МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное образовательное
учреждение высшего образования
ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

Институт математики, механики и компьютерных наук
имени И. И. Воровича

Направление подготовки 02.03.02 — "Фундаментальная
информатика и информационные технологии"

Выпускная квалификационная работа
на степень бакалавра

Рахуба А.М., 4 курс, 9 группа

РЕШЕНИЕ ЗАДАЧИ АВТОМАТИЧЕСКОГО ОПРЕДЕЛЕНИЯ ЦЕНЫ НА ПРОДУКТЫ ПО ИХ ТЕКСТОВОМУ ОПИСАНИЮ

Научный руководитель:
к.ф.-м.н., доцент А. В. Абрамян

Допущено к защите:
Заместитель директора ИММКН
по направлению ФИИТ

В.С. Пилиди

Ростов-на-Дону
2020

Отзыв научного руководителя
о выпускной квалификационной работе Рахубы А.М.
«Решение задачи автоматического определения цены на продукты по их текстовому описанию»

Перед бакалавром Рахубой А.М. была поставлена цель написать приложение, решающее задачу автоматического определения цены на продукты по их текстовому описанию, для этого необходимо решить следующие задачи: исследовать входные данные и выполнить их предобработку, построить модель, которая автоматически предлагает правильные цены на продукцию по ее текстовому описанию, исследовать эффективность созданной модели.

С поставленной задачей бакалавр Рахуба А.М. полностью справился, разработав и подробно описав соответствующее приложение.

Работал бакалавр Рахуба А.М. добросовестно и регулярно.

Считаю, что квалификационная работа бакалавра Рахубы А.М. на тему «Решение задачи автоматического определения цены на продукты по их текстовому описанию» полностью отвечает требованиям, предъявляемым к выпускной квалификационной работе, и заслуживает оценки «отлично».

К. ф.-м. н., доцент
кафедры информатики
и вычислительного эксперимента

А.В. Абрамян

Задание на квалификационную работу бакалавра

Направление подготовки: 02.03.02 — Фундаментальная информатика и информационные технологии

Студент: Рахуба А.М.

Научный руководитель: к.ф.-м.н., доцент Абрамян А.В.

Год защиты: 2020

Тема работы: Решение задачи автоматического определения цены на продукты по их текстовому описанию

Цель работы*:* написать программу, решающую задачу автоматического определения цены на продукты по их текстовому описанию

Задачи работы:

1. исследовать входные данные и выполнить их предобработку;
2. построить модель, которая автоматически предлагает правильные цены на продукцию по ее текстовому описанию;
3. исследовать эффективность созданной модели.

Научный руководитель  Абрамян А. В.

Студент  Рахуба А. М.

АНТИПЛАГИАТ
ТВОРИТЕ СОБСТВЕННЫМ УМОМ

Южный Федеральный Университет

СПРАВКА

о результатах проверки текстового документа на наличие заимствований

Проверка выполнена в системе Антиплагиат.ВУЗ

Автор работы	**Рахуба А.М.**
Подразделение	Институт математики, механики и компьютерных наук
Тип работы	Выпускная квалификационная работа
Название работы	Рахуба Алексей ВКР
Название файла	Рахуба Алексей ВКР.pdf
Процент заимствования	**4.72 %**
Процент самоцитирования	**0.00 %**
Процент цитирования	**5.19 %**
Процент оригинальности	**90.09 %**
Дата проверки	**19:05:37 17 июня 2020г.**
Модули поиска	Модуль поиска ИПС "Адилет"; Модуль выделения библиографических записей; Сводная коллекция ЭБС; Модуль поиска "Интернет Плюс"; Коллекция РГБ; Цитирование; Модуль поиска переводных заимствований; Модуль поиска переводных заимствований по elibrary (EnRu); Модуль поиска переводных заимствований по интернет (EnRu); Коллекция eLIBRARY.RU; Коллекция ГАРАНТ; Коллекция Медицина; Диссертации и авторефераты НББ; Модуль поиска перефразирований eLIBRARY.RU; Модуль поиска перефразирований Интернет; Коллекция Патенты; Модуль поиска "ЮФУ"; Модуль поиска общеупотребительных выражений; Кольцо вузов
Работу проверил	Абрамян Анна Владимировна ФИО проверяющего
Дата подписи	Подпись проверяющего

Чтобы убедиться в подлинности справки, используйте QR-код, который содержит ссылку на отчет.

Ответ на вопрос, является ли обнаруженное заимствование корректным, система оставляет на усмотрение проверяющего. Предоставленная информация не подлежит использованию в коммерческих целях.

Содержание

Введение

В сегодняшнем мире оценить, сколько объективно стоит тот или иной товар, может являться довольно сложной задачей. Различные небольшие детали могут порождать большие различия в цене. Например, цены на одежду могут сильно колебаться в зависимости от сезонных трендов и влияний брендов, цены на электронику зависят от множества различных технических характеристик, а цены на покупки в интернете могут меняться в зависимости от времени и типа доставки и т.д. К примеру, один из этих свитеров стоит 335$, а другой 10$ (см. рис. 1).

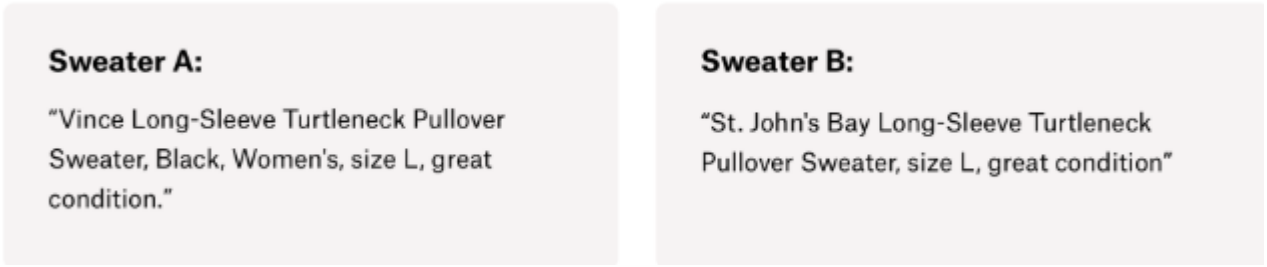

Рисунок 1 — Два свитера с большой разницей в цене

Смогли бы вы понять, какой свитер сколько стоит?
Mercari – крупнейшее в Японии приложение для совершения онлайн-покупок, в котором люди могут продавать и покупать различные новые и бывшие в употреблении товары разных брендов, начиная с одежды и электроники и заканчивая товарами ручной работы (см. рис. 2)
Mercari хотели бы внедрить в своё приложение систему, которая могла бы помочь пользователям объективно оценить свой продукт при подаче объявления. Но так как пользователи могут размещать на платформе Mercari практически любые товары, это может быть непростой задачей [1].

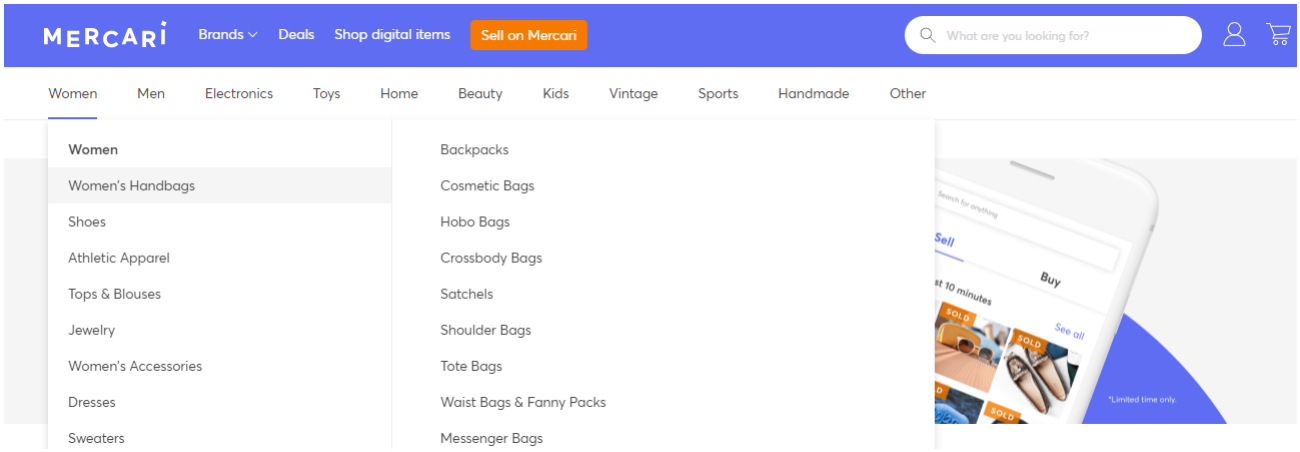

Рисунок 2 — Приложение Mercari

1. Описание данных

	train_id	name	item_condition_id	category_name	brand_name	price	shipping	item_description
0	0	MLB Cincinnati Reds T Shirt Size XL	3	Men/Tops/T-shirts	NaN	10.0	1	No description yet
1	1	Razer BlackWidow Chroma Keyboard	3	Electronics/Computers & Tablets/Components & P...	Razer	52.0	0	This keyboard is in great condition and works ...
2	2	AVA-VIV Blouse	1	Women/Tops & Blouses/Blouse	Target	10.0	1	Adorable top with a hint of lace and a key hol...
3	3	Leather Horse Statues	1	Home/Home Décor/Home Décor Accents	NaN	35.0	1	New with tags. Leather horses. Retail for [rm]...
4	4	24K GOLD plated rose	1	Women/Jewelry/Necklaces	NaN	44.0	0	Complete with certificate of authenticity

Рисунок 3 — Несколько примеров из данных

Данные состоят из списка, содержащего 1482535 различных товаров. Каждый товар содержит в себе следующие поля:

- **train_id** - номер товара;
- **name** - текстовое название товара;
- **item_condition_id** ($1 \leqslant$ item_condition_id $\leqslant 5$) - состояние товара, которое указал продавец (1 - самое лучшее состояние, 5 - худшее);

- **category_name** - категория товара в виде трёхуровневой иерархии (разделенная символом /);
- **brand_name** - бренд товара;
- **price** - цена товара, по которой товар был продан (целевая переменная, которую будем предсказывать);
- **shipping** - тип доставки (1 - доставка оплачивается продавцом, 0 - доставка оплачивается покупателем);
- **item_description** - подробное текстовое описание товара.

2. Исследовательский анализ данных

Исследовательский анализ данных является важным процессом выполнения первоначальных исследований данных в целях выявления закономерностей и аномалий, характера и свойств анализируемых данных с помощью сводной статистики и графических представлений.

