


| | |
|---|---|
|  | <p>Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)</p> |
|---|---|

ФАКУЛЬТЕТ _____ Информатика и системы управления _____

КАФЕДРА _____ Программное обеспечение ЭВМ и информационные технологии _____

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Студент _____ Левушкин Илья Кириллович _____
фамилия, имя, отчество

Группа _____ ИУ7-62 _____

Тип практики _____ Производственная _____

Название предприятия _____ ООО «Корел Рус» _____

| | | |
|---------|----------------------|----------------------|
| Студент | _____ | _____ Левушкин И.К. |
| | <i>подпись, дата</i> | <i>фамилия, и.о.</i> |

| | | |
|-----------------------|----------------------|-----------------------|
| Руководитель практики | _____ | _____ Толпинская Н.Б. |
| | <i>подпись, дата</i> | <i>фамилия, и.о.</i> |

| | | |
|---|----------------------|----------------------|
| Руководитель практики от предприятия | _____ | _____ Дяйкин А.Д. |
| | <i>подпись, дата</i> | <i>фамилия, и.о.</i> |

Оценка _____

2020 г.

Индивидуальное задание

Проанализировать существующие системы автоматического анализа обзоров, использующих механизмы AI и обработки больших данных. На указанном примере автоматизировать выгрузку ревью, идентифицировать ключевые категории оценки продукта пользователями (стабильность, энергопотребление, конкретная функциональность и т.д.), реализовать автоматизированный подсчет количества положительных и отрицательных отзывов по конкретным категориям, автоматизировать оценку трендов изменения оценки категорий по годам, кварталам или месяцам.

Содержание

| | |
|---|-----------|
| Введение | 4 |
| 1 Аналитический раздел | 5 |
| 1.1 Подходы автоматического анализа обзоров, использующих механизмы AI и обработки больших данных | 5 |
| 1.2 Идентификация ключевых категорий оценки продукта пользователями (Taxonomy) | 5 |
| 1.3 Подготовка данных для обучения модели (Feature Extraction) | 5 |
| Выводы | 6 |
| 2 Конструкторский раздел | 7 |
| 2.1 Сбор данных | 7 |
| 2.2 Предварительная обработка | 8 |
| 2.3 Векторизация | 8 |
| 2.4 Обучение и тестирование модели | 8 |
| Выводы | 8 |
| 3 Технологический раздел | 9 |
| 3.1 Требования к программному обеспечению | 9 |
| 3.2 Средства реализации | 9 |
| 3.3 Листинг программы | 9 |
| Выводы | 11 |
| 4 Исследовательский раздел | 12 |
| 4.1 Тестирование на отзывах Parallels | 12 |
| 4.2 Тестирование на отзывах Excel | 12 |
| Выводы | 13 |
| Заключение | 14 |
| Список литературы | 15 |

Введение

На сегодняшний день магазины приложений, такие как Google Play или Apple Store, различные Web-сайты позволяют пользователям оставлять отзывы об используемых сервисах, публикуя комментарии и выставяя оценки. Эти платформы представляют собой полезный электронный ресурс, в котором разработчики приложений и пользователи могут продуктивно обмениваться информацией о сервисах.

В частности, отзывы пользователей могут содержать опыт использования приложений, сообщения об ошибках и предложения для улучшения сервиса. Вся эта информация может помочь разработчикам приложений выполнять задачи по сопровождению, развитию и модификации программного обеспечения. Однако большой объем полученных отзывов, их неструктурированный характер и различное качество могут сделать выявление полезных отзывов пользователей очень сложной задачей, на которую придется тратить много времени.

Таким образом, возникает потребность в автоматизации этого процесса, используя механизмы AI и обработки больших данных.

Цель практики: проанализировать существующие системы автоматического анализа обзоров, использующих механизмы AI и обработки больших данных, и применить полученные знания на практике.

Задачи работы:

1. изучить существующие подходы автоматического анализа обзоров, использующих механизмы AI и обработки больших данных;
2. выбрать наиболее подходящие из них для решения индивидуального задания;
3. автоматизировать выгрузку ревью продукта Parallels Desktop;
4. идентифицировать ключевые категории оценки продукта пользователями;
5. реализовать автоматизированный подсчет количества положительных и отрицательных отзывов по конкретным категориям;
6. автоматизировать оценку трендов изменения оценки категорий по годам, кварталам или месяцам.

