбуках и построили модели для прогнозирования их цен с помощью моделей полиномиальной регрессии и линейной регрессии. Решены следующие задачи:
* Выполнен анализ проблемы и обоснована ее актуальность для рынка ноутбуков.
* Осуществлена загрузка данных и их подготовка к анализу.
* Проведен предварительный анализ данных, включая выявление и обработку выбросов и проверку нормальности распределения.
* Использованы модели полиномиальной регрессии и линейной регрессии для прогнозирования цен ноутбуков и оценка их качества с помощью R^2.
* Выполнена интерпретация полученных результатов и предложены рекомендации по улучшению точности прогнозирования.

**Список использованных источников и литературы**

1. Амоа К.А. Разработка программных пакетов на языке Python: учебное пособие / К.А. Амоа, Н.А. Рындин, Ю.С. Скворцов. – Воронеж: Воронежский государственный технический университет, ЭБС АСВ, 2020. – 61 c. // Электронно-библиотечная система IPR BOOKS: [сайт].
2. Вандер П. Python для сложных задач: наука о данных и машинное обучение. — СПб.: Питер, 2018. — 576 с.
3. Васильев, А.Н. Программирование на Python в примерах и задачах / А,Н. Васильев. — Москва : Эксмо, 2021. — 616 с.
4. Волкова В.М., Программные системы статистического анализа. Обнаружение закономерностей в данных с использованием системы R и языка Python [Электронный ресурс]: учебное пособие / Волкова В.М. - Новосибирск : Изд-во НГТУ, 2017. - 74 с. - ISBN 978-5-7782-3183-2 - Режим доступа: http://www.studentlibrary.ru/book/ISBN9785778231832.html
5. Гуриков С. Р. Основы алгоритмизации и программирования на Python : учеб. пособие / С.Р. Гуриков. ? М. : ФОРУМ : ИНФРА-М, 2018. ? 343 с. ? (Высшее образование: Бакалав-риат). - Режим доступа: http://znanium.com/catalog/product/924699
6. Криволапов С.Я. Математика на Python : учебник / С.Я. Криволапов, М.Б. Хрипунова. — Москва: КНОРУС, 2022. — 456 с.
7. Маккини У. Python и анализ данных / пер. с анг. А. А. Слинкина. – М.: ДМК Пресс, 2020. – 540 с.

**Приложения**

Приложение 1

Программный код

import pandas as pd

import warnings

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

import matplotlib.pyplot as plt

import seaborn as sns

warnings.filterwarnings("ignore")

# Чтение файла CSV

data = pd.read\_csv('laptop-price.csv')

# Вывод первых 10 строк

print("Первые 10 строк:")

print(data.head(10))

data.info()

# Название столбцов

print("\nНазвание столбцов:")

print(data.columns)

# Определение числовых столбцов

numeric\_columns = data.select\_dtypes(include=['int64', 'float64']).columns

print("\nЧисловые столбцы:")

print(numeric\_columns)

from sklearn.preprocessing import LabelEncoder

# Кодирование категориальных переменных

categorical\_features = ['Company', 'TypeName', 'Cpu brand', 'Gpu brand', 'os']

for feature in categorical\_features:

    le = LabelEncoder()

    data[feature] = le.fit\_transform(data[feature])

numeric\_features = ['Ram', 'Weight', 'ppi', 'HDD', 'SSD', 'Hybrid', 'Flash\_Storage', 'Price']

categorical\_features = ['Company', 'TypeName', 'Cpu brand', 'Gpu brand', 'os' , 'Touchscreen', 'Ips']

data.info()

# Проверка пропущенных значений

missing\_values = data.isnull().sum()

print("\nПропущенные значения:")

print(missing\_values)

data.transpose().isna().sum()

data.describe()

def beautify\_float(value):

  return "{:.2f}".format(value)

data.describe().applymap(beautify\_float)

data.drop('index', inplace=True, axis=1)

data.head()

from sklearn.preprocessing import StandardScaler

# Создаем экземпляр StandardScaler

scaler = StandardScaler()