Для начала проанализируем целевую переменную **price**(см. рис. 4, 5).

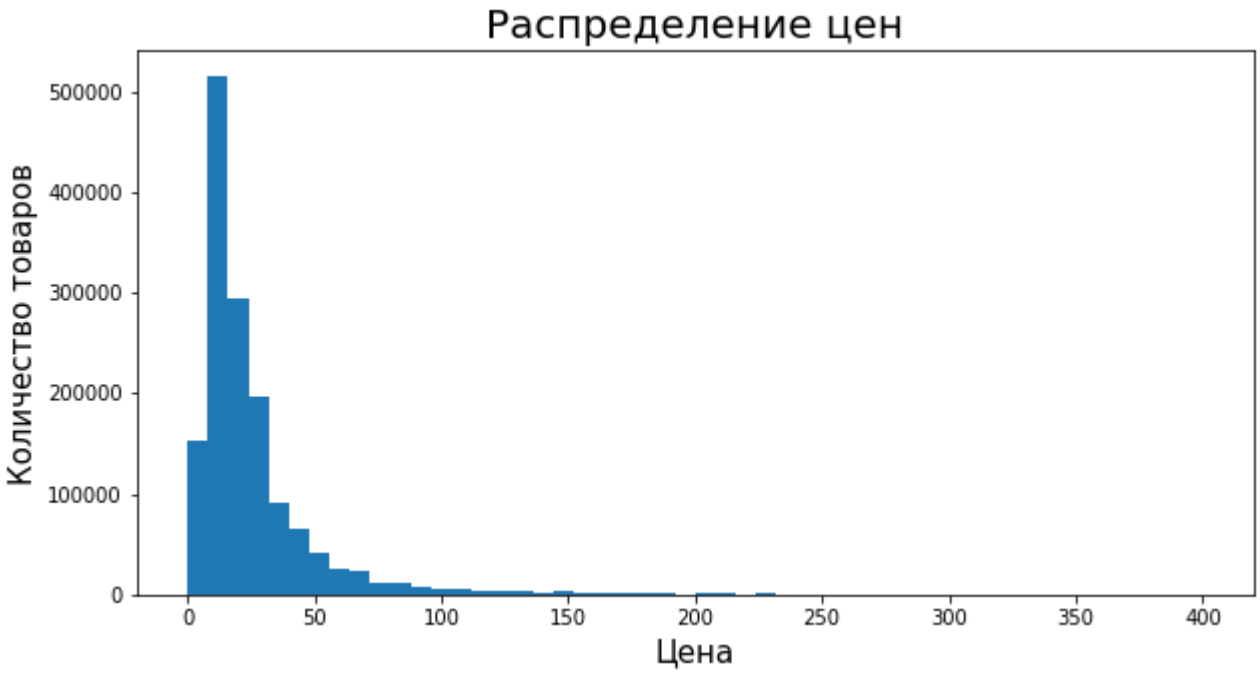

Рисунок 4 — Распределение целевой переменной

count	1.482535e+06
mean	2.673752e+01
std	3.858607e+01
min	0.000000e+00
25%	1.000000e+01
50%	1.700000e+01
75%	2.900000e+01
max	2.009000e+03

Рисунок 5 — Статистические данные

Таким образом, видим, что распределение цены сильно смещено к минимальной границе (75% всех товаров имеют цену ниже 29 долларов). Медианная цена равна 17 долларам. Так же во время анализа было определено, что данные содержат 874 товаров с ценой менее либо равной нуля. Данные товары были удалены из датасета.

Проанализируем **brand_name**. Видим, что сразу же в первых строках бренд не задан (значение NaN). В процентном соотношении бренд не

задан в 42.68 %. Проверим остальные признаки на наличие пропущенных значений (см. рис. 6).

```
В train_id пропущено 0.0 %
В name пропущено 0.0 %
В item_condition_id пропущено 0.0 %
В category_name пропущено 0.42676901388500105 %
В brand_name пропущено 42.675687251902986 %
В price пропущено 0.0 %
В shipping пропущено 0.0 %
В item_description пропущено 0.0002698081326916397 %
```

Рисунок 6 — Пропущенные значения в признаках

Как видим, пропущенные значения присутствуют в **brand_name**, **category_name**, **item_description**. Так как пропущенные значения в brand_name составляют значительную часть данных, сделаем это отдельной категорией, заменив их на текстовое значение 'none'. Товары с пропущенными значениями в category_name и item_description удалим.

Посмотрим на распределение товаров по их брендам (см. рис. 7).

Рисунок 7 — Распределение товаров по брендам

Можем заметить, что бренд не определен у абсолютного большинства товаров. Посмотрим, как определенность бренда влияет на цену(см. рис. 8).

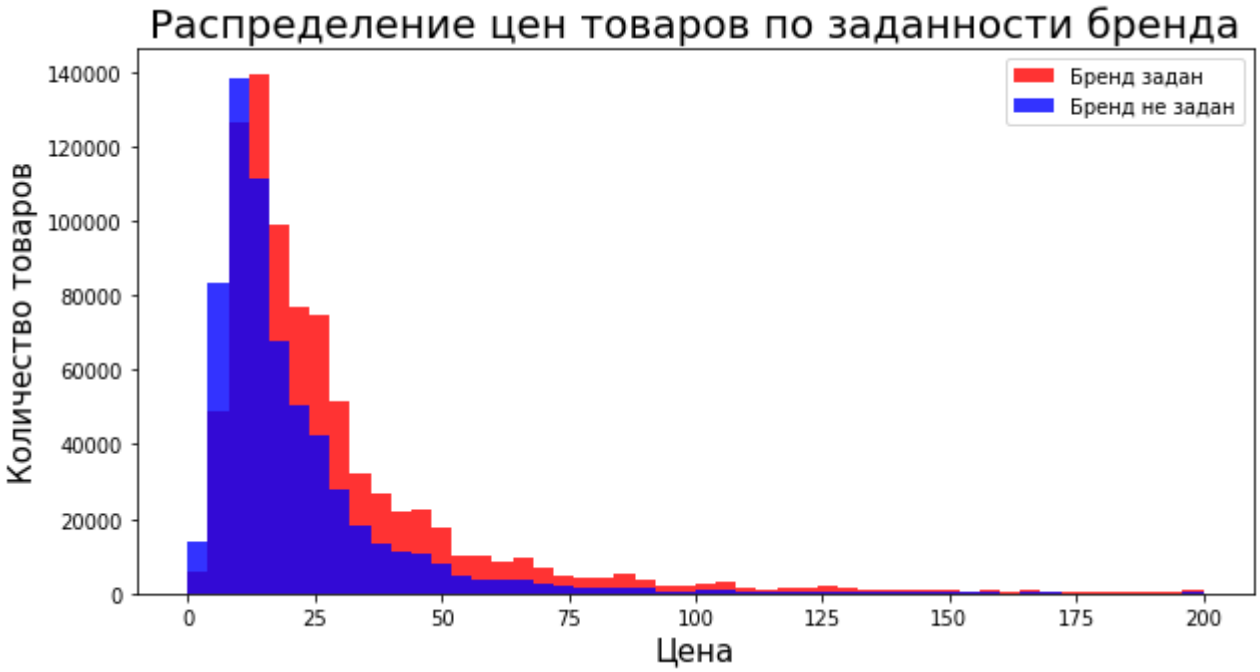

Рисунок 8 — Взаимосвязь определённости бренда и цены

Медианная цена товаров, у которых бренд задан: 20.0.
Медианная цена товаров, у которых бренд не задан: 14.0.
Как мы видим, медианная цена товаров заметно выше, если у них задан бренд. Добавим новый бинарный признак в датасет: 1 - бренд задан, 0 - бренд не задан.
Далее исследуем **shipping** и посмотрим на его связь с ценой.
0: 55.27 %
1: 44.73 %,
где 0 - доставка оплачивается покупателем, 1 - доставка оплачивается продавцом. То есть, доставка 55% товаров оплачивается покупателем, 45% продавцом.
Из графиков видно (см. рис. 9, 10), что медианная цена по одному типу доставки заметно выше другой [2].