1 Аналитический раздел

В данном разделе будут описаны существующие подходы автоматического анализа обзоров, использующих механизмы AI и обработки больших данных; идентификация ключевых категорий оценки продукта пользователями; подготовка данных для обучения модели и метод, использующийся для ее обучения.

Оценка трендов изменения оценки категорий по годам, кварталам или месяцам в данной работе не производилась, поскольку данный материал слишком обширный, и на его изучение требуется больше времени, чем отводится в рамках учебной практики.

1.1 Подходы автоматического анализа обзоров, использующих механизмы AI и обработки больших данных

Процесс анализа, классификации и выбора отзывов пользователей, полезных для разработчиков, состоит из 4 шагов [1]:

- Таксономия обслуживания и развития программного обеспечения (Taxonomy for Software Maintenance and Evolution).
Она подразумевает анализ отзывов конкретного программного обеспечения и выделения ключевых категорий, наиболее полезных для разработчиков.
- Извлечение признаков (Feature Extraction).
Целью данного шага является извлечение набора наиболее значащих признаков из отзывов пользователей, которые потом будут использованы алгоритмами машинного обучения.
- Обучение модели на основе выделенных признаков (Learning Classifiers).
- Оценка полученных результатов (Evaluation).

Рассмотрим подробнее каждый из них.

1.2 Идентификация ключевых категорий оценки продукта пользователями (Taxonomy)

Цель этого первого шага состоит в том, чтобы вывести таксономию категорий отзывов пользователей, которая имеет отношение к обслуживанию и развитию программного обеспечения.

Чтобы достичь поставленной цели необходимо проанализировать отзывы пользователей и ответы разработчиков на них. Анализ выполняется вручную.

1.3 Подготовка данных для обучения модели (Feature Extraction)

Существуют три основных этапа, применяющихся для анализа содержания отзывов приложений и извлечения наиболее важных признаков (features) для следующего шага (Learning Classifiers):

- Text Analysis (TA);

Этот этап включает в себя выбор метода, подходящего для извлечения наиболее важной информации из текста (textual features). Он состоит из двух этапов:

- Предварительная обработка текста (Preprocessing).

Все слова, содержащиеся в нашем наборе отзывов пользователей, используются в качестве информационной базы для создания текстового вокабуляра, который подвергается предварительной обработке с применением удаления стоп-слов (предлогов, союзов, знаков и тд) и lemmatization (преобразует слова в начальную форму).

- Векторизация текста (Textual Feature Weighting).

Существует множество способов векторизовать текст, но наиболее популярные, зарекомендовавшие себя, способы - это TF-IDF, прямое и частотное кодирование. В данной работе используется частотное кодирование, суть которого заключается в том, чтобы представить каждый отзыв в виде вектора, элементы которого являются числом вхождения каждого слова из вокабуляра, полученного на предыдущем этапе.

- Natural Language Processing (NLP);

Этот этап включает в себя в нахождении рекуррентных лингвистических шаблонов среди отзывов, которые можно будет использовать для распознавания предложений, относящихся к той или иной категории (Раздел 1.2 Тахоному).

Поскольку задача подразумевает просмотр и глубокий ручной анализ большого количества отзывов для нахождения подобных шаблонов (рис.), а времени на ее реализацию не достаточно в рамках учебной практики, было решено классифицировать все отзывы вручную, не прибегая к поиску шаблонов.

- Sentiment Analysis (SA).

Анализ настроений - это процесс присвоения количественного значения фрагменту текста, выражающему аффект или настроение. В данной работе рассматривается анализ настроений как задача классификации текста, которая присваивает каждый отзыв одному соответствующему классу.

Для решения этой задачи классы определяются как три различных уровня интенсивности настроения: положительный, отрицательный и нейтральный.

В качестве модели для прогнозирования настроений отзывов пользователей было решено использовать модель дискретного Наивного Байесовского классификатора, поскольку в источнике [1] говорится, что наивный байесовский метод показывает лучшие результаты, чем другие алгоритмы машинного обучения, традиционно используемые для классификации текста при анализе настроений.

Выводы

- Были изучены подходы, использующиеся для автоматического анализа отзывов, использующих механизмы AI и обработки больших данных; и выделены их основные этапы: Taxonomy, Feature Extraction, Learning Classifiers, Evaluation.
- Было решено выполнять таксономию вручную.
- Были выделены и подробно разобраны основные этапы извлечения признаков (Feature Extraction). Для каждого этапа был выбран метод, позволяющий решить поставленную задачу.

