МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра теоретических основ компьютерной безопасности и криптографии

АНАЛИЗ ТОНАЛЬНОСТИ ОТЗЫВОВ О ФИЛЬМАХ С ПОМОЩЬЮ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ

КУРСОВАЯ РАБОТА

студента 4 курса 431 группы
направления 100501 — Компьютерная безопасность
факультета КНиИТ
Улитина Ивана Владимировича

Научный руководитель
доцент _____ Слеповичев И. И.

Заведующий кафедрой
_____ Абросимов М. Б.

Саратов 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В течении последних нескольких лет одним из актуальных направлений искусственного интеллекта является обработка и генерация текста. Технологии в рамках этой сферы машинного и глубокого обучения постоянно развиваются и незамедлительно находят применение в человеческом обиходе. Примерами применения результатов изучения алгоритмов в этой области являются суммаризаторы и классификаторы текста, различные голосовые помощники или чат-боты с искусственным интеллектом. Последние, в свою очередь, получили широкое распространение из-за удобства их использования при решении самых разных задач — от генерации ими рецептов различных блюд или анекдотов, до генерации рабочего и компилирующегося кода, который выполняет некоторую описанную пользователем функцию.

Одной из классических задач этой области искусственного интеллекта считается анализ тональности, суть которого состоит в том, чтобы при некотором входном тексте сделать вывод о том, какой эмоциональный окрас имеет этот текст (например, анализ текста комментария пользователя на сайте для просмотра фильмов, отражающий впечатления человека относительно просмотренного кино). Такая задача распространена и её решение может входить в основу различных рекомендательных систем, статистических сводок относительно продаваемой продукции или других аспектов маркетинга.

Целью данной курсовой работы является построение системы анализа тональности отзывов о фильмах на английском языке. В рамках теоретической части данной курсовой работы будут рассматриваться алгоритмы машинного обучения, применяемые при решении поставленной задачи, а также способы обработки используемого набора данных и методы оценки качества системы. На основе проделанной работы будут сформулированы выводы о различных способах решения проблемы.

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Искусственный интеллект (англ. Artificial Intelligence) — технология создания алгоритмов, лежащих в основе проектирования интеллектуальных машин и программ, способных имитировать деятельность человека.

Нейронная сеть (нейросеть) (англ. Neural Network) — математическая модель, чаще всего имеющая программную интерпретацию, сутью которой является реализация деятельности, похожей на деятельность биологических нейронных сетей. Нейронная сеть используется при создании какого-либо из алгоритмов искусственного интеллекта и состоит из совокупности нейронов, соединенных между собой связями.

Обработка текста на естественном языке (англ. Natural Language Processing, NLP) — одно из направлений развития машинного обучения, искусственного интеллекта и науки о данных, а также математической лингвистики. В данной области знаний рассматриваются проблемы компьютерного анализа и преобразования текстов на языках, используемых людьми для общения.

Набор данных, выборка (англ. dataset) — совокупность элементов, которая охватывается экспериментом (наблюдением, опросом). В частности, этот эксперимент представляет собой обучение нейронной сети или алгоритма машинного обучения с целью решения задачи классификации.

Элемент выборки (англ. sample) — один объект из набора данных, который характеризуется некоторыми (возможно, уникальными) свойствами.

Признак (англ. feature) — одно из свойств или качеств элемента выборки, значение которого будет использоваться нейронной сетью или алгоритмом машинного обучения.

Целевой признак — признак, значение которого необходимо предсказать с помощью нейронной сети или алгоритма машинного обучения, и для качественного предсказания которого осуществляется обучение модели.

1 Теоретическая часть

Тональность некоторого текста — это эмоциональное отношение/окрас автора текста относительно некоторого объекта, о котором идет речь в высказывании. Это может быть объект реального мира, событие, процесс или свойства чего-либо.

Анализ тональности текста заключается в процессе автоматического определения и классификации тональности (положительной, отрицательной или в некоторых случаях нейтральной) текстового документа с использованием методов и алгоритмов обработки естественного языка.

Один из самых распространенных подходов к анализу тональности текста — это использование машинного обучения и алгоритмов классификации.

Этот процесс состоит из следующих шагов:

1. Первичная подготовка данных (предобработка)
2. Обучение алгоритма машинного обучения
3. Тестирование и оценка качества обучения

1.1 Способы предобработки текста

Важным этапом решения задачи применения алгоритмов машинного обучения является первичная обработка (предобработка) данных, которые в дальнейшем будут использоваться алгоритмами для обучения и на основе содержимого которых будут функционировать построенные предиктивные системы. Чем более качественная выборка используется для задачи, тем лучше будут результаты работы таких алгоритмов.

В зависимости от типа данных для обучения (изображения, текст, числовые значения), используются соответствующие методы обработки выборки. Описанные ниже способы предобработки будут применяться в дальнейшем при написании практической части курсовой работы.

1.1.1 Мешок слов

Проблема текстов заключается в том, что они беспорядочны и могут иметь разную длину, а большинство алгоритмов машинного обучения предполагают входные и выходные параметры фиксированной длины.

Исходя из этого, алгоритмы машинного обучения не могут работать напрямую с необработанным текстом: его необходимо преобразовать в последовательности чисел (векторы). При языковой обработке векторы формируются из

текстовых данных, отражая лингвистические и статистические свойства текста. Это называется извлечением или кодированием признаков.

Мешок слов (англ. Bag-of-Words, BoW) — один из таких методов обработки. Это представление текста в виде мультимножества без учета его грамматических особенностей и порядка слов, которое описывает информацию об их количестве в этом тексте. Подход очень прост и гибок, его можно использовать множеством способов для извлечения характеристик текста. В частности, практическая часть в общем виде подразумевает следующий порядок действий:

1. Удаление из текста знаков пунктуации, спецсимволов (различных скобок и т.п.).
2. Перевод текста в нижний регистр (в силу отсутствия необходимости знания о порядке слов).
3. Преобразование каждого текста в список из слов.
4. Создание словаря – списка уникальных слов, присутствующих во всех кодируемых текстах, где каждому слову будет соответствовать некоторое число.
5. Замена списка слов на векторы, состоящие из чисел, которые этим словам соответствуют (токенизация).
6. Формирование на основе векторов матрицы-результата, в которой для каждого слова (столбца) и каждого текста (строки) соответствует количество использования этого слова в этом тексте.

Это называется "мешком" слов, потому что всякая информация о порядке или структуре слов в документе отбрасывается. Полученная структура в первую очередь предназначена для хранения информации о частоте использования слова в тексте, а не об их порядке. [3]

1.1.2 Коллокации

Существует несколько способов улучшить применение мешка слов по отношению к тексту, и часть этих способов образуется путем удаления из текстов мало информативных конструкций. Помимо удаления пунктуации, сокращений, стоп-слов и замены больших букв, можно также использовать n-граммы.

Как уже ранее упоминалось, при токенизации одно слово заменяется на одно числовое значение. n-граммой в данном случае называется токен, определяющий совокупность из n слов. Таким образом, можно выделить биграммы, триграммы и т.д.

Используя n-граммы в качестве токенов, возникает проблема высокой размерности результирующей матрицы, так как все пары слов значительно увеличивают длину словаря. В качестве уменьшения размерности могут использоваться различные способы удаления не слишком информативных n-грамм (например, удаление биграмм, содержащих междометия, частицы, артикли).