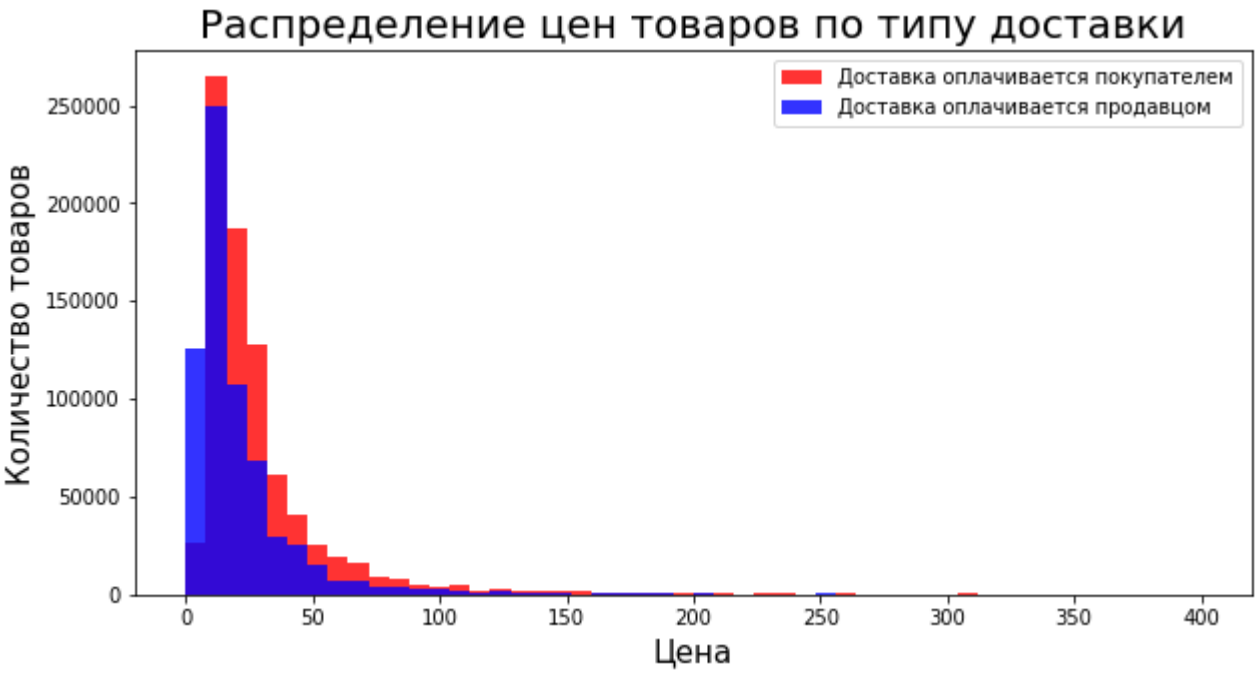

Рисунок 9 — Распределение товаров по типу доставки

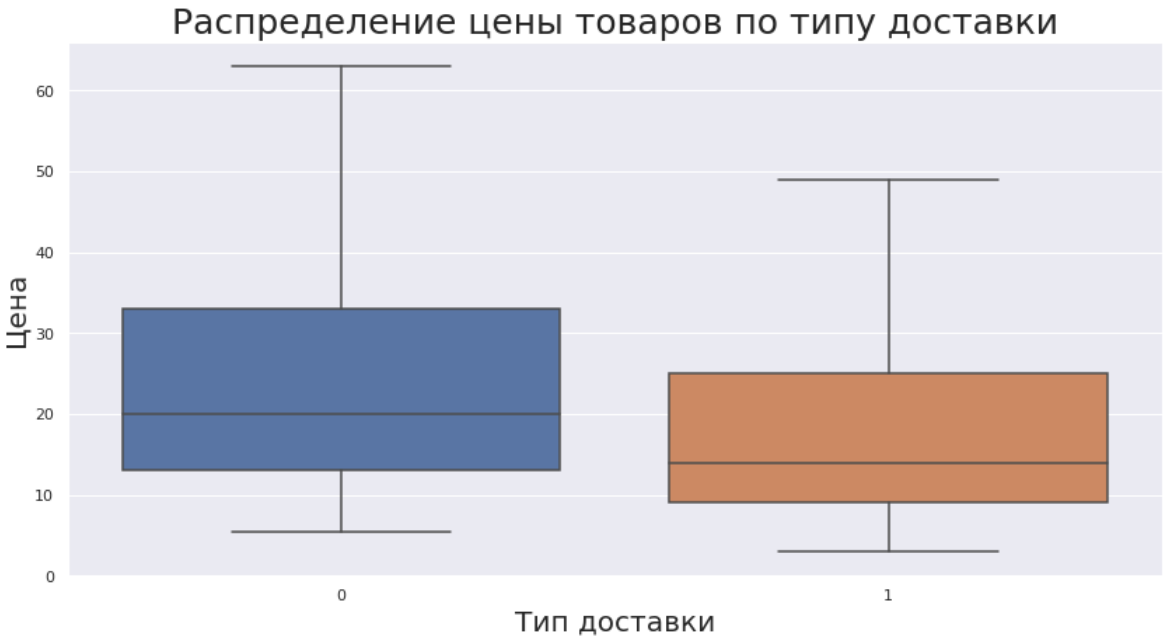

Рисунок 10 — Распределение цены товаров по типу доставки

Медианная цена товаров, доставку которых оплачивает покупатель: 20.0.
Медианная цена товаров, доставку которых оплачивает продавец: 14.0.
Исследуем **item_condition_id**:

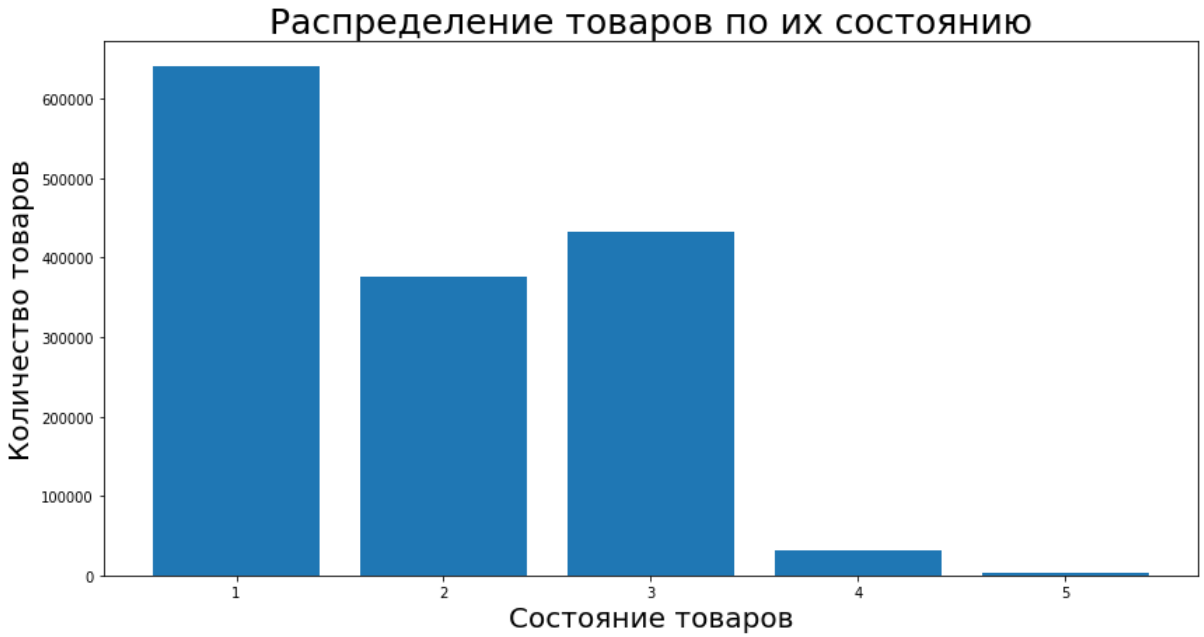

Рисунок 11 — Распределение товаров по их состоянию

1: 43.21 %
2: 25.33 %
3: 29.15 %
4: 2.16 %
5: 0.16 %, где 1 - самое лучшее состояние, 5 - худшее.
Таким образом, можем отметить, что большинство продавцов оценили состояние своего товара как наилучшее. И только 0.16% товаров имеют указанное худшее состояние.
Посмотрим на то, как распределяется цена товаров в зависимости от их состояния (см. рис. 12).

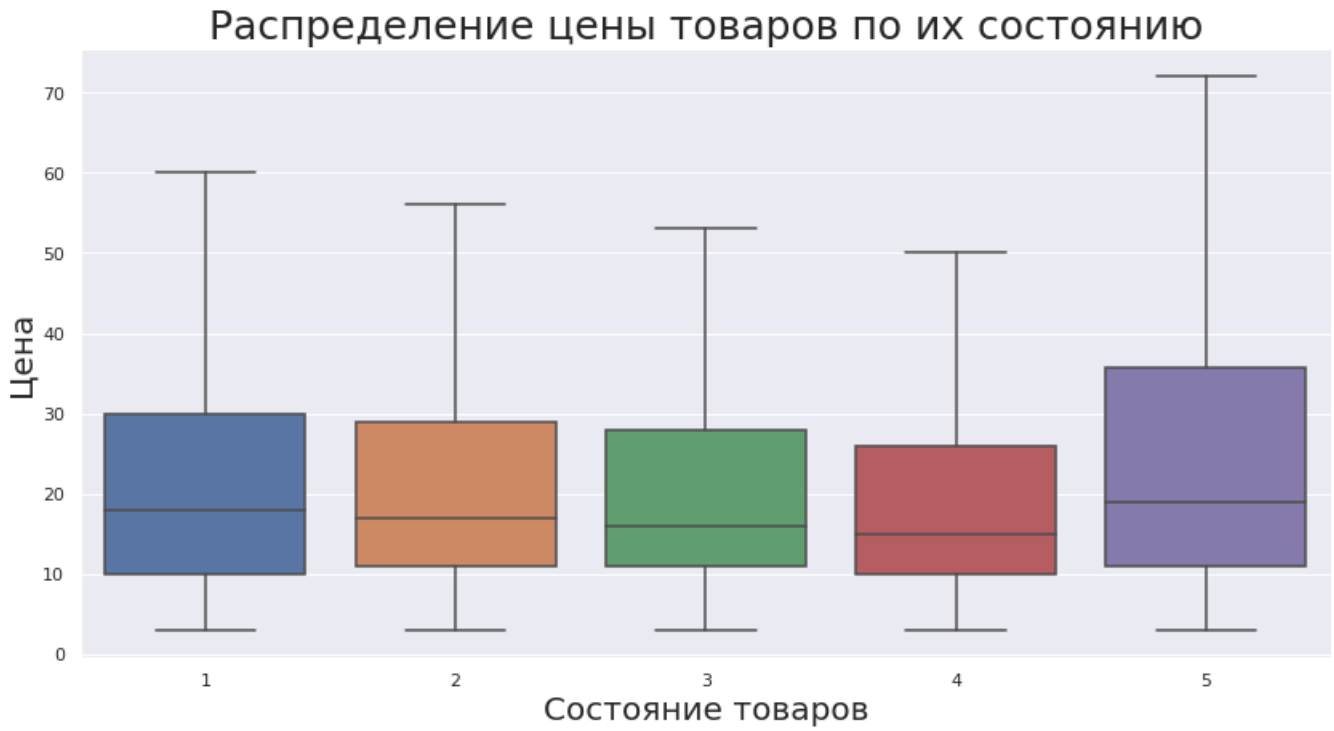

Рисунок 12 — Распределение цены товаров по их состоянию

Медианная цена товаров с состоянием 1: 18.0.
Медианная цена товаров с состоянием 2: 17.0.
Медианная цена товаров с состоянием 3: 16.0.
Медианная цена товаров с состоянием 4: 15.0.
Медианная цена товаров с состоянием 5: 19.0.

Медианная цена предсказуемо соотносится с состоянием товаров: чем лучше состояние, тем медианная цена выше. Исключение лишь составляют товары с самым худшим состоянием. Их медианная цена выше медианной цены товаров с более хорошим состоянием. После просмотра некоторых таких товаров становится ясно, что более высокая цена связана с тем, что эти товары могут являться винтажными вещами известных брендов, частями сломанной электроники и т.д (см. рис. 13).

653	653	Fossil vintage renewal purse	5	Women/Women's Handbags/Shoulder Bag	Fossil	36.0	0	No description yet

Gameboy advance sp ags-101 FOR PARTS	5	Other/Other/Other	none	24.0	0	For parts . Turns on and hold battery charge ....
Hobo International Lauren Wallet	5	Women/Women's Accessories/Wallets	none	24.0	0	Pretty wallet but the back closing clip is bro...

Рисунок 13 — Вещи с худшим состоянием и высокой медианной ценой

Исследуем **category_name**. Как мы могли заметить, категории состоят из троих частей в отношении вложенности. Таким образом, сначала разделим категории на составные части и добавим их в датасет.
Первая подкатегория содержит 10 уникальных значений.
Вторая подкатегория содержит 113 уникальных значений.
Третья подкатегория содержит 863 уникальных значений.
Проанализируем распределение подкатегорий в датасете и посмотрим на их связь с ценой.