- Также, было решено не выполнять оценку трендов изменения оценки категорий по годам, кварталам или месяцам вследствие нехватки времени на поставленную задачу.

2 Конструкторский раздел

В этом разделе приводится описание сбора данных (отзывов), предварительной обработки и векторизации, а также особенности обучения и тестирования модели на собранных данных.

2.1 Сбор данных

Отзывы были предоставлены ментором данного задания из источника [12] - всего 284 шт.

Очевидно, что такого количества недостаточно, чтобы обучить и протестировать модель, поэтому было принято решение добавить отзывы из магазина приложений App Store [4] с помощью ресурса Appfigures [5], предоставляющего API к App Store.

Поскольку большинство отзывов написано на английском языке, оставшиеся отзывы были переведены на английский язык. Итого получилось 1023 отзыва, что безусловно мало для полноценного обучения и тестирования модели, но достаточно, чтоб получить хоть какие-то вменяемые результаты.

Каждому отзыву соответствует оценка по пятибальной шкале: от 1-5. Если оценка меньше 3, то отзыв будет считаться отрицательным (всего 525 шт.), если больше 3 (363), то положительным, и если равно 3, то нейтральным (всего 160 шт.).

Все отзывы были разбиты по соответствующим категориям:

- Лицензия, пробная версия, цена (390 шт.).

Пример отзыва: *Paid for 1 year for 3990 rubles, but the license was not activated !!! The support servant is silent, what should I do?*

- Функциональность (качество работы утилит, программ, настроек) (174 шт.).

Пример отзыва: *Does not have OpenGL support in the NVIDIA Geforce GT 750M drivers for Linux. So unfortunately the whole purpose of using Linux to avoid Mac depreciating OpenGL/OpenCL for 3D rendering is undermined by the lack of support in the driver too. From what I have read this is true for Windows graphics card drivers in Parallels as well.*

- Производительность и скорость работы (50 шт.).

Пример отзыва: *Downloading win system is too slow, 500m broadband, 9 hours to download 5g win10, drunk*

- Установка и работа системы, интеграция (251 шт.).

Пример отзыва: *I used the trial version to decide if Parallels was worth the investment. I was pleasantly surprised with the ease of installation and configuration. I installed both Windows 10 and Ubuntu 18.04 on my iMac-Pro, with Catalina. Both worked flawlessly.*

- Отзывы, не несущие полезную информацию для разработчика (158 шт.).

Пример отзыва: *very good*

2.2 Предварительная обработка

Предварительная обработка включает в себя токенизацию отзывов (разбиение текста на слова и биграммы), нормализацию получившихся слов и создание текстового вокабуляра слов.

Нормализация будет осуществляться посредством приведения слов к нижнему регистру, удалению стоп-слов (предлогов, союзов знаков, символов и тд) и лемматизации (приведению слов к начальной форме).

Вокабуляр будет содержать в себе список наиболее часто встречающихся нормализованных слов.

Итого, на выходе этого этапа получается список токенизированных и нормализованных отзывов и вокабуляр слов.

2.3 Векторизация

Как уже было сказано ранее, для векторизации текста в данной работе будет использоваться частотное кодирование.

То есть каждый отзыв будет представлять из себя вектор, элементы которого являются числом вхождения каждого слова из вокабуляра.

2.4 Обучение и тестирование модели

В качестве модели используется модель Наивного Байесовского классификатора (дискретного).

Байесовский классификатор имеет гиперпараметр *alpha*, который отвечает за сглаживание модели. Наивный Байес вычисляет вероятности принадлежности каждого отзыва ко всем классам, для этого перемножая условные вероятности появления всех слов отзыва, при условии принадлежности к тому или иному классу. Но если какое-то слово отзыва не встречалось в обучающем наборе данных, то его условная вероятность равна нулю, что обнуляет вероятности принадлежности отзыва к какому-либо классу. Чтобы избежать этого, по умолчанию ко всем условным вероятностям слов добавляется единица, то есть *alpha* равняется одному. Однако это значение может быть неоптимальным. Поэтому его значение будет определено в процессе выполнения эксперимента с помощью поиска по сетке и кросс валидации.