Информативные n-граммы называются **коллокациями**. [4] Для того, чтобы в обрабатываемом тексте оставались исключительно коллокации, можно:

1. удалить часто встречающиеся n-граммы (те же артикли, которые в английском языке широко распространены в текстах). Чем чаще встречается некоторая n-грамма, тем меньше конкретной информации оно содержит и, как следствие, будет слабо охарактеризовывать некоторый текст;
2. удалить слишком редко встречающиеся n-граммы (это могут быть опечатки, слишком специфичная лексика);

1.1.3 Частота слова и обратная частота документа

С помощью частоты слова (англ. term frequency, TF) можно оценить то, насколько часто встречаются токены/слова в конкретном документе. Значимость некоторого слова пропорциональна частоте использования этого слова в тексте и обратно пропорциональна частоте использования слова во всех текста выборки. Частоту слова можно определить следующей формулой:

$$tf(t,d) = \frac{n_t}{\sum_k n_k}, \tag{1}$$

где n_t — число вхождений слова t в текст, а $\sum_k n_k$ — общее число слов в данном тексте.

Обратная частота документа (англ. inverse document frequency, IDF) — это инверсия частоты слова, встречающегося во всех документах выборки. Она определяет, насколько часто слова появляются во всех остальных документах. Для каждого уникального токена/слова в пределах одной выборки документов существует только одно значение idf:

$$idf(t,D) = \log \frac{|D|}{|\{d_i \in D | t \in d_i\}|}, \tag{2}$$

где $|D|$ — число документов в выборке, $|\{d_i \in D | t \in d_i\}|$ — число документов

из выборки D, в которых встречается слово t (при $n_t \neq 0$).

Значение основания логарифма в формуле не имеет существенной важности в силу того, что его изменение может привести только к изменению веса каждого токена/слова на постоянный множитель, но это не влияет на соотношение весов между собой.

С помощью этих двух статистических мер может быть сформирована ещё одна мера ($TFIDF$), которая выглядит следующим образом [6]:

$$TFIDF(t, d, D) = tf(t, d) \cdot idf(t, D) \tag{3}$$

Большую значимость в $TFIDF$ получат токены/слова с высокой частотой в рамках конкретного текста и с низкой частотой упоминаний в других документах. [5]

1.1.4 Стемминг и лемматизация

Стеммингом (англ. stemming) называется преобразование слова, после которого от него остается только корень. Это своего рода нормализация слов. Стемминг важен тогда, когда при формировании мешка слов обнаруживается большое количество однокоренных слов. Например, слова ”wait”, ”waiting”, ”waited”, ”waits” имеют схожую смысловую нагрузку, однако в мешке слов будут представлять собой разные сущности, что будет способствовать ухудшению работы алгоритма машинного обучения. Проще вместо четырех однокоренных слов оставить один термин — ”wait”, на которое и будет заменены все вариации этого слова.

Лемматизация – это процесс определения леммы слова исходя из его значения. Лемматизацию относят к морфологическому анализу слов, задачей которого является удаление флективных окончаний (тех, что соотносятся по значению с корнем). Это способствует преобразованию слова в свою базовую или словарную форму, которое также называется леммой.

Применение стемминга и лемматизации к тексту способствует сокращению размера формирующегося мешка слов. Это вызвано тем, что приведение различных форм слова к одной единственной, а также удаление флективных окончаний уменьшает количество разнообразных слов в наборе данных, что приводит к повышению качества работы алгоритмов машинного обучения за счет однозначности кодирования разных форм слов, имеющих один и тот же

смысл. [4]

1.2 Алгоритмы машинного обучения

1.2.1 Полиномиальный наивный байесовский классификатор

Полиномиальный наивный байесовский алгоритм – одна из разновидностей наивного байесовского алгоритма в машинном обучении, который очень полезен для использования в наборе данных, который распределяется полиномиально. При решении задачи мультиклассовой классификации может использоваться именно этот алгоритм, в силу того, что для прогнозирования метки текста он вычисляет вероятность каждой метки для входного текста, после чего генерирует метку с наибольшей вероятностью в качестве выходных данных. [7]

Для полиномиальной классификации выделяют следующие преимущества этого алгоритма:

1. Удобство применения на непрерывных и дискретных данных.
2. Может обрабатывать большие наборы данных.
3. Возможность классификации данных по нескольким меткам.
4. Хорошо применимо для обучения моделей обработки естественного языка.

Класс MultinomialNB из библиотеки scikit-learn для Python реализует наивный байесовский алгоритм для полиномиально распределенных данных и является одним из двух классических наивных байесовских вариантов, используемых в классификации текста (где данные обычно представлены как счетчики векторов слов, составленных с помощью мешка слов, хотя векторы $TFIDF$ также хорошо работают на практике). Распределение параметризуется векторами $\theta_y = (\theta_{y_1}, \ldots, \theta_{y_n})$ для каждого класса y, где n — количество признаков (так как решается задача классификации текста — в данном случае это размер словарного запаса) и θ_{y_i} это вероятность $P(x_i|y)$ признака i войти в элемент выборки, принадлежащий к классу y.

Параметры θ_y оцениваются сглаженной версией максимального правдоподобия, то есть подсчетом относительной частоты:

$$\hat{\theta} y_i = \frac{N_{y_i} + \alpha}{N_y + \alpha n}, \tag{4}$$

где $N_{y_i} = \sum_{x \in T} x_i$ это количество появления признака i в объекте класса y в

обучающем наборе T, и $N_y = \sum_{i=1}^{n} N_{y_i}$ это общее число всех признаков для класса y.

Сглаживающие приоры $\alpha \geq 0$ учитывают признаки, отсутствующие в обучающей выборке, и предотвращают нулевые вероятности в дальнейших вычислениях. Установка парамера $\alpha = 1$ называется сглаживанием Лапласа, а $\alpha < 1$ называется сглаживанием Лидстоуна. [14]

1.2.2 Метод опорных векторов

Метод опорных векторов (англ. Support Vector Machines, SVM) — это набор методов обучения с учителем, используемых для классификации, регрессии и обнаружения выбросов.

Главный принцип метода состоит в представлении векторов пространства признаков, представляющих классифицируемые объекты, в пространство большей размерности. Это основано на том факте, что в пространстве более высокой размерности линейная разделимость множества выше, чем в пространстве с меньшей размерностью. Причина этого явления интуитивна: чем больше признаков используется для распознавания объектов, тем лучше ожидаемое качество распознавания.

После перевода в пространство большей размерности, в нём формируется разделяющая гиперплоскость. Вместе с этим все векторы, расположенные с одной ”стороны” гиперплоскости, относятся к одному некоторому классу, а расположенные с другой — ко второму (или любому другому). По обе стороны основной разделяющей гиперплоскости, параллельно ей и на равном расстоянии от неё формируются две вспомогательные гиперплоскости, расстояние между которыми называют зазором.

Целью является построение разделяющей гиперплоскости таким образом, чтобы увеличить зазор — область пространства признаков между вспомогательными гиперплоскостями, в которой вектора должны отсутствовать. Допускается, что разделяющая гиперплоскость, построенная по данному правилу, обеспечит наиболее четкое разделение классов и минимизирует среднюю ошибку распознавания.