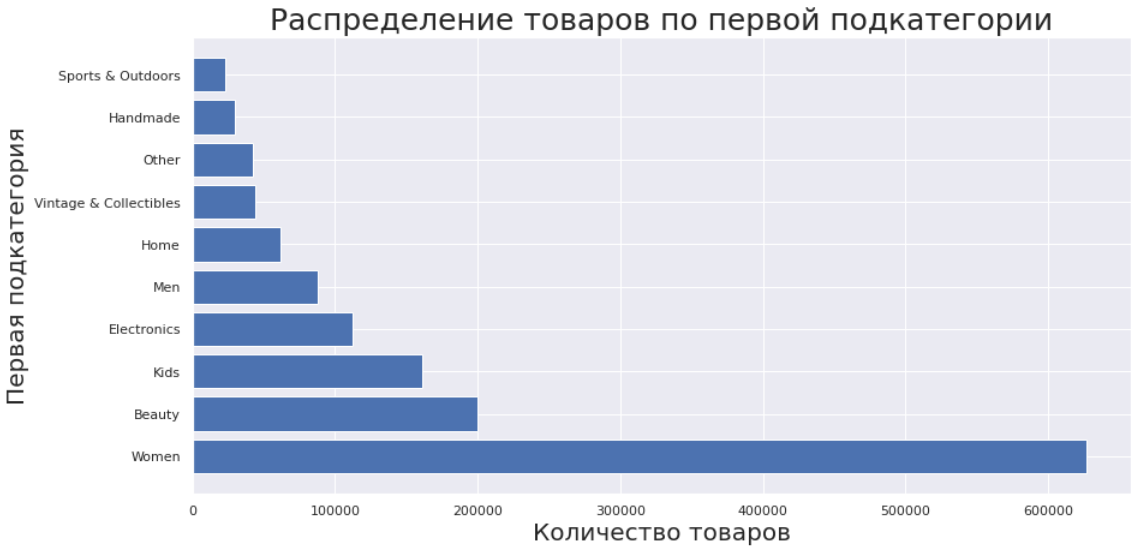

Рисунок 14 — Распределение товаров по первой подкатегории

Как можем заметить, подавляющее большинство товаров имеют первую подкатегорию Women.

```
Women: 45.1 %
Beauty: 14.39 %
Kids: 11.58 %
Electronics: 8.07 %
Men: 6.3 %
Home: 4.48 %
Vintage & Collectibles: 3.18 %
Other: 3.08 %
Handmade: 2.14 %
Sports & Outdoors: 1.67 %
```

Рисунок 15 — Процентное соотношение первых подкатегорий

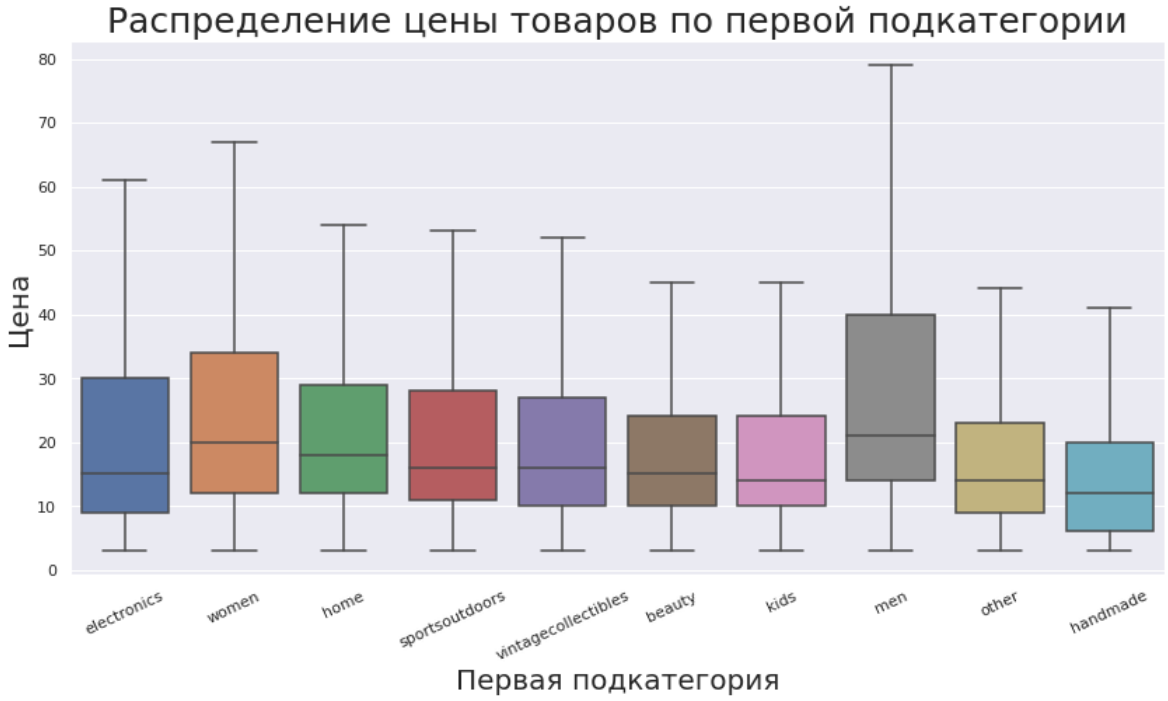

Рисунок 16 — Распределение цен товаров по первой подкатегории

Первые 10 вторых подкатегорий по популярности:

Рисунок 17 — Распределение товаров по второй подкатегории

```
Athletic Apparel: 9.06 %
Makeup: 8.41 %
Tops & Blouses: 7.21 %
Shoes: 6.78 %
Jewelry: 4.17 %
Toys: 3.92 %
Cell Phones & Accessories: 3.59 %
Dresses: 3.09 %
Women's Handbags: 3.09 %
Women's Accessories: 2.86 %
```

Рисунок 18 — Процентное соотношение вторых подкатегорий

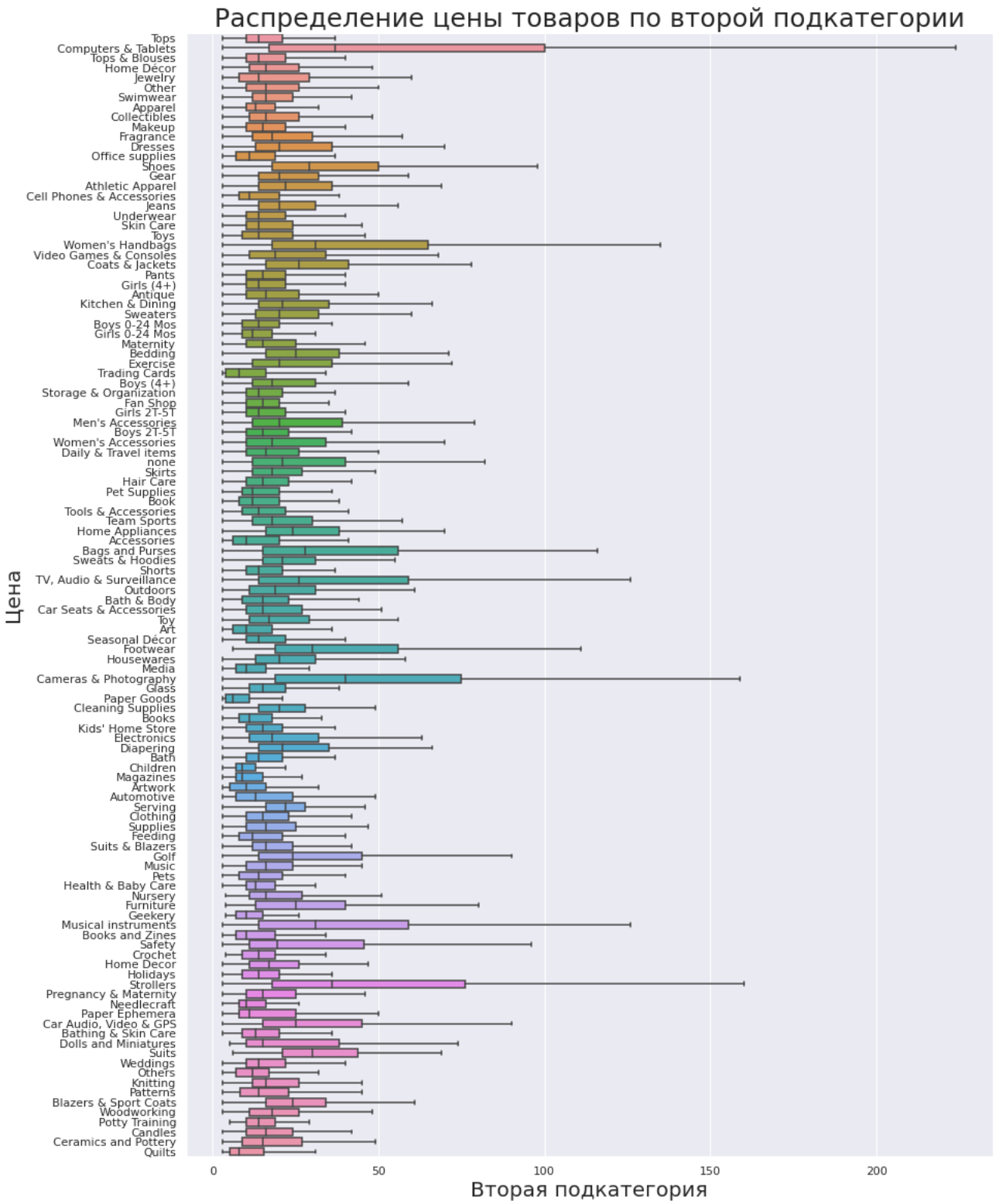

Рисунок 19 — Распределение цен товаров по второй подкатегории

Топ 300 самых популярных слов, встречающихся в названиях товаров (см. рис. 20):