Данные будут разбиты на две части - обучающую и тестирующую, в соотношении 3:1, соответственно.

Выводы

В разделе были даны описания сбора данных (отзывов), предварительной обработки и векторизации, а также особенности обучения и тестирования модели на собранных данных.

3 Технологический раздел

Здесь описываются требования к программному обеспечению и средства реализации, приводятся листинги программы.

3.1 Требования к программному обеспечению

Необходимо подготовить программный комплекс, который будет считывать отзывы из файла формата csv, преобразовывать их в список и далее обрабатывать в соответствии со схемой изложенной в Конструкторском разделе.

На выходе получается обученная модель, готовая для тестирования.

3.2 Средства реализации

Для реализации поставленной задачи был использован язык программирования Python [7]. Проект был выполнен в среде The Jupyter Notebook [11].

Используемые библиотеки: nltk [6], scipy и numpy [9], pandas [10].

3.3 Листинг программы

Реализованный программный комплекс представлен в листингах 1-3.

Листинг 1: Предварительная обработка

```
1  import nltk
2  from nltk.corpus import PlaintextCorpusReader
3  from nltk.probability import FreqDist
4  from nltk.tokenize import RegexpTokenizer
5  from nltk import bigrams
6  from nltk import pos_tag
7  from nltk.stem import WordNetLemmatizer
8  from collections import OrderedDict
9  from sklearn.metrics import classification_report, accuracy_score
10 from sklearn.naive_bayes import MultinomialNB
11 from sklearn.model_selection import GridSearchCV
12 from sklearn.utils import shuffle
13 from multiprocessing import Pool
14 import numpy as np
15 from scipy.sparse import csr_matrix
16 import pandas as pd
17
18 def lower_pos_tag(words):
19     lower_words = []
20     for i in words:
21         lower_words.append(i.lower())
22     pos_words = pos_tag(lower_words)
23     return pos_words
24
25 def clean(words):
26     wordnet_lemmatizer = WordNetLemmatizer()
27     cleaned_words = []
28     types = ['JJ', 'JJR', 'JJS',
```

```

29         'MD', 'NN', 'NNS',
30         'NP', 'NPS', 'POS',
31         'PP', 'RB', 'RBR',
32         'RBS', 'VV', 'VVD',
33         'VG', 'VN', 'VVP', 'VZ']
34     for i in words:
35         if i[1] in types:
36             cleaned_words.append(wordnet_lemmatizer.lemmatize(i[0]))
37     return cleaned_words
38
39 def get_review_by_status(data_path, status):
40     frame = pd.read_csv(data_path, header=0, sep=',')
41     return frame[frame['status'].isin([status])].review.to_list()
42
43 def get_review_by_label(data_path, label):
44     if (label == 'neutral'):
45         return get_review_by_status(data_path, 0)
46     elif (label == 'bad'):
47         return get_review_by_status(data_path, -1)
48     elif (label == 'good'):
49         return get_review_by_status(data_path, 1)
50     else:
51         return None
52
53 def process(data_path, label):
54     # Wordmatrix – list of reviews with tokens
55     # All words – list of all words
56     data = {'Word_matrix': [], 'All_words': []}
57     # Intermediate list for removing gapaxes
58     templist_allwords = []
59     reviews = get_review_by_label(data_path, label)
60     # Creating a tokenizer
61     tokenizer = RegexpTokenizer(r'\w+|(^\\w\\s)+')
62     for review in reviews: # Case processing
63         bag_words = tokenizer.tokenize(review)
64         lower_words = lower_pos_tag(bag_words)
65         cleaned_words = clean(lower_words)
66         finalist = list(bigrams(cleaned_words)) + cleaned_words
67         data['Word_matrix'].append(finalist)
68         templist_allwords.extend(cleaned_words)
69     # Definition of hapaxes
70     templistfreq = FreqDist(templist_allwords)
71     hapaxes = templistfreq.hapaxes()
72     # Filtration from gapaxes
73     for word in templist_allwords:
74         if word not in hapaxes:
75             data['All_words'].append(word)
76     return {label: data}

```

Листинг 2: Векторизация

```

1 # Create tagged data with structure:
2 # [(list of review words), class_label])
3 def union_data(data):
4     labels = ['neutral', 'bad', 'good']
5     labeled_data = []