Векторы, попадающие на границы зазора (т.е. те, что лежат на вспомогательных гиперплоскостях), называют опорными векторами (что и определяет название для метода). [10]

Выделяют следующие преимущества описанного алгоритма:

1. Эффективен при работе с пространствами больших размеров.
2. Эффективен в случаях, когда количество измерений превышает количество образцов.
3. Использует подмножество обучающих точек в функции принятия решений (называемых опорными векторами), поэтому это также эффективно с точки зрения памяти.
4. Универсальность: для функции принятия решения могут быть указаны различные функции ядра. Предоставляются общие ядра, но также можно указать собственные ядра.

Также у этого подхода выделяют следующие недостатки:

1. Если количество признаков намного превышает количество элементов выборки, необходимо избегать переобучения и более ответственно подходить к определению параметров регуляризации.
2. SVM не предоставляют оценки вероятностей напрямую, они могут рассчитываться с помощью ресурсозатратной пятикратной перекрестной проверки (англ. cross-validation).

1.2.3 Логистическая регрессия

При решении задачи классификации средствами машинного обучения могут упоминаются алгоритмы, которые в качестве своей основы имеют линейный классификатор — подход классификации, когда решение принимается на основании применения линейной комбинации над входными данными (признаками элементов выборки).

Логистическая регрессия является частным случаем линейного классификатора с важным свойством — способностью прогнозировать вероятность отношения объекта к некоторому классу A. [8]

Пусть $P(X)$ — вероятность происходящего события X. Тогда отношение вероятностей $OR(X)$ определяется из выражения $\frac{P(X)}{1-P(X)}$, а это — отношение вероятностей того, произойдет ли событие или не произойдет. Вероятность и отношение шансов содержат одинаковую информацию, но в то время как $P(X)$ находится в пределах от 0 до 1, $OR(X)$ находится в пределах от 0 до ∞. Пусть $w_0, w_1, w_2, \ldots$ — веса модели, $x_0, x_1, x_2, \ldots$ — признаки элемента выборки. Этапы вычисления логистической регрессией прогноза $p_A = P\left(y_i = 1 \mid \vec{x_i}, \vec{w}\right)$ можно определить следующим образом:

1. Вычислить значение $w_0 + w_1x_1 + w_2x_2 + ... = \vec{w}^T\vec{x}$. (уравнение $\vec{w}^T\vec{x} = 0$

задает гиперплоскость, разделяющую примеры на 2 класса);

2. Вычислить логарифм отношения шансов: $\log(OR_A) = \vec{w}^T\vec{x}$.
3. Имея прогноз шансов на отнесение к некоторому классу A – OR_A, вычислить p_A с помощью простой зависимости:

$$p_A = \frac{OR_A}{1 + OR_A} = \frac{\exp^{\vec{w}^T\vec{x}}}{1 + \exp^{\vec{w}^T\vec{x}}} = \frac{1}{1 + \exp^{-\vec{w}^T\vec{x}}} = \sigma(\vec{w}^T\vec{x}). \quad (5)$$

Таким образом, логистическая регрессия прогнозирует вероятность отнесения примера к некоторому классу (при условии, что известны признаки и веса модели) как сигмоид-преобразование линейной комбинации вектора весов модели и вектора признаков [9]:

$$p_A(x_i) = P\left(y_i = 1 \mid \vec{x_i}, \vec{w}\right) = \sigma(\vec{w}^T\vec{x_i}). \quad (6)$$

1.3 Метрики оценки качества обучения

Для оценки качества обучения нейронной сети или алгоритма машинного обучения используются специальные функции от выходных значений модели и эталонных значений выборки, называемые метриками. Выбор той или иной метрики напрямую определяется типом решаемой задачи, характеристиками набора данных (например, является ли выборка несбалансированной), какими-либо предпочтениями и т.д.

Важной концепцией в терминах вычисления значений ошибок классификации является матрица ошибок (англ. confusion matrix). Пусть имеется два класса и алгоритм, предсказывающий принадлежность каждого объекта одному из классов. Пусть $\hat{y}$ — это ответ алгоритма на объекте, y — истинная метка класса на этом объекте. Тогда матрица ошибок классификации будет определена следующим образом [11]:

	$y = 1$	$y = 0$
$\hat{y} = 1$	True Positive (TP)	False Positive (FP)
$\hat{y} = 0$	False Negative (FN)	True Negative (TN)

Рисунок 1 – Матрица ошибок

Таким образом, можно определить 4 значения этой матрицы: True Positive

(TP) или True Negative (TN) — количество объектов "класса А" или количество объектов не из "класса А", которые были правильно распознаны нейросетью. Величины False Positive (FP) и False Negative (FN) — количество объектов, которые были неправильно отнесены или не отнесены к "классу А", соответственно.

Одними из самых распространенных метрик обучения являются доля правильных ответов алгоритма (англ. Accuracy), точность (англ. Precision), полнота (англ. Recall) и F_β. [12]

Интуитивно понятной, очевидной и почти неиспользуемой метрикой является доля правильных ответов алгоритма:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{7}$$

Эта метрика бесполезна в задачах с неравными классами — при определении всех элементов выборки как объектов преобладающего класса значение метрики будет расти с увеличением разницы в количестве объектов между преобладающим и иными классами.

Для оценки качества работы алгоритма на каждом из классов по отдельности вводятся точность и полнота:

$$Precision = \frac{TP}{TP + FP}, \tag{8}$$

$$Recall = \frac{TP}{TP + FN}. \tag{9}$$

$Precision$ можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, а $Recall$ показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм.

F_β (или F-мера) является средним гармоническим между двумя предыдущими метриками:

$$F_\beta = (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall}. \tag{10}$$

Изменение значение параметра β в формуле определяет то, насколько для определения метрики важно значение той или иной метрики: $Precision$ или

$Recall$.

Каждая из представленных выше метрик (в том числе показатели в матрице ошибок) имеет значения в диапазоне от 0 до 1. Чем ближе значение метрики к 1, тем лучше работает модель машинного обучения.

2 Практическая часть

2.1 Описание инструментов и библиотек программной реализации

В течение последнего десятилетия язык программирования Python показал себя как широко используемый язык для создания, обучения и работы с моделями машинного и глубокого обучения с помощью таких инструментов, как scikit-learn [14], TensorFlow [15] и PyTorch [16], которые в основном предоставляют интерфейсы Python. Совокупность библиотек NumPy [17], Pandas [18], Matplotlib [19] и Jupyter Notebook [20] упрощает анализ результатов обучения моделей нейросетей и исследование данных.

Вопрос способа сопоставления текста комментария к фильму и целевого признака решается с помощью представления этой связи в виде таблицы, в которой в первой колонке указывается содержимое комментария, а во второй колонке записывается значение целевого признака (то есть, ”позитивный” отзыв или ”негативный”). Способ хранения таких данных не важен, и основным требованием к нему является простота способа считывания данных, поэтому в качестве формата для хранения таблицы был выбран CSV (Comma-Separated Values) — текстовый формат, предназначенный для представления табличных данных, где значения полей одной строки разделяются между собой запятыми. Для работы с данными такого вида на языке Python самым распространенным является инструментарий библиотеки Pandas.