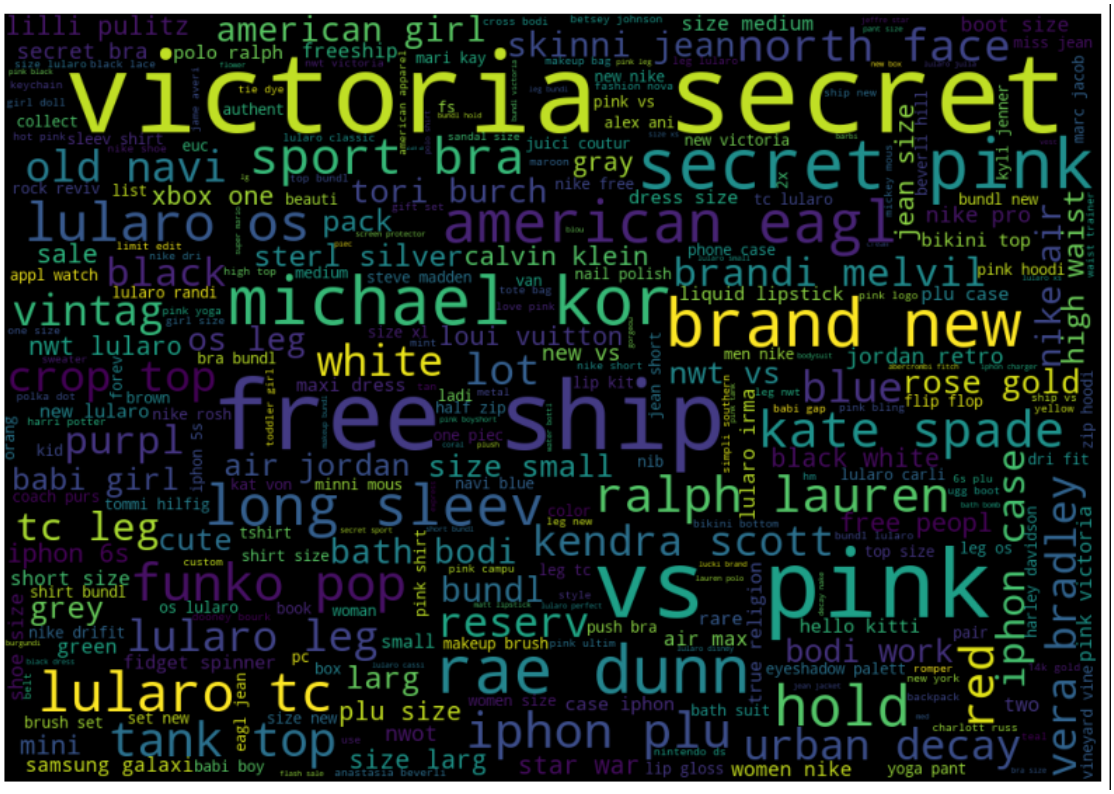

Рисунок 20 — 300 самых популярных слов из названий товаров

Топ 300 самых популярных слов, встречающихся в описаниях товаров (см. рис. 21):

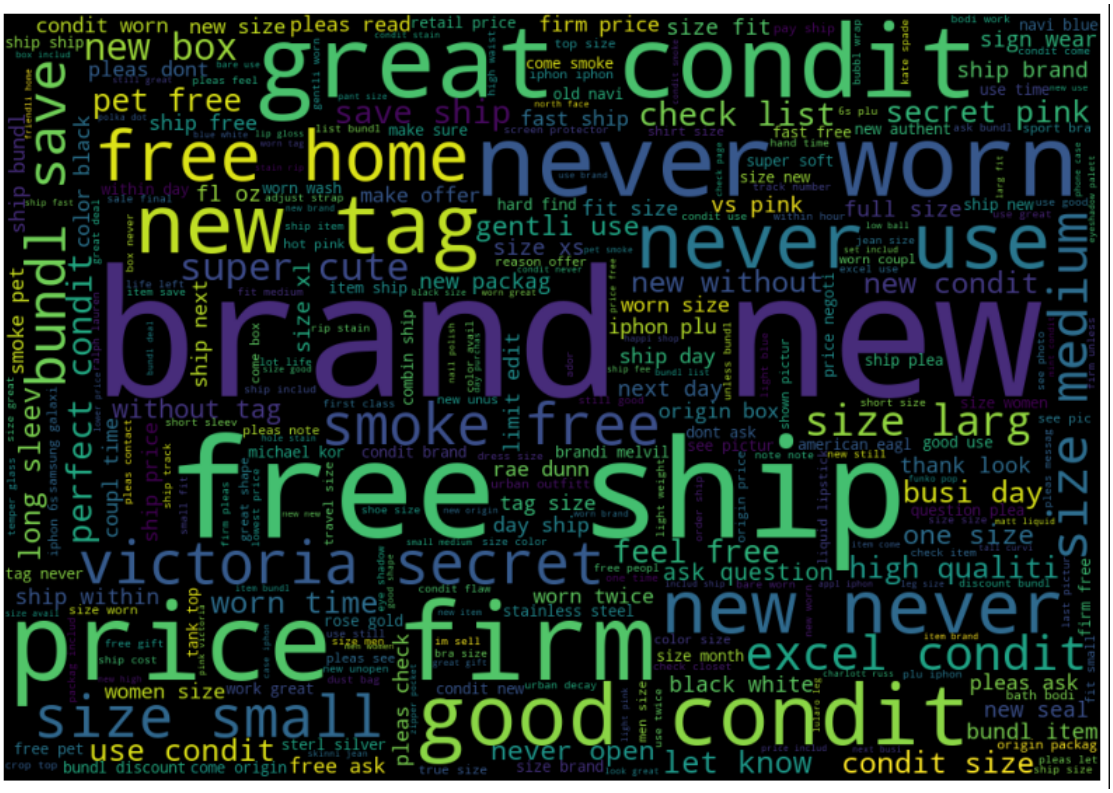

Рисунок 21 — 300 самых популярных слов из описаний товаров

Можем заметить, что многие продавцы указывают в своих описаниях товаров такие слова, как "новый", "хороший", "прекрасный", "бесплатная доставка" и т.д., что может говорить о том, что они хотят тем самым произвести впечатление на покупателей положительными характеристиками.

Далее добавим новый признак в датасет - количество слов в **item_description** и посмотрим на его связь с ценой (см. рис. 22).

Рисунок 22 — Зависимость цен товаров от количества слов в описаниях

Можем отметить, что наблюдается тенденция, при которой с увеличением количества слов в описании уменьшается цена товаров. Таким образом, данный признак может быть нам полезен.

3. Предобработка данных

В первую очередь обратим внимание на данные с некорректной ценой. Было обнаружено порядка 900 товаров с ценой менее либо равной нулю. Данные объявления о продаже возможно были созданы вследствие человеческой ошибки, поэтому мы их просто удалим, воспринимая как некую погрешность. Затем проверим данные на наличие дубликатов и удалим их. А также удалим все строки, содержащие пустые значения.

Так как **item_description** и **name** содержат неструктурированные текстовые данные, то необходимо их сначала обработать. Удалим пунк-

туационные знаки и символы, которые не представляют из себя никакой ценности для обучения. Приведем все слова к нижнему регистру для однообразности и удалим стоп-слова (союзы, междометия, артикли и т.д.), которые не несут полезной информации, а являются лишь "шумом". Затем проведем анализ настроений описаний товаров. Анализ настроений является распространенной задачей обработки естественного языка, которая включает в себя классификацию текстовых данных в определенное мнение. Мы хотим оценить эмоциональную окраску описаний товаров, предполагая, что это может коррелировать с целевой переменной цены. Будем использовать для этой цели функцию SentimentIntensityAnalyzer из библиотеки NLTK (The Natural Language Toolkit). В результате анализа мы получим положительную, отрицательную, нейтральную и совокупную оценки настроения описаний товаров. Построим таблицу корреляции наших признаков (см. рис. 23). Матрица корреляции может использоваться для оценки линейной взаимосвязи между различными признаками.

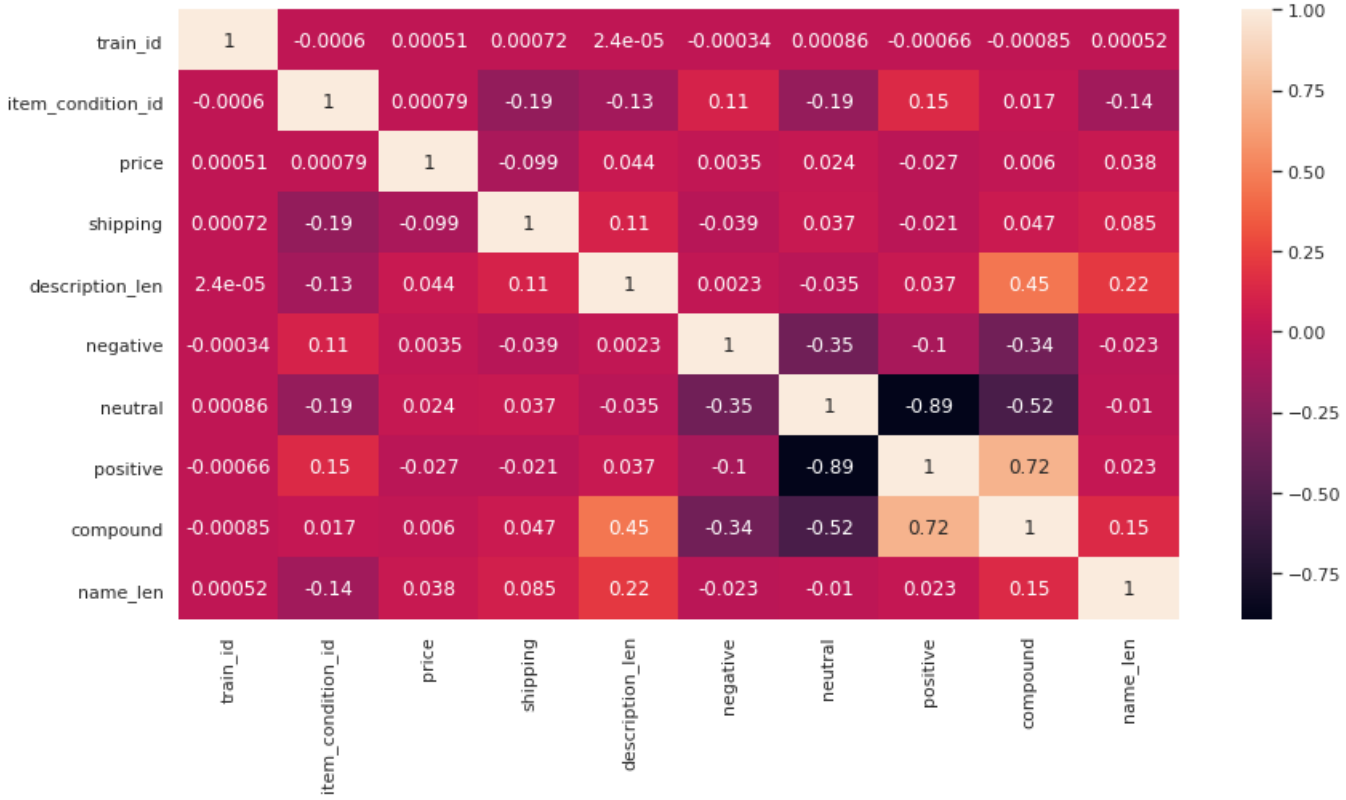

Рисунок 23 — Матрица корреляции признаков

Можем заметить, что действительно существует некоторая корреляция между целевой переменной цены и добавленными признаками. Нормализуем новые признаки, используя так называемое Z-масштабирование:

$$Z = \frac{x - \mu}{\sigma},$$

где x - признак,
$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$ - среднее значение,
$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$ - стандартное отклонение [3].