# Стандартизируем данные

data[['ppi', 'HDD', 'SSD', 'Flash\_Storage']] = scaler.fit\_transform(data[['ppi', 'HDD', 'SSD', 'Flash\_Storage']])

print(data.head())

data.drop(columns=['Price']).boxplot(rot=45, figsize=(15, 9))

plt.tight\_layout()

plt.show()

print(data['Ram'].sort\_values(ascending=False))

data.drop([68, 699, 1034], axis=0, inplace=True)

print(data['Hybrid'].sort\_values(ascending=False))

data.drop('Hybrid', axis=1, inplace=True)

data

data.boxplot(column='Price', figsize=(6, 6))

plt.show()

numeric\_features = ['Ram', 'Weight', 'ppi', 'HDD', 'SSD',  'Flash\_Storage', 'Price']

# Визуализация распределения числовых переменных

plt.figure(figsize=(12, 9))

for i, col in enumerate(numeric\_features):

    plt.subplot(2, 5, i + 1)

    sns.boxplot(x=data[col])

    plt.title(col)

plt.tight\_layout()

plt.show()

data.Weight.replace(data.Weight.max(), data.Weight.quantile(0.99), inplace=True)

data.ppi.replace(data.ppi.max(), data.ppi.quantile(0.99), inplace=True)

numeric\_features = ['Ram', 'Weight', 'ppi', 'HDD', 'SSD',  'Flash\_Storage', 'Price']

# Визуализация распределения числовых переменных

plt.figure(figsize=(15, 9))

for i, col in enumerate(numeric\_features):

    plt.subplot(2, 5, i + 1)

    sns.boxplot(x=data[col])

    plt.title(col)

plt.tight\_layout()

plt.show()

plt.figure(figsize=(15, 10))

i = 1

for col in data.columns[:-1]:

  plt.subplot(4, 4, i)

  plt.hist(data[col], bins=30, color="blue", linewidth=1, edgecolor="black")

  plt.title(col)

  i += 1

plt.tight\_layout()

plt.show()

import scipy.stats as stats

for col in data.columns[:-1]:

  print(col, stats.shapiro(data[col]))

import seaborn as sns

import matplotlib.pyplot as plt

# Вычисление матрицы корреляции

correlation\_matrix = data.corr()

print(correlation\_matrix)

# Создание heatmap

plt.figure(figsize=(15, 10))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm')

plt.title('Матрица корреляции')

plt.show()

X\_cols = ["Ram", "SSD", "ppi", "Gpu brand", 'Ips', 'Cpu brand']

y\_col = "Price"

import statsmodels.api

statsmodels.sandbox.stats.runs.runstest\_1samp(data[y\_col], cutoff="median")

X = data[X\_cols]

y = data[y\_col]

# Разделение данных на обучающую и тестовую выборки

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

from sklearn.linear\_model import LinearRegression

# Обучение модели

model = LinearRegression().fit(X\_train, y\_train)

import sklearn.metrics as metrics

y\_model = model.predict(X\_test)

R2 = metrics.r2\_score(y\_test, y\_model)

print(R2)

from scipy import stats

def Ftest(alpha, df1, df2, R2):

    F = (R2 / (1 - R2)) \* (df2 / df1)

    Fcrit = stats.f.ppf(1 - alpha, dfn=df1, dfd=df2)

    return F >= Fcrit

alpha = 0.05

df1 = len(X\_cols)  # Количество признаков

df2 = len(X\_test) - df1 - 1  # Степени свободы ошибки

print(f"Гипотеза о незначимости полной модели отклоняется на уровне значимости {alpha}:",

      Ftest(alpha=alpha, df1=df1, df2=df2, R2=R2))

from itertools import combinations

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import r2\_score

# Перебор всех комбинаций признаков, удаляя по одному признаку

for comb in combinations(X\_cols, r=len(X\_cols) - 1):  # Убираем по одному признаку

    print(f"Комбинация признаков: {comb}")