```

```

6     for label in labels:
7         for document in data[label]['Word_matrix']:
8             labeled_data.append((document, label))
9     return labeled_data
10
11 def vocabular_create(labels, data):
12     # Creating a vocabulary with unique tokens
13     all_words = []
14     for label in labels:
15         frequency = FreqDist(data[label]['All_words'])
16         common_words = frequency.most_common(1000)
17         words = [i[0] for i in common_words]
18         all_words.extend(words)
19     # Extraction of unique tokens
20     unique_words = list(OrderedDict.fromkeys(all_words))
21     return unique_words
22
23 def frequency_coding(labeled_data, unique_words):
24     # Frequency coding for classifiers scikit-learn
25     # Sparse matrix for features
26     matrix_vec = csr_matrix((len(labeled_data), len(unique_words)), dtype=np
27                             .int8).toarray()
28     # Array for class labels
29     target = np.zeros(len(labeled_data), 'str')
30     for index_doc, document in enumerate(labeled_data):
31         for index_word, word in enumerate(unique_words):
32             # Counting the number of occurrences of a word in the review
33             matrix_vec[index_doc, index_word] = document[0].count(word)
34             target[index_doc] = document[1]
35     return matrix_vec, target

```

Листинг 3: Обучение модели

```

1 # Reading, processing and vectorization of training data
2 teaching_data = {}
3 labels = ['neutral', 'bad', 'good']
4 for label in labels:
5     teaching_data.update(process('teaching_sentiments.csv', label))
6
7 unique_words = vocabular_create(labels, teaching_data)
8
9 matrix_vec, target = frequency_coding(union_data(teaching_data),
10                                         unique_words)
11
12 # Shuffling the dataset
13 X, Y = shuffle(matrix_vec, target)
14
15 model = MultinomialNB(0.1)
16 model.fit(X, Y)

```

Выводы

В данном разделе были рассмотрены требования к программному обеспечению, обоснован выбор средств реализации, приведены листинги программы.

4 Исследовательский раздел

В этом разделе представлены результаты работы программного комплекса на тестирующих выборках.

Тестовые данные обрабатывались также, как и обучающая выборка, однако для векторизации использовался вокабуляр обучающего датасета.

Листинг 4: Тестирование модели

```
1 predicted = model.predict(X_test)
2 # Accuracy on the control dataset
3 score_test = accuracy_score(Y_test, predicted)
4 # Classification report
5 report = classification_report(Y_test, predicted)
```

4.1 Тестирование на отзывах Parallels

Тестирование производилось на выборке объемом 281 отзывом ($\frac{1}{4}$ датасета).

На рисунке 1 представлена таблица результатов и оценка точности.

| | | | | | |
|---------------------|-----------|--------|----------|---------|--|
| 0.45907473309608543 | | | | | |
| | precision | recall | f1-score | support | |
| b | 0.39 | 0.90 | 0.54 | 81 | |
| g | 0.62 | 0.53 | 0.57 | 100 | |
| n | 0.38 | 0.03 | 0.06 | 100 | |
| accuracy | | | 0.46 | 281 | |
| macro avg | 0.46 | 0.49 | 0.39 | 281 | |
| weighted avg | 0.47 | 0.46 | 0.38 | 281 | |

Рис. 1: Результаты тестирования на данных Parallels

Где precision - доля правильно угаданных оценок (идет счет по *predicted*), recall - доля оценок, которые правильно угадали (идет счет по Y_{test}).

Видно, что точность оценки не сильно впечатляет (менее 50%). Хотя это и лучше простого угадывания (33%). Также, можно заметить, что модель хорошо справляется с плохими отзывами и неплохо с хорошими. Хуже всего получается предсказывать нейтральные отзывы. Можно заметить, что результаты пропорциональны количеству отзывов (плохих - 525, хороших - 363, нейтральных - 160), из чего можно сделать вывод, что модели не хватает данных, чтобы обучиться.

4.2 Тестирование на отзывах Excel

Тестирование производилось на выборке объемом 300 отзывов. Были взяты 100 положительных, 100 отрицательных и 100 нейтральных отзывов.

На рисунке 2 представлена таблица результатов и оценка точности.