Следующим шагом, чтобы сделать единообразными для моделей решений все слова, которые несут один и тот же смысл, но образованы с разными окончаниями, суффиксами, приставками и т.д., приведём все слова к единой форме, используя алгоритм стемминга.

Существует несколько типов алгоритмов стемминга, которые различаются по отношению производительности, точности. В данной задаче был использован алгоритм стемминга Портера, потому что он достаточно эффективен, так как вместо поиска по таблице слов, он просто применяет некоторые правила, которые опираются на особенности используемого языка, отсекая у слова суффиксы, окончания и оставляя основу слова [4].
Примеры правил для английского языка:

- отсечь ’ed’, если слово оканчивается на ’ed’;
- отсечь ’s’, если слово оканчивается на ’s’;
- отсечь ’ing’, если слово оканчивается на ’ing’.

Пример: описанные выше преобразования будут действовать следующим образом:
“The boy’s dogs are different size.” $\Longrightarrow$ “boy dog differ size”
После произведенной вышеописанной предварительной обработки данных, можем приступить к векторизации категориальных и текстовых признаков. Так как многие алгоритмы машинного обучения не могут работать непосредственно с категориальными и текстовыми признаками, необходимо, чтобы все входные и выходные переменные были числовыми.
Приведем категориальные признаки **category**, **brand_name**, **item_condition_id**, **shipping** к числовому виду, используя one-hot кодирование. В результате этого кодирования каждому признаку будет соответствовать N новых бинарных признаков, где N — число всех возможных категорий в этом признаке (единица на месте категории текущего признака и нули на всех остальных местах).

Листинг 3.1. Пример one-hot кодирования

```
from sklearn.preprocessing import OneHotEncoder
import numpy as np
oh = OneHotEncoder(handle_unknown='ignore')
X = np.array(['dog',
              'cat',
              'mouse'])
X = oh.fit_transform(X.reshape(-1,1))
print(oh.categories_)
print(X.toarray())
```

```
[array(['cat', 'dog', 'mouse'], dtype='<U5')]
[[0. 1. 0.]
 [1. 0. 0.]
 [0. 0. 1.]]
```

Рисунок 24 — Вывод примера

Приведем признак **name** из текстового вида в числовой. Для начала токенизируем каждый признак в датасете на отдельные слова, на биграммы (пары соседних слов) и на триграммы (тройки соседних слов). Для векторизации наших признаков будем использовать представление мешок слов с подсчётом числа вхождений токенов в текущий документ. В результате применения этого представления каждому признаку будет соответствовать N новых признаков, где N — число уникальных токенов во всей коллекции документов.

Листинг 3.2. Пример представления текстовых данных в числовом виде с помощью мешка слов

```
from sklearn.feature_extraction.text import CountVectorizer
corpus = ['This is the first document.',
          'This document is the second document.',
          'And this is the third one.',
          'Is this the first document?']
cv = CountVectorizer()
X = cv.fit_transform(corpus)
print(cv.get_feature_names())
print(X.toarray())
```

```
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
[[0 1 1 1 0 0 1 0 1]
 [0 2 0 1 0 1 1 0 1]
 [1 0 0 1 1 0 1 1 1]
 [0 1 1 1 0 0 1 0 1]]
```

Рисунок 25 — Вывод примера

Приведем признак **item_description** из текстового вида в числовой, используя представление мешок слов с подсчётом частотности слов (чем чаще слово встречается во всём датасете - тем оно менее важно) с помощью алгоритма TF-IDF [5].

TF-IDF используется для оценки важности слова в контексте документа, являющегося частью коллекции документов. Вес некоторого слова пропорционален частоте употребления этого слова в документе и обратно пропорционален частоте употребления слова во всех документах коллекции.

$$TFIDF_a = TF_a \times IDF_a,$$

где $TF_a = k/N$ - частота употребления токена в пределах отдельного

документа, где k - количество раз, когда токен a встретился в документе, N - количество всех слов в документе,
$IDF_a = log(D/m)$ - инверсия частоты, с которой некоторое слово встречается в документах коллекции, где D - общее количество документов, m - количество документов, в которых встречается токен .
Сначала **item_description** токенизируем на отдельные слова, на биграммы (пары соседних слов) и на триграммы (тройки соседних слов), после чего применяем алгоритм TF-IDF.

Листинг 3.3. Пример представления текстовых данных в числовом виде с помощью TF-IDF

```
from sklearn.feature_extraction.text import TfidfVectorizer
corpus = ['This is the first document.',
          'This document is the second document.']
tv = TfidfVectorizer(ngram_range=(1,2), max_features=10)
X = tv.fit_transform(corpus)
print(tv.get_feature_names())
print(X.toarray())
```

```
['document', 'document is', 'first', 'first document', 'is', 'is the', 'second', 'second document', 'the', 'this']
[[0.33425073 0.         0.46977774 0.46977774 0.33425073 0.33425073
  0.         0.         0.33425073 0.33425073]
 [0.53594084 0.37662308 0.         0.         0.26797042 0.26797042
  0.37662308 0.37662308 0.26797042 0.26797042]]
```

Рисунок 26 — Вывод примера

4. Метрика оценки

Для оценки точности наших моделей решения данной задачи использовалась метрика RMSLE (Root Mean Squared Logarithmic Error).

$$RMSE(y_i, \hat{y}_i) = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

$$RMSLE(y_i, \hat{y}_i) = \sqrt{\frac{\sum_{i=1}^{n}(\log(y_i + 1) - \log(\hat{y}_i + 1))^2}{n}} =$$

$$= RMSE(\log(y_i + 1), \log(\hat{y}_i + 1)),$$

где n - общее количество примеров в датасете,
y_i - фактическая цена,
$\hat{y}_i$ - прогноз цены от модели.
Особенности данной метрики оценки:

- устойчива к воздействию выбросов;
  Товары, которые будут иметь очень высокое значение цены, будут иметь немного более высокую ошибку по сравнению с товарами с низкой стоимостью, потому что RMSLE имеет логарифмическое выражение в метрике, которое смягчает этот эффект. Если же оценивать с помощью RMSE, то такие выбросы могут исказить модель.

- учитывает только относительную ошибку, а не абсолютную;

  $$\log x - \log y = \frac{\log x}{\log y}$$

  Отсюда видно, что благодаря свойству логарифмов RMLSE можно рассматривать как относительную ошибку между прогнозируемыми и фактическими значениями, а масштаб ошибки не имеет значения.

- влечет за собой большее наказание за недооценку целевой переменной, чем за переоценку (см. рис. 27);
  Проще говоря, больше штрафа возникает, когда прогнозируемое значение меньше фактического значения, чем, когда прогнозируемое значение больше фактического значения. Это особенно может быть полезно для некоторых бизнес-проектов. Например,

если модель регрессии, которую мы построили, даёт немного более высокую оценку цены для определенных товаров, чем фактическая, то эта небольшая переоценка является приемлемой. Если же мы недооцениваем цену товара, то пользователю, вероятно, не будет выгодно продавать свой продукт на нашей площадке [6].

Рисунок 27 — Сравнение роста RMSLE при недооценке и переоценке целевой переменной

На левой половине графика RMSLE мы можем заметить, как быстро увеличивается ошибка по мере недооценки целевой переменной, в то время как на правой половине при переоценке ошибка растёт медленнее.

5. Моделирование решений

Объединяем все наши преобразованные признаки вместе в одну матрицу. Затем разделяем признаки и цены на тренировочный и те-

стовый наборы данных в отношении 0.8 к 0.2 соответственно. Тестовый набор нужен для того, чтобы после обучения оценить точность модели на данных, которая она еще не видела и не могла запомнить. Так как в качестве функции оценки точности модели в этой задаче используется $RMSLE(y_i, \hat{y}_i) = \sqrt{\frac{\sum_{i=1}^{n}(\log(y_i+1)-\log(\hat{y}_i+1))^2}{n}}$, считаем $\log(p+1)$, где p - цена, для того, чтобы использовать среднеквадратичную логарифмическую ошибку $RMSE(y_i, \hat{y}_i) = \sqrt{\frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{n}}$ для оценки моделей.

Данная задача представляет собой задачу регрессионного моделирования, так как целевая переменная имеет непрерывный характер. То есть, наше решение должно моделировать зависимость между независимыми переменными (нашими признаками) и зависимой переменной (наша целевая переменная price). Таким образом, модель решения рассматривает характеристики продуктов, которые были похожими, а также цену, по которой товар был продан, и предлагает схожие цены.

Для начала нам нужно задать основу для сравнения результатов. Так как мы работаем с задачей регрессии, то можем использовать медиану всех цен в тренировочном наборе в качестве результата для всех прогнозов. Эта основа будет служить для нас некоей базовой наивной моделью решения, с помощью которой мы посчитаем ошибку на тестовом наборе. После оценки точности данной модели была достигнута оценка RMSLE = 0.749506. Полученная оценка будет отправной точкой, которую мы будем улучшать с помощью более сложных моделей. Если мы не сможем получить точность больше, чем у нашей базовой модели, то это может говорить о том, что нужно выбрать другой подход к решению или что машинное обучение может быть вообще неприменимо для данной задачи [7].

Для подбора гиперпараметров для наших моделей использовалась функция случайного поиска RandomizedSearchCV из scikit-learn [8]. При использовании случайного поиска создается сетка гиперпара-

метров, модель обучается и тестируется на некоторых случайных комбинациях гиперпараметров из этой сетки, и выбирается лучшая комбинация на основе оценки качества на тестовом наборе. Так как тестовый набор данных нам нужен для оценки итоговой проверки точности обученной модели, то нам необходимы еще тестовые выборки для проверки качества обученных моделей при подборе гиперпараметров. Для этого существует техника перекрестной проверки. При использовании перекрестной проверки мы разделяем тренировочный датасет на k частей. Затем на k-1 частях данных проводится обучение модели, а оставшаяся часть используется для тестирования. Данный процесс повторяется k раз. В результате каждая из k частей тренировочного набора используется для тестирования (см. рис. 28). После выполнения k раз вышеописанного процесса результаты тестирования, полученные на каждой итерации, усредняются.