Чтобы осуществить предобработку текстов комментариев с помощью описанных методов (мешок слов, стемминг, лемматизация и т.д.), был использован Natural Language Toolkit (или NLTK) — набор библиотек и программ, написанных на языке программирования Python, для символьной и статистической обработки естественного языка англоязычных текстов. [21]

Для применения описанных выше алгоритмов машинного обучения был использован инструмент scikit-learn (ранее scikits.learn, также известная как sklearn) — бесплатная библиотека машинного обучения для языка программирования Python. Она представляет собой модуль, объединяющий широкий спектр современных алгоритмов машинного обучения для задач среднего масштаба с учителем и без учителя. Этот пакет методов ориентирован на предоставление машинного обучения неспециалистам с использованием языка высокого уровня общего назначения. Акцент делается на простоте использования, производительности, доступной документации и согласованности API. С помощью

данного инструмента были реализованы и обучены все алгоритмы (наивный байесовский классификатор, метод опорных векторов, логистическая регрессия), а также получены значения всех метрик (accuracy, precision, recall, f-score) и матрицы ошибок.

Для проверки качества обученных алгоритмов была использована библиотека Matplotlib — модуль для создания статических, анимированных и интерактивных визуализаций на Python. Являясь инструментом построения графиков для Python и его расширения для числовой математики NumPy, он предоставляет объектно-ориентированный API для встраивания графиков в приложения с помощью инструментов общего назначения с графическим интерфейсом, таких как Tkinter, wxPython, Qt или GTK. Дополнительные настройки отрисовки графиков позволял осуществить Seaborn — ещё одна библиотека визуализации данных, основанная на matplotlib. Она предоставляет высокоуровневый интерфейс для рисования привлекательных и информативных статистических графиков. При выполнении данной исследовательской работы с помощью Matplotlib и Seaborn было осуществлено построение матриц ошибок с подробной легендой.

2.2 Описание набора данных для обучения и теста

В качестве используемого набора данных для решения поставленной задачи была взята выборка ”IMDB Dataset”. Это набор данных для бинарной классификации тональности текста, содержащий значительно больше данных, чем предыдущие эталонные наборы данных, представленные от IMDB. IMDb — это онлайн-база данных, содержащая информацию о фильмах, телесериалах, подкастах, домашнем видео, видеоиграх и потоковом онлайн-контенте, включая актеров, съемочную группу и личные биографии, краткое изложение сюжета, мелочи, рейтинги, а также фанатские и обзоры критиков. Набор состоит из 50000 обзоров на популярные фильмы, каждый из которых является элементом выборки. Выборка является сбалансированной — представлены два класса объектов-отзывов (”положительный” и ”отрицательный”), и 25000 отзывов относятся к классу положительных, а другие 25000 — к классу отрицательных отзывов.

Набор данных находится в открытом доступе и его можно загрузить как с официального сайта IMDB и сайта с ссылкой на научную работу на тему вектора слов, так и с других различных форумов и сайтов для машинного и глубокого обучения. В частности, для обучения алгоритмов машинного обучения,

описываемых в данной работе, такая коллекция комментариев с соответствующими определенными тональностями, представленная в файле формата CSV, была взята с Kaggle — платформы для проведения соревнований по анализу данных. [13]

Пример двух элементов выборки из CSV-файла:

```
review,sentiment
"One of the other reviewers...with your darker side.",positive
"A wonderful little production...terribly well done.",positive
```

2.3 Программная реализация

Перед применением алгоритмов машинного обучения, была осуществлена предобработка данных, которую содержит файл ”preprocessing.py”. Она состояла из нескольких этапов.

Сначала осуществлялась очистка содержимого комментариев от малоиспользуемых, не соответствующих задаче символов и текстовых структур:

1. Удаление ненужных символов из текстов;
2. Изменение кодировки текста комментария на UNICODE;
3. Удаление частей ссылок (подпись ”http” и т.д.);
4. Удаление символов-эмодзи;
5. Удаление спецсимволов и квадратных скобок из текста;
6. Приведение текстов к нижнему регистру.

Второй этап предобработки представлял собой удаление стоп-слов из элементов выборки. Для этого выбиралось множество стоп-слов, которое было сформировано с помощью библиотеки NLTK, и затем по каждому комментарию проходился ”фильтр”, который удалял каждое встречающееся стоп-слово.

Третьим и четвертым этапом препроцессинга являлись стемминг и лемматизация соответственно.

Осуществив перечисленные выше действия, получилось сформировать облако слов, которое позволило отразить частоту встречающихся в комментариях слов.

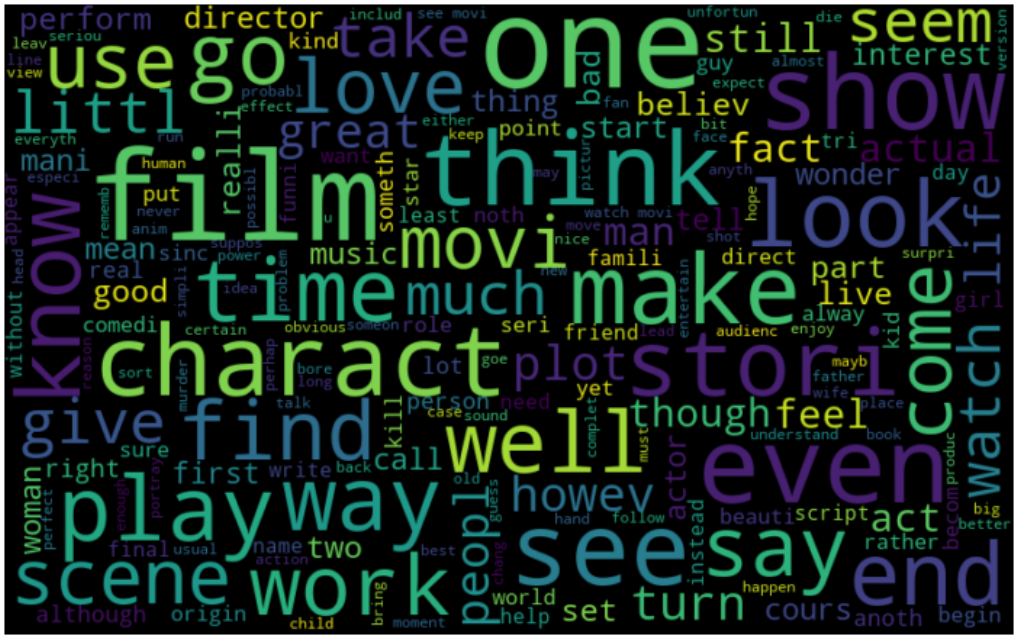

Рисунок 2 – Облако слов в комментариях после предобработки

Далее предобработанная выборка сохранялась в отдельный файл формата CSV, который использовался в дальнейшем.

Файл с предобработанной выборкой открывался с помощью программы ”algos.py”, в котором загруженный набор данных использовался для двух различных экспериментов:

Первый эксперимент начинался с кодировки целевого признака — производилась замена значений ”positive” и ”negative” на 1 и 0 соответственно.

После кодирования самого целевого признака осуществлялось кодирование комментариев с помощью функции ”CountVectorizer” — улучшенной версии ”мешка слов” из библиотеки sklearn, в которой каждый комментарий представлялся вектором, который определял не только факт наличия того или иного слова в комментарии, но и его количества (т.е. вместо значений 0 и 1 для каждого слова из словаря там также содержались натуральные числа).

Затем набор данных разделялся на две части — тренировочную и тестовую выборки, где 80% данных случайным образом распределялись в первую группу. На первом множестве алгоритмы обучались, а на втором — проверялось качество обучения с помощью определенных метрик.

Далее поочередно осуществлялось определение, обучение на тренировочном наборе, предсказание значений на тестовом наборе и подсчет значений матрицы ошибок и всех остальных метрик для алгоритмов:

1. наивный байесовский классификатор;

2. логистическая регрессия;
3. метод опорных векторов.

Второй эксперимент также включал в себя преобразование элементов предобработанной выборки с помощью кодировщика целевого признака, однако к комментариям применялся алгоритм преобразования текстов в матрицу признаков для $TFIDF$.

Векторизованный закодированный набор данных также разделялся на тренировочную и тестовую выборки в соотношении 4 к 1, после чего осуществлялось обучение и проверка метрик алгоритмов в неотличном от первого эксперимента порядке.

2.4 Результаты обучения

Предобработка набора данных и обучение алгоритмов происходило на стационарном компьютере с операционной системой Windows с применением процессора Intel Core i5-8400 с мощностью 2.80 ГГц. В среднем процесс предобработки и обучения любого из алгоритмов занимает около 5 минут реального времени.

В качестве результатов обучения описанных в работе алгоритмов представлена таблица со значениями метрик для каждого из методов предобработки текста и каждого классификатора (BoW — мешок слов, TFIDF — частота слова и обратная частота документа):

	Метрика			
Предобработка и модель	Accuracy	Precision	Recall	F1-score
BoW, MultinomialNB	0.8598	0.8605	0.8598	0.8598
BoW, Лог. регрессия	0.8791	0.8791	0.8791	0.8791
BoW, SVM	0.8597	0.8597	0.8597	0.8597
TFIDF, MultinomialNB	0.8617	0.8624	0.8617	0.8616
TFIDF, Лог. регрессия	0.8948	0.8949	0.8948	0.8947
TFIDF, SVM	0.8909	0.8909	0.8909	0.8909

Таблица 1 – Значения метрик

Также были вычислены значения матриц ошибок:

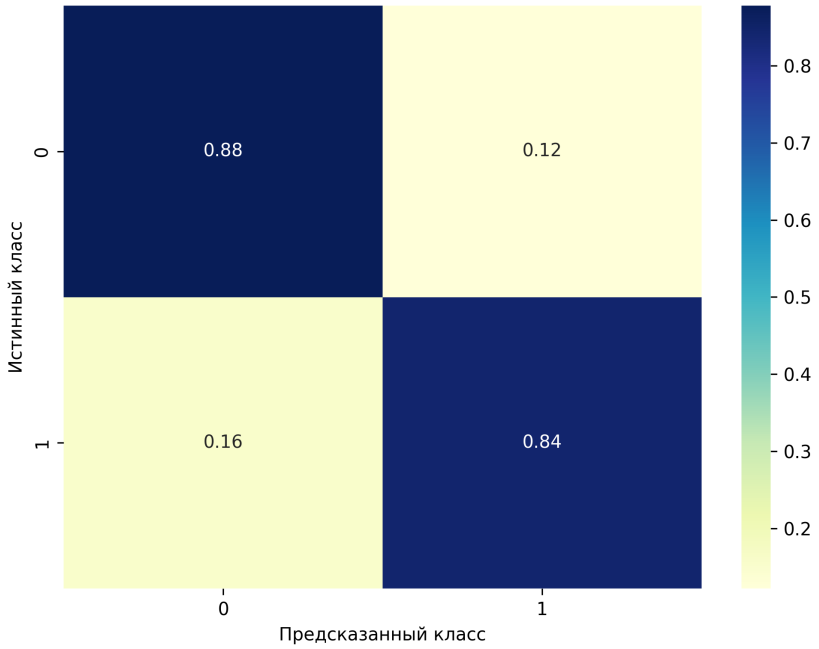

Рисунок 3 – Матрица ошибок наивного байесовского классификатора после применения мешка слов

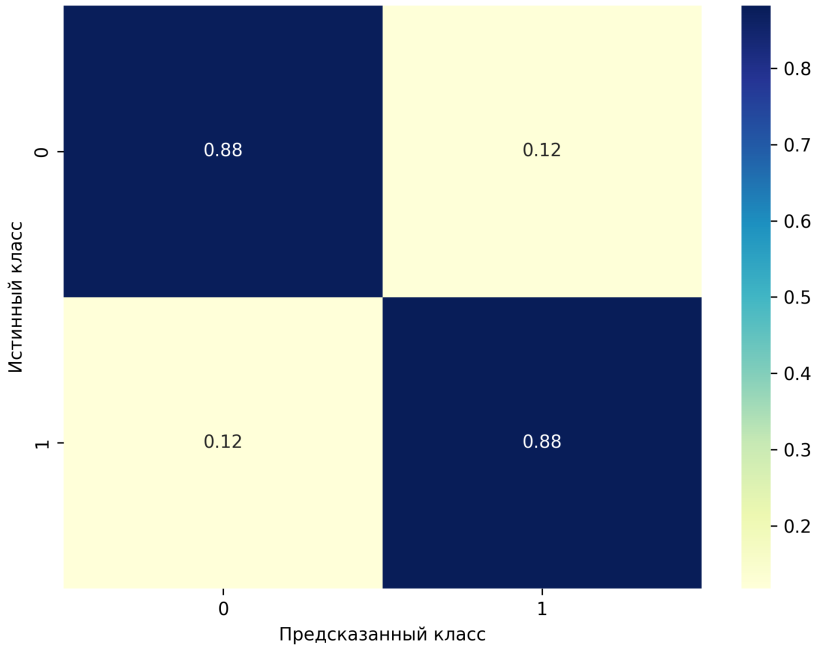

Рисунок 4 – Матрица ошибок логистической регрессии после применения мешка слов

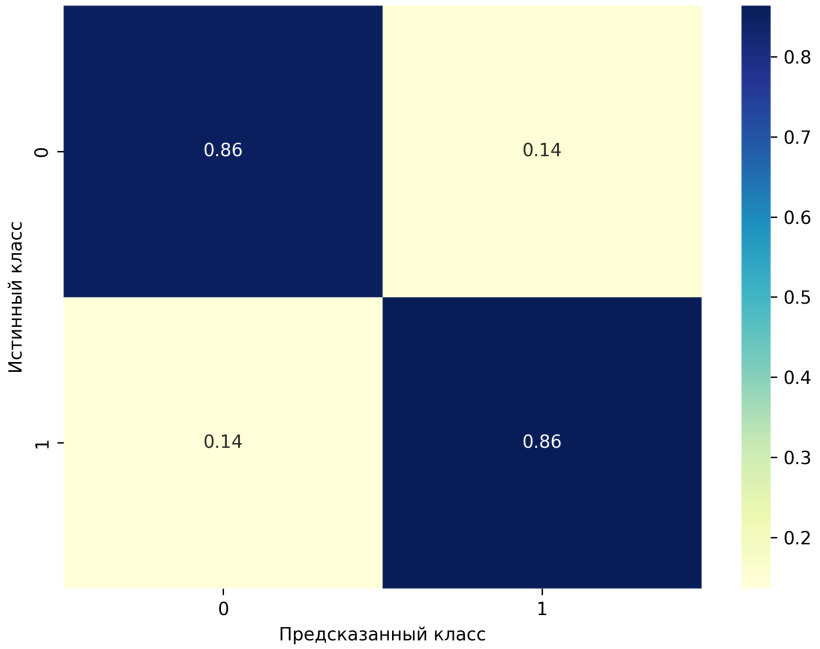

Рисунок 5 – Матрица ошибок метода опорных векторов после применения мешка слов

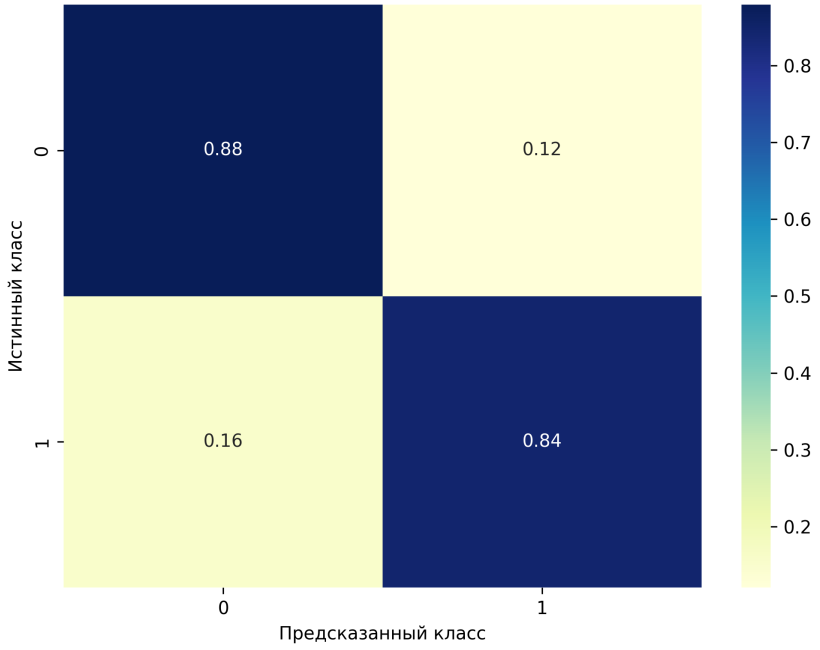

Рисунок 6 – Матрица ошибок наивного байесовского классификатора после создания признаков TFIDF

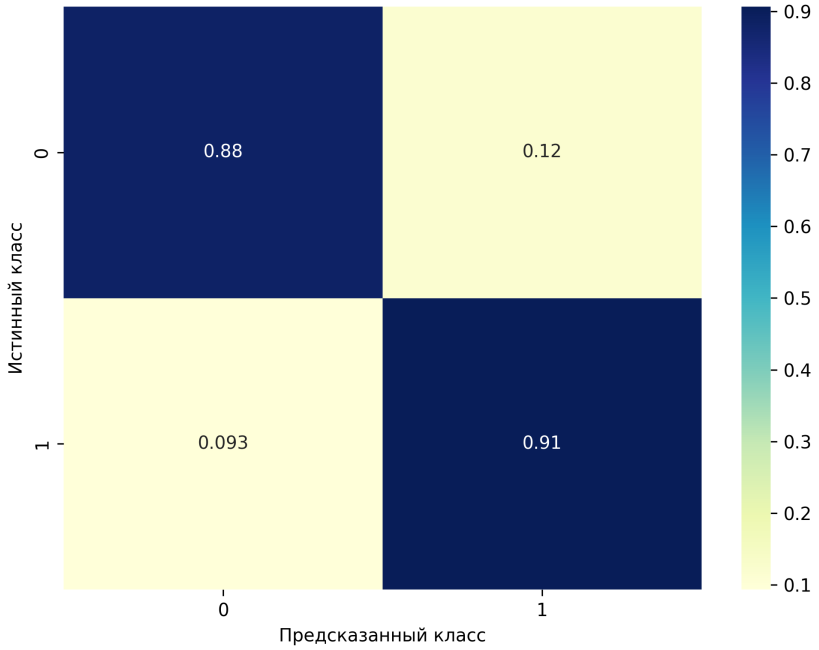

Рисунок 7 – Матрица ошибок логистической регрессии после создания признаков TFIDF

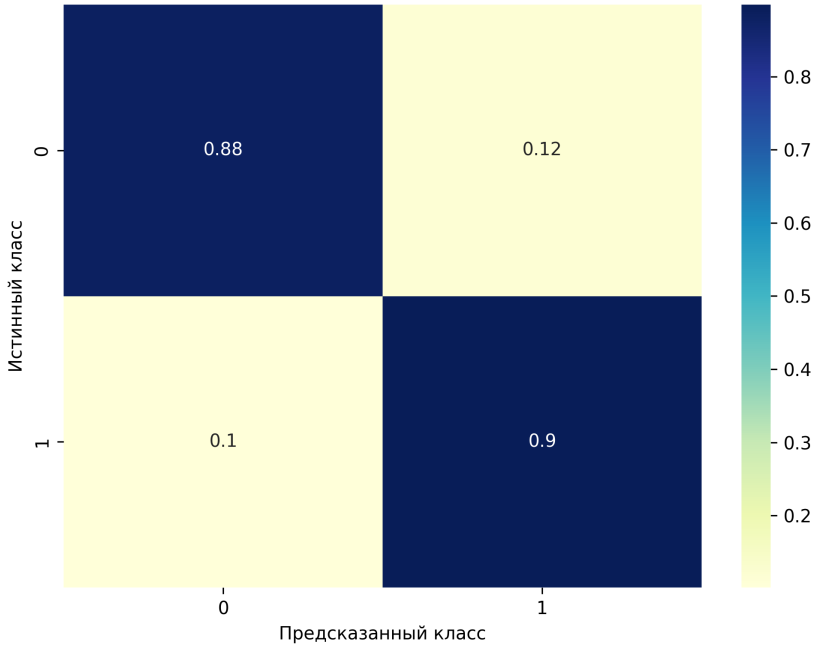

Рисунок 8 – Матрица ошибок метода опорных векторов после создания признаков TFIDF

ЗАКЛЮЧЕНИЕ

На основе полученных результатов обучения, рассмотренных ранее, был сделан ряд выводов:

1. При решении задачи бинарной классификации текстов на сбалансированной выборке такие классические алгоритмы машинного обучения как ”полиномиальный наивный байесовский классификатор”, ”метод опорных векторов”, ”логистическая регрессия” имеют значения метрик ≥ 0.85 (где значение 1 является наилучшим показателем, а значение 0 — наихудшим);
2. Предобработка текста с помощью частоты слова и обратной частоты документа (TFIDF) способствует лучшему обучению модели, нежели предобработка с помощью мешка слов;
3. При предобработке с помощью мешка слов наилучшие значения метрик среди представленных алгоритмов имеет логистическая регрессия;
4. При предобработке с помощью TFIDF наилучшие значения метрик среди представленных алгоритмов имеет также логистическая регрессия, однако разница между этими результатами и значениями метрик метода опорных векторов незначительна (≈ 0.039);
5. Наилучшим подходом для решения задачи анализа тональности отзывов о фильмах с помощью представленных алгоритмов машинного обучения является предварительная обработка текста с помощью применения анализа TFIDF, затем обучение на этой выборке алгоритма логистической регрессии. Значение метрики $Accuracy$ при этом подходе будет равно 0.8948, что меньше на 0.0043 чем лучший показатель значения этой же метрики при решении данной задачи с учетом того же набора данных с помощью алгоритма Linear Support Vector Machine (LSVM) и предобработки с помощью TFIDF в опубликованной в 2018-м году работе [6].

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Короткий, С. Нейронные сети: Основные положения, [Электронный ресурс] : [статья] / URL: http://www.shestopaloff.ca/kyriako/Russian/Artificial_-Intelligence/Some_publications/Korotky_Neuron_network_Lectures.pdf (дата обращения 27.04.2022) Загл. с экрана. Яз. рус.

2 Гудфеллоу Я., Иошуа Б., Курвилль А. Глубокое обучение. – Litres, 2022.

3 Qader W. A., Ameen M. M., Ahmed B. I. An overview of bag of words; importance, implementation, applications, and challenges //2019 international engineering conference (IEC). – IEEE, 2019. – С. 200-204.

4 Большакова Е. И. и др. Автоматическая обработка текстов на естественном языке и компьютерная лингвистика. – 2011.

5 Jalilifard A. et al. Semantic sensitive TF-IDF to determine word relevance in documents //Advances in Computing and Network Communications: Proceedings of CoCoNet 2020, Volume 2. – Singapore : Springer Singapore, 2021. – С. 327-337.

6 Das B., Chakraborty S. An improved text sentiment classification model using TF-IDF and next word negation //arXiv preprint arXiv:1806.06407. – 2018.

7 Zhang H. The optimality of naive Bayes //Aa. – 2004. – Т. 1. – №. 2. – С. 3.

8 Chung M. K. Introduction to logistic regression //arXiv preprint arXiv:2008.13567. – 2020.

9 Hosmer Jr D. W., Lemeshow S., Sturdivant R. X. Applied logistic regression. – John Wiley & Sons, 2013. – Т. 398.

10 Platt J. Probabilistic outputs for SVMs and comparisons to regularized likehood methods //Advances in Large Margin Classifiers. – MIT Press, 1999.

11 Düntsch I., Gediga G. Confusion matrices and rough set data analysis //Journal of Physics: Conference Series. – IOP Publishing, 2019. – Т. 1229. – №. 1. – С. 012055.

12 Flach P. Performance evaluation in machine learning: the good, the bad, the ugly, and the way forward //Proceedings of the AAAI conference on artificial intelligence. – 2019. – Т. 33. – №. 01. – С. 9808-9814.

13 LAKSHMIPATHI N, IMDB Dataset of 50K Movie Reviews, [Электронный ресурс] : [статья] / URL https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews (дата обращения 29.04.2023) — Загл. с экрана.

14 scikit-learn [Электронный ресурс] : [сайт] / URL: https://scikit-learn.org/stable/ (дата обращения 13.05.2022) — Загл. с экрана.

15 tensorflow [Электронный ресурс] : [сайт] / URL: https://www.tensorflow.org (дата обращения 13.05.2023) — Загл. с экрана.

16 PyTorch [Электронный ресурс] : [сайт] / URL: https://pytorch.org/ (дата обращения 13.05.2023) — Загл. с экрана.

17 NumPy. The fundamental package for scientific computing with Python [Электронный ресурс] : [сайт] / URL: https://numpy.org/ (дата обращения 13.05.2023) — Загл. с экрана.

18 pandas [Электронный ресурс] : [сайт] / URL: https://pandas.pydata.org/ (дата обращения 13.05.2023) — Загл. с экрана.

19 Matplotlib: Visualization with Python [Электронный ресурс] : [сайт] / URL: https://matplotlib.org/ (дата обращения 13.05.2023) — Загл. с экрана.

20 Jupyter. Free software, open standards, and web services for interactive computing across all programming languages [Электронный ресурс] : [сайт] / URL: https://jupyter.org/ (дата обращения 13.05.2023) — Загл. с экрана.

21 Natural Language Toolkit [Электронный ресурс] : [сайт] / URL: https://www.nltk.org/ (дата обращения 13.05.2023) — Загл. с экрана.

ПРИЛОЖЕНИЕ А
Код preprocessing.py

```
1  import pandas as pd
2  import numpy as np
3
4  import matplotlib.pyplot as plt
5  import seaborn as sns
6  from wordcloud import WordCloud
7
8  import re
9  from bs4 import BeautifulSoup
10 import nltk
11 from nltk.corpus import stopwords
12 from nltk.tokenize.toktok import ToktokTokenizer
13 from nltk.tokenize import word_tokenize
14 from nltk.stem import WordNetLemmatizer,SnowballStemmer
15 from nltk.tag import pos_tag
16 from nltk.tokenize import word_tokenize
17 nltk.download('punkt')
18 nltk.download('averaged_perceptron_tagger')
19 nltk.download('stopwords')
20 nltk.download('omw-1.4')
21
22
23 def remove_pattern(input_txt, pattern):
24     """Удаление ненужных частей текстов из элементов выборки"""
25     r = re.findall(pattern, input_txt)
26     for word in r:
27         input_txt = re.sub(word, "", input_txt)
28     return input_txt
29
30
31 def strip_html(text):
32     """Преобразовать обрабатываемый текст в формат html библиотеки обработки
33     текста beautifulSoup"""
34     soup = BeautifulSoup(text, "html.parser")
35     return soup.get_text()
36
37
38 def to_unicode(text):
39     """Преобразование текста в кодировку UNICODE с учетом целых цифр и цифр с
```

```
40      плавающей точкой"""
41      if isinstance(text, float) or isinstance(text, int):
42          text = str(text)
43      if not isinstance(text, str):
44          text = text.decode('utf-8', 'ignore')
45      return text
46
47
48  def deleteEmoji(text):
49      """Удаление эмодзи"""
50      regrex_pattern = re.compile(pattern = "["
51          u"\U0001F600-\U0001F64F"
52          u"\U0001F300-\U0001F5FF"
53          u"\U0001F680-\U0001F6FF"
54          u"\U0001F1E0-\U0001F1FF"
55                              "]+", flags = re.UNICODE)
56      return regrex_pattern.sub(r'',text)
57
58
59  def remove_specchars_brackets_spaces(text):
60      """Удаление спецсимволов, скобок и пробелов"""
61      text = re.sub('\[[^]]*\]', ' ', text)
62      text = re.sub(r'[^a-zA-z0-9\s]',' ',text)
63      while "  " in text:
64          text = re.sub('  ', ' ', text)
65      return text
66
67
68  def delete_noise_text(text):
69      """Удаление различных 'шумов' из текста"""
70      text = to_unicode(text)
71      soup = BeautifulSoup(text, "html.parser")
72      text = strip_html(text)
73      text = re.sub(r"http\S+", " ", text)
74      text = deleteEmoji(text)
75      text = text.encode('ascii', 'ignore')
76      text = to_unicode(text)
77      text = remove_specchars_brackets_spaces(text)
78      text = text.lower() # замена больших букв на маленькие
79      return text
80
```

```
81  tokenizer = ToktokTokenizer()
82  stopword_list = nltk.corpus.stopwords.words('english')
83
84  def remove_stopwords(text):
85      """Удаление стоп-слов английского языка из текста"""
86      # stop = set(stopwords.words('english'))
87
88      tokens = tokenizer.tokenize(text)
89      tokens = [token.strip() for token in tokens]
90      filtered_tokens = [token for token in tokens if token.lower() not in
        ↪ stopword_list]
91      filtered_text = ' '.join(filtered_tokens)
92      return filtered_text
93
94
95  def simple_stemmer(text):
96      """Применение стемминга"""
97      ps = SnowballStemmer(language='english')
98      return ' '.join([ps.stem(word) for word in tokenizer.tokenize(text)])
99
100
101 def lemmatize_all(sentence):
102     """Генератор для лемматизации текста"""
103     wnl = WordNetLemmatizer()
104     for word, tag in pos_tag(word_tokenize(sentence)):
105         if tag.startswith("NN"):
106             yield wnl.lemmatize(word, pos='n')
107         elif tag.startswith('VB'):
108             yield wnl.lemmatize(word, pos='v')
109         elif tag.startswith('JJ'):
110             yield wnl.lemmatize(word, pos='a')
111         else:
112             yield word
113
114
115 def lemmatize_text(text):
116     """Лемматизация текста"""
117     return ' '.join(lemmatize_all(text))
118
119
120 def make_wordcloud(df):
```

```
121     """Создание облака слов из текстов выборки, построение гистограммы по
122     классам целевого признака"""
123     all_words = " ".join([sentence for sentence in df['review']])
124     wordcloud = WordCloud(width=800, height=500, random_state=42,
        ↪ max_font_size=100).generate(all_words)
125     plt.figure(figsize=(15,8))
126     plt.imshow(wordcloud, interpolation='bilinear')
127     plt.axis('off')
128     plt.savefig('wordcloud.png', dpi=300)
129
130
131
132 def main():
133     csv_filepath = "review-data/IMDB-Dataset.csv"
134     df = pd.read_csv(csv_filepath)
135     df['review'] = np.vectorize(remove_pattern)(df['review'], "@[\w]*")
136     df['review'] = df['review'].apply(delete_noise_text)
137     df['review'] = df['review'].apply(remove_stopwords)
138     df['review'] = df['review'].apply(simple_stemmer)
139     df['review'] = df['review'].apply(lemmatize_text)
140     df.to_csv("processed_review.csv", index=False)
141
142     df = pd.read_csv("processed_review.csv")
143     make_wordcloud(df)
144
145
146 if __name__ == "__main__":
147     main()
```

ПРИЛОЖЕНИЕ Б

Код algos.py

```
# Обработка csv-файлов
import pandas as pd
# Метрики, визуализация и сохранение файлов
from sklearn.metrics import confusion_matrix, precision_score,
    ↪ accuracy_score,recall_score, f1_score
import matplotlib.pyplot as plt
import seaborn as sns
import os
# Предобработка с помощью мешка слов и tf-idf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
# Модели
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
# Для сохранения моделей
import pickle


def build_conf_matrix(labels, predict, class_name):
    """Функция отображения матрицы искажений"""
    lab, pred = [], []
    for i in range(len(labels)):
        if predict[i] == class_name:
            pred.append(0)
        else:
            pred.append(1)
        if labels[i] == class_name:
            lab.append(0)
        else:
            lab.append(1)
    return confusion_matrix(lab, pred, normalize='true')


def get_confusion_matrix_picture(y_test, y_pred, path):
    get_res = {0: "Negative", 1: "Positive"}
    for i in range(2):
```

```
39          plt.figure(figsize=(8, 6), dpi=70)
40          heatmap = sns.heatmap(build_conf_matrix(y_test, y_pred, i),
41                                annot=True,
42                                cmap='YlGnBu')
43          heatmap.set_title(get_res[i], fontdict={'fontsize':26}, pad=10)
44          plt.xlabel('Предсказанный класс')
45          plt.ylabel('Истинный класс')
46          dir_list = [x[0] for x in os.walk("./")]
47          is_exists = False
48          for dir in dir_list:
49              if path in dir:
50                  is_exists = True
51                  break
52          if not is_exists:
53              os.mkdir(path)
54          plt.savefig(f"{path}_{i}.png", dpi=300)
55
56
57  def data_processing(df, processing_type):
58      X = df["review"]
59      y = df["sentiment"]
60      encoder = LabelEncoder()
61      y = encoder.fit_transform(y)
62
63      if processing_type == "bow":
64          cv = CountVectorizer()
65          X = cv.fit_transform(X).toarray()
66          X_train, X_test, y_train, y_test = train_test_split(X, y,
            ↪  test_size=0.2,
67                                                              random_state = 5)
68      elif processing_type == "tfidf":
69          tv = TfidfVectorizer()
70          tv_fit = tv.fit(X)
71          X_train, X_test, y_train, y_test = train_test_split(X, y,
            ↪  test_size=0.2,
72                                                              random_state = 5)
73          X_train = tv_fit.transform(X_train)
74          X_test = tv_fit.transform(X_test)
75      else:
76          print("Неверный тип предобработки данных!")
77          raise RuntimeError()
```

```
78
79      return X_train, X_test, y_train, y_test
80
81
82  preproc_types = {
83      "1": "bow",
84      "2": "tfidf"
85  }
86
87
88  models = {
89      "1": ("naive bayes", MultinomialNB()),
90      "2": ("log reg", LogisticRegression(penalty='l2',
91                                          max_iter=100,
92                                          C=1,
93                                          random_state=42)),
94      "3": ("svm", LinearSVC(random_state=0, tol=1e-5))
95  }
96
97
98  def fit_predict_model(X_train, X_test, y_train, y_test, model_name,
    ↪  preproc=None):
99      model_name, model = models[model_name]
100     print(f"Обучение модели {model_name}")
101     model.fit(X_train, y_train)
102     print(f"Получение предсказаний модели")
103     y_pred = model.predict(X_test)
104     print("Вычисление метрик")
105     conf_matrix = confusion_matrix(y_test, y_pred)
106     acc = accuracy_score(y_test, y_pred)
107     prec = precision_score(y_test, y_pred ,average='weighted')
108     recall = recall_score(y_test, y_pred, average='weighted')
109     f1 = f1_score(y_test, y_pred, average='weighted')
110
111     path = f'{model_name}_{preproc}_metrics.txt'
112     with open(path, 'w') as f:
113         f.write("conf_matrix: " + str(conf_matrix) + "\n")
114         f.write("accuracy: " + str(acc) + "\n")
115         f.write("precision: " + str(prec) + "\n")
116         f.write("recall: " + str(recall) + "\n")
117         f.write("f1-score: " + str(f1) + "\n")
```

```
118
119     get_confusion_matrix_picture(y_test, y_pred, f"{model_name}_{preproc}")
120
121     print("Сохранение модели")
122     filename = f'{model_name}_{preproc}_model.sav'
123     pickle.dump(model, open(filename, 'wb'))
124
125
126 def main():
127     csv_filepath = "processed_review.csv"
128     df = pd.read_csv(csv_filepath)
129
130     print("Выберите способ предобработки данных (введите 1 или 2):")
131     print("1. Bag-of-words")
132     print("2. TF-iDF")
133     preproc_type = input()
134
135     print("Выберите алгоритм машинного обучения (введите 1, 2 или 3):")
136     print("1. Multinomial Naive Bayes")
137     print("2. Logistic Regression")
138     print("3. Support Vector Classifier")
139     model_type = input()
140
141     preproc = preproc_types[preproc_type]
142     print("Препроцессинг данных")
143     X_train, X_test, y_train, y_test = data_processing(df, preproc)
144     fit_predict_model(X_train, X_test