    # Обучаем модель на текущей комбинации признаков

    model\_comb = LinearRegression().fit(X\_train[list(comb)], y\_train)

    # Предсказания модели

    y\_pred\_comb = model\_comb.predict(X\_test[list(comb)])

    # Вычисление R2

    R2\_comb = r2\_score(y\_test, y\_pred\_comb)

    print(f"\tR2 = {R2\_comb:.4f}")

    # Проверка значимости модели с использованием F-теста

    df1\_comb = len(comb)  # Степени свободы модели

    df2\_comb = len(X\_test) - df1\_comb - 1  # Степени свободы ошибки

    print(f"\tГипотеза о незначимости модели отклоняется на уровне значимости {alpha}:",

          Ftest(alpha=alpha, df1=df1\_comb, df2=df2\_comb, R2=R2\_comb))

print(f"Коэффициенты модели: {model.coef\_}, свободный член: {model.intercept\_}")

import matplotlib.pyplot as plt

sorted\_indices = X\_test.sort\_values(by='Ram').index

X\_test\_sorted = X\_test.loc[sorted\_indices]

y\_model\_sorted = model.predict(X\_test\_sorted)

# Построение графика

plt.figure(figsize=(10, 6))

# Scatter plot для реальных данных

plt.scatter(X\_test['Ram'], y\_test, color="blue", label="Реальные данные")

# Линия регрессии

plt.plot(X\_test\_sorted['Ram'], y\_model\_sorted, color="red", label="Модель линейной регрессии")

# Настройка меток и заголовка

plt.xlabel('Ram')

plt.ylabel('Price')

plt.title('Зависимость цены ноутбука от объема оперативной памяти')

plt.legend()

y\_pred = model.predict(X\_test)

R2 = metrics.r2\_score(y\_test, y\_pred)

print(R2)

# Отображение графика

plt.show()

y\_pred = model.predict(X\_test)

R2 = metrics.r2\_score(y\_test, y\_pred)

print(R2)

import matplotlib.pyplot as plt

import numpy as np

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import PolynomialFeatures

# Создание модели полиномиальной регрессии

model = Pipeline([

    ('poly\_features', PolynomialFeatures(degree=3)),

    ('linear\_regression', LinearRegression())

])

# Обучение модели

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_train)

r2 = r2\_score(y\_train, y\_pred)

print("Коэффициент детерминации на обучающей выборке:", r2)

print()

# Предсказание цен ноутбуков

# Оценка качества модели

from sklearn.metrics import r2\_score

y\_pred = model.predict(X\_test)

r2 = r2\_score(y\_test, y\_pred)

print("Коэффициент детерминации на тестовой выборке:", r2)

plt.scatter(y\_test, y\_pred)

plt.xlabel("Реальные цены")

plt.ylabel("Предсказанные цены")

plt.title("Реальные и предсказанные цены ноутбуков")

# Добавление линии тренда

z = np.polyfit(y\_test, y\_pred, 3)

p = np.poly1d(z)

plt.plot(y\_test, p(y\_test), "r")

plt.show()

import matplotlib.pyplot as plt

sorted\_indices = X\_test.sort\_values(by='Ram').index

X\_test\_sorted = X\_test.loc[sorted\_indices]

y\_model\_sorted = model.predict(X\_test)

# Построение графика

plt.figure(figsize=(10, 6))

# Scatter plot для реальных данных

plt.scatter(X\_test['Ram'], y\_test, color="blue", label="Реальные данные")

# Линия регрессии

plt.plot(X\_test\_sorted['Ram'], y\_model\_sorted, color="red", label="Модель линейной регрессии")

# Настройка меток и заголовка

plt.xlabel('Ram')

plt.ylabel('Price')

plt.title('Зависимость цены ноутбука от объема оперативной памяти')

plt.legend()

from sklearn.metrics import r2\_score

y\_pred\_test = model.predict(X\_test)

r2\_test = r2\_score(y\_test, y\_pred)

print(f"R^2 на тестовой выборке: {r2\_test}")

# Отображение графика

plt.show()