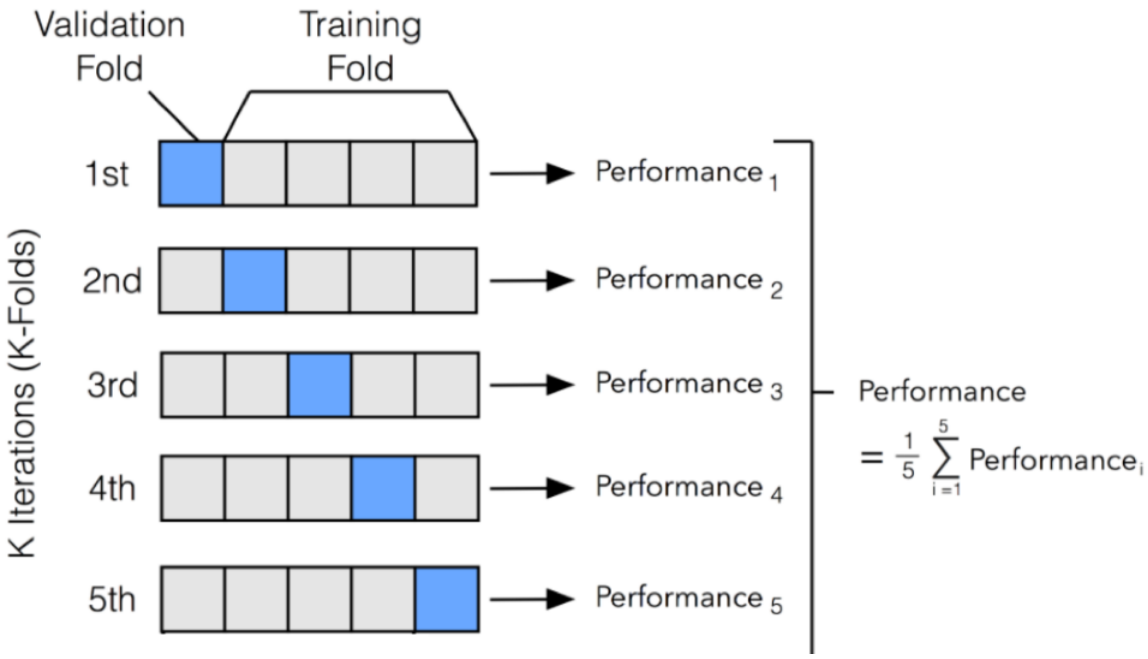

Рисунок 28 — Перекрёстная проверка

В качестве первой модели решения выбрана модель SGDRegressor из библиотеки sklearn. Это модель представляет собой модель линейной

регрессии.

$$y = w_0 + \sum_{i=1}^{n}(w_i x_i) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n,$$

где y – целевая прогнозируемая переменная, x_i – признаки, w_i – коэффициенты. Обучение происходит с помощью алгоритма стохастического градиентного спуска. То есть, на каждом шаге выбирается случайный элемент из обучающей выборки, считается градиент функции потерь (направление наискорейшего роста функции), затем обновляются веса в направлении антиградиента, чтобы минимизировать функцию потерь. Для данной модели необходимо было подобрать гиперпараметр α - скорость обучения (коэффициент, который при обучении определяет размер шага на каждой итерации в направлении антиградиента). Для этого был задан набор предполагаемых значений α и затем передан в функцию RandomizedSearchCV, с помощью которой после обучения и тестирования на каждом из значений с помощью перекрестной проверки было выбрано лучшее α =1e-05. Процесс подбора α можно наблюдать на графике (см. рис. 29).

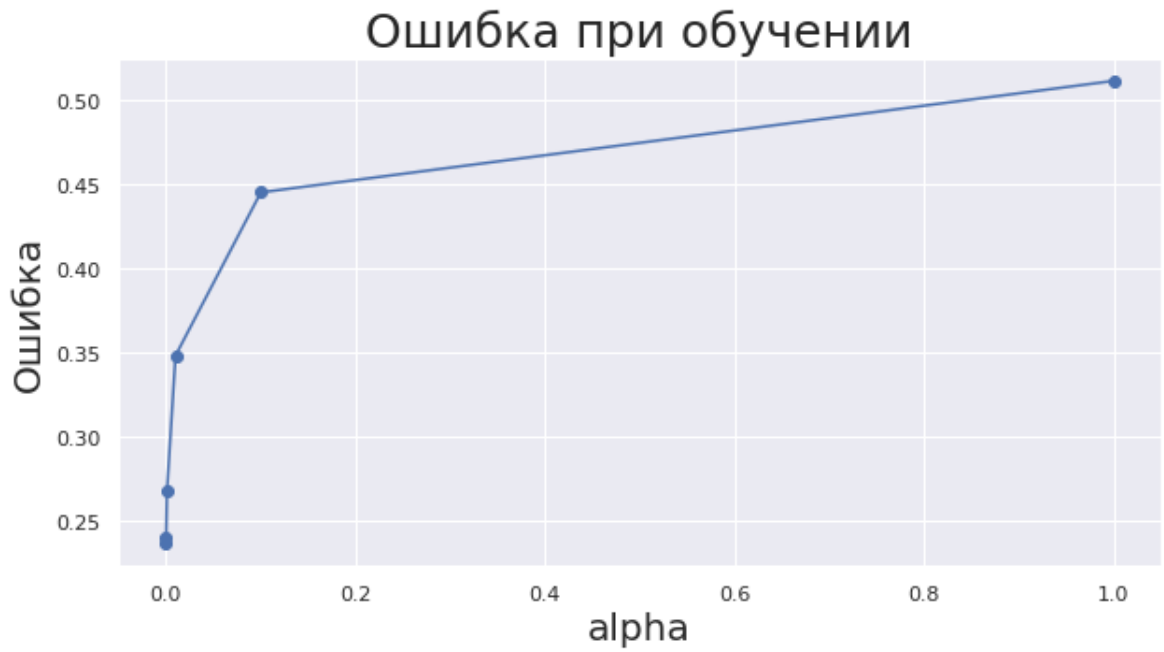

Рисунок 29 — Динамика изменения ошибки при подборе гиперпараметра α

После чего с использованием этого значения была обучена модель и проведена оценка качества на тестовом наборе = 0.48248.
Следующей моделью решения была использована модель Ridge из библиотеки scikit-learn (также известна как регуляризация Тихонова). Это регрессионная модель, которая в качестве функции потерь при обучении использует функцию наименьших квадратов вместе с L2-регуляризацией. То есть, другими словами, модель минимизирует следующую функцию ошибки:

$$E(y_i, \hat{y}_i) = \sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=0}^{k} w_j^2 =$$

$$= \sum_{i=1}^{N}(y_i - \sum_{j=0}^{k} x_j w_j)^2 + \lambda \sum_{j=0}^{k} w_j^2,$$

где слагаемое $\sum_{j=0}^{k} w_j^2$ - L2-регуляризация, N – число примеров в обучающей выборке, y_i – фактическое значение предсказываемой пере-

менной, $\hat{y}_i$ - предсказанное моделью значение, k – число признаков, x_i - признаки, w_i – обучаемые коэффициенты, λ - сила регуляризации. Регуляризация помогает уменьшить степень переобучения модели. Переобучение - это ситуация, при которой с повышением сложности модели ошибка на обучающей выборке продолжает снижаться, а ошибка на тестовой выборке увеличивается (см. рис. 30).

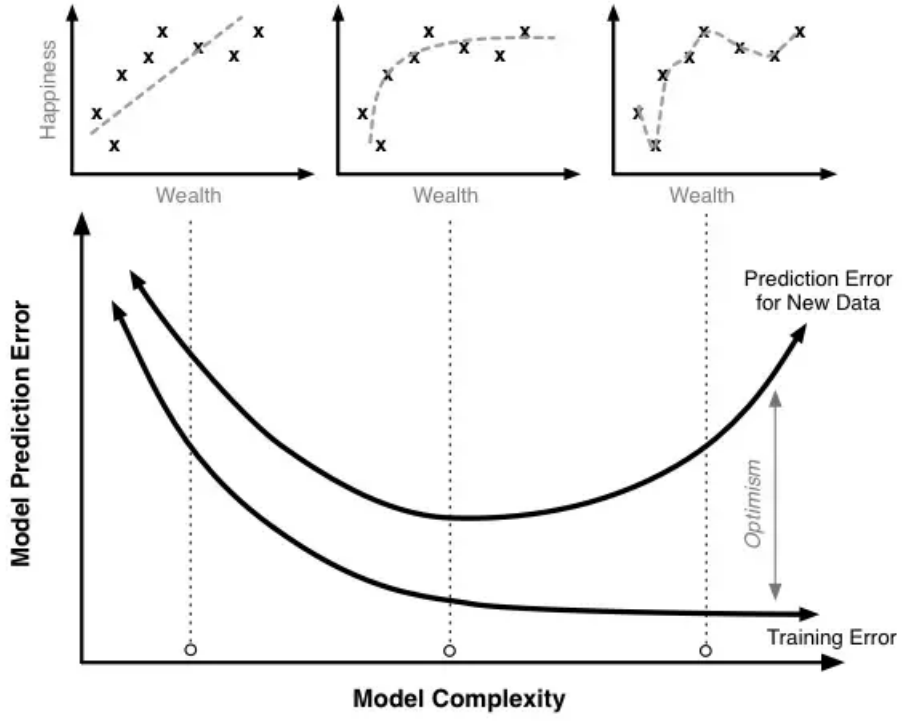

Рисунок 30 — Взаимосвязь сложности модели и ошибки предсказаний при недообучении и переобучении

То есть, при переобучении модель фактически «запоминает» данные из тренировочного набора, из чего следует плохая точность на новых данных из тестирующего набора, которые модель еще не видела. Чтобы значения коэффициентов модели не становились большими, при которых модель начинает запоминать данные, а не обобщать, регуляризация штрафует коэффициенты. L2-регуляризация штрафует коэффициенты модели, добавляя сумму их квадратов к ошибке. Настройка гиперпараметров для этой модели аналогична предыдущей. Передаём набор рассматриваемых значений α в функцию случайного

поиска, в результате чего функция выдаёт нам лучший вариант гиперпараметра α. В данном случае лучший $\alpha = 5$ (см. рис. 31).

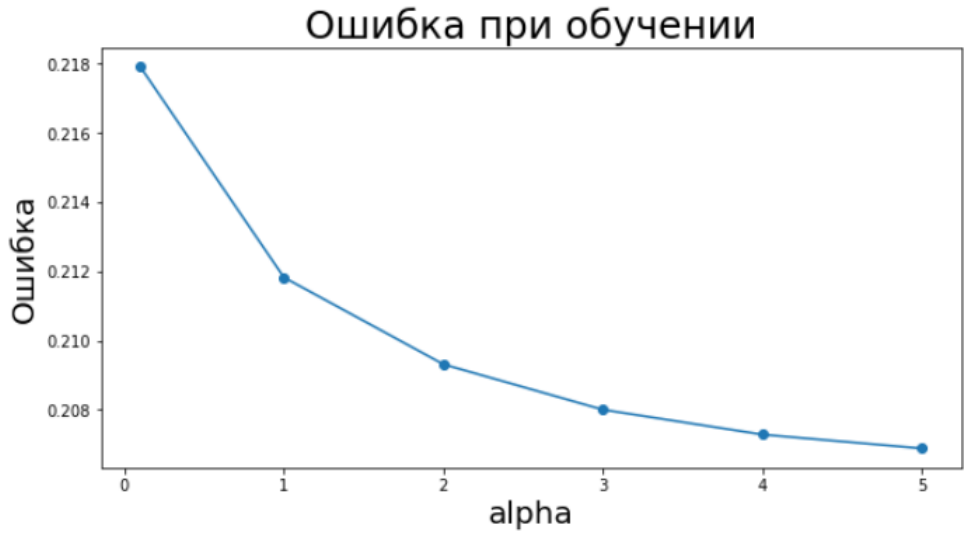

Рисунок 31 — Динамика изменения ошибки при подборе гиперпараметра α

С помощью данной модели была достигнута ошибка на тестовом наборе = 0.450486.

В качестве следующего решения была использована модель LightGBM из библиотеки lightgbm. LightGBM - это реализация алгоритма градиентного бустинга.

Градиентный бустинг - это метод машинного обучения, который создает модель прогнозирования в виде множества слабых предсказателей, обычно деревьев решений. Деревья решений - это техника разделения данных на основе признаков для прогнозирования некоторого значения. В каждой ветви в дереве решений данные разделяются на одну из двух групп. После чего каждому листу в дереве решений присваивается прогнозируемое значение (см. рис. 32).

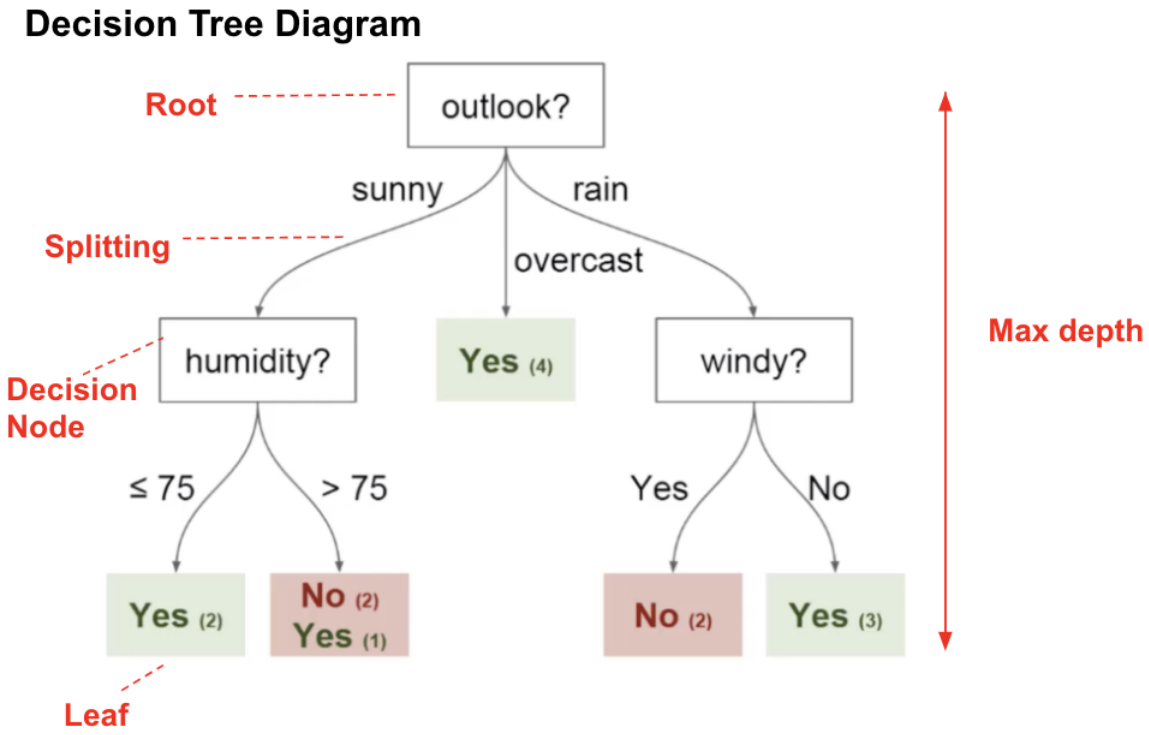

Рисунок 32 — Пример дерева решений

Однако одно дерево решений склонно к переобучению и поэтому не будет хорошо предсказывать на новых данных. Поэтому обычно вместо одного дерева решений используется множество деревьев. В основе градиентного бустинга лежит следующая идея: объединить предсказания нескольких деревьев решений, сложив их вместе. Например, в нашей задаче прогнозируемая цена для любого товара будет суммой предсказаний цен каждого отдельного дерева решений. Градиентный бустинг обучается итеративно, по одному дереву за итерацию. Дерево решений будет минимизировать функцию потерь с помощью рекурсивного разделения данных, пока не будет достигнут некоторый предел - например, глубина дерева или число листьев. Затем обученное дерево решений добавляется в модель, уже добавленные деревья в модели не изменяются. Для минимизации функции потерь при добавлении деревьев используется алгоритм градиентного спуска. Обучение прекращается, если в итоговую модель добавляется фиксированное количество деревьев или если потеря достигнет приемлемого

уровня. Для настройки данной модели необходимо подобрать следующие гиперпараметры: learning_rate - скорость обучения (определяет влияние каждого дерева на результат), n_estimators - количество деревьев, num_leaves - максимальное количество листьев в деревьях (используется, чтобы управлять сложностью модели и соответственно контролировать переобучение), max_depth - максимальная глубина деревьев в модели (используется, чтобы контролировать переобучение). Для каждого гиперпараметра было задано распределение с некотором диапазоне, из которых на каждой итерации случайного поиска RandomizedSearchCV выбиралась случайная комбинация параметров, которая тестировалась с помощью перекрёстной проверки, в результате чего была выбрана лучшая комбинация гиперпараметров. На данной комбинации было произведено обучение модели.
Достигнутая данной моделью ошибка на тестовом наборе = 0.44524.
Следующей моделью стало объединение моделей Ridge и LightGBM. К данным были добавлены прогнозы Ridge и с помощью полученного датасета была обучена модель LightGBM. В результате была немного улучшена оценка качества, которая составила 0.43601.
В качестве последней модели был составлен взвешенный ансамбль из первых трёх моделей. Ансамблем называется алгоритм, который состоит из нескольких разных алгоритмов машинного обучения. Взвешенный ансамбль - это ансамбль, в котором вклад каждого члена в окончательный прогноз взвешивается некоторым весом. Эти веса представляют собой небольшие положительные значения, а сумма всех весов равна единице, что позволяет весами указывать процент доверия для каждой модели:

$$F(x) = \sum_{i=1}^{N} \alpha_i f_i(x),$$

$$\sum_{i=1}^{N} \alpha_i = 1, \alpha_i > 0,$$

где F - ансамбль, x - часть данных, N - количество алгоритмов (членов ансамбля), f_i - члены ансамбля, α_i - веса членов ансамбля [9].
Веса были подобраны таким образом, что больший процент доверия (больший вес) получили модели с более высокой точностью прогнозирования. В результате тестирования данного ансамбля была получена самая высокая оценка из всех моделей = 0.43145. Однако, стоит отметить, что так как ансамбль требует участия трёх разных моделей для совершения прогноза, соответственно и обучается он дольше всего, с суммарным временем обучения трёх моделей-членов ансамбля.

6. Сравнение моделей

Таблица 1 — Сравнение реализованных моделей

Модель	Подбор гиперпараметров	Обучение	Время работы	RMSLE на обучающей выборке	RMSLE на тесте
Baseline	-	-	-	-	0.74950
SGDRegressor	2 мин	14.8 сек	0.05596 сек	0.47923	0.48248
Ridge	41.2 мин	2.05 мин	0.05898 сек	0.42168	0.45048
LightGBM	277.6 мин	24.1 мин	21.9 сек	0.39403	0.44524
Ridge+LightGBM	277.6 мин	24.6 мин	22.6 сек	0.37899	0.43601
Взвешенный ансамбль моделей	320.8 мин	26.2 мин	22.1 сек	-	0.43145

Как можем заметить, самая быстро обучаемая модель показала худший результат предсказания. Самая долгая в обучении модель же напротив показала себя лучше всех в обобщении данных, что является предсказуемым результатом.

Заключение

В процессе работы было проведено исследование данных, с помощью которого были выявлены некоторые дополнительные зависимости в данных и добавлены новые признаки, произведена очистка и предобработка данных. В результате работы с помощью библиотек машинного обучения были реализованы несколько моделей решений задачи определения цены товаров по их текстовому описанию. Также было проведено сравнение этих моделей между собой по различным характеристикам.

Список литературы

1. Mercari price suggestion challenge. — 2018. — URL: https://www.kaggle.com/c/mercari-price-suggestion-challenge.
2. Understanding Boxplots. — 2018. — URL: https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51.
3. About Feature Scaling and Normalization – and the effect of standardization for machine learning algorithms. — URL: https://sebastianraschka.com/Articles/2014_about_feature_scaling.html.
4. Stemming. — 2019. — URL: https://en.wikipedia.org/wiki/Stemming.
5. TF-IDF. — URL: https://ru.wikipedia.org/wiki/TF-IDF.
6. All about the metric: RMSLE. — 2019. — URL: https://www.kaggle.com/c/ashrae-energy-prediction/discussion/113064.

7. How To Get Baseline Results And Why They Matter. — 2014. — URL: https://machinelearningmastery.com/how-to-get-baseline-results-and-why-they-matter/.
8. Библиотека scikit-learn для машинного обучения. — URL: https://scikit-learn.org/stable/.
9. Виды ансамблей. — URL: http://neerc.ifmo.ru/wiki/index.php?title=%D0%92%D0%B8%D0%B4%D1%8B_%D0%B0%D0%BD%D1%81%D0%B0%D0%BC%D0%B1%D0%BB%D0%B5%D0%B9.

A. Приложение

Ссылка на репозиторий: https://github.com/nisded/graduation_work