Ровно такая же тенденция прослеживается на отзывах Excel, что подтверждает выводы, сделанные в предыдущем тестировании.

| | | | | |
|--------------------|-----------|--------|----------|---------|
| 0.4866666666666667 | | | | |
| | precision | recall | f1-score | support |
| b | 0.44 | 0.90 | 0.59 | 100 |
| g | 0.60 | 0.53 | 0.56 | 100 |
| n | 0.38 | 0.03 | 0.06 | 100 |
| accuracy | | | 0.49 | 300 |
| macro avg | 0.47 | 0.49 | 0.40 | 300 |
| weighted avg | 0.47 | 0.49 | 0.40 | 300 |

Рис. 2: Результаты тестирования на данных Excel

Выводы

Было проведено тестирование на 2 разных группах отзывов. Оба они показывают довольно плохие плохие результаты, что связано с недостаточным количеством обучающей выборки.

Заключение

В ходе работы выполнено следующее:

- 1) изучены существующие подходы автоматического анализа обзоров, использующих механизмы AI и обработки больших данных;
- 2) выбран наиболее подходящий из них для решения индивидуального задания;
- 3) произведена выгрузка ревью продукта Parallels Desktop с сервиса Trustpilot и App Store;
- 4) идентифицированы ключевые категории оценки продукта пользователями;
- 5) реализован автоматический подсчет количества положительных, отрицательных и нейтральных отзывов по конкретным категориям с точностью 45,91%;

В результате проведенных тестов выяснилось, что обученная модель имеет точность предсказания 45,91%. Такая низкая точность связана с нехваткой данных для обучения. Об этом свидетельствует прямая зависимость между распределением количества положительных, отрицательных и нейтральных отзывов в датасете и долей оценок, которые правильно угадали по соответствующему классу.

Таким образом, текущую модель можно улучшить, подготовив большее количество данных (порядка 3 тыс.), примерно до 62% точности [2]. Это почти в два раза лучше простого угадывания, но точность все ещё довольно низка.

Также, можно было бы свести задачу до бинарной классификации отзывов на положительные и отрицательные, что также повысило бы точность.

Список литературы

- [1] How can I improve my app? Classifying user reviews for software maintenance and evolution [Электронный ресурс]. – Режим доступа: <https://www.zora.uzh.ch/id/eprint/113425/1/C17.pdf>, свободный. (Дата обращения: 18.07.2020 г.)
- [2] Анализ эмоциональной окраски отзывов с Кинопоиска [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/467081/>, свободный. (Дата обращения: 18.07.2020 г.)
- [3] Stemming and Lemmatization with Python NLTK [Электронный ресурс]. – Режим доступа: <https://www.guru99.com/stemming-lemmatization-python-nltk.html>, свободный. (Дата обращения: 18.07.2020 г.)
- [4] Mac App Store Preview; Parallels Desktop; Ratings and Reviews [Электронный ресурс]. – Режим доступа: <https://apps.apple.com/app/parallels-desktop/id1085114709#see-all/reviews>, свободный. (Дата обращения: 18.07.2020 г.)
- [5] Appfigures [Электронный ресурс]. – Режим доступа: <https://appfigures.com/reports/reviews>, по подписке. (Дата обращения: 18.07.2020 г.)
- [6] Natural Language Toolkit [Электронный ресурс]. – Режим доступа: <https://www.nltk.org/>, свободный. (Дата обращения: 18.07.2020 г.)
- [7] Python 3.8.5 documentation [Электронный ресурс]. – Режим доступа: <https://docs.python.org/3/index.html>, свободный. (Дата обращения: 18.07.2020 г.)
- [8] Anaconda Documentation [Электронный ресурс]. – Режим доступа: <https://docs.anaconda.com/>, свободный. (Дата обращения: 18.07.2020 г.)
- [9] Numpy and Scipy Documentation [Электронный ресурс]. – Режим доступа: <https://docs.scipy.org/doc/>, свободный. (Дата обращения: 18.07.2020 г.)
- [10] Pandas Documentation [Электронный ресурс]. – Режим доступа: <https://pandas.pydata.org/pandas-docs/stable/index.html>, свободный. (Дата обращения: 18.07.2020 г.)
- [11] Jupyter Documentation [Электронный ресурс]. – Режим доступа: <https://jupyter.org/documentation>, свободный. (Дата обращения: 18.07.2020 г.)
- [12] Parallels Reviews [Электронный ресурс]. – Режим доступа: <https://www.trustpilot.com/review/parallels.com?languages=en>, свободный. (Дата обращения: 18.07.2020 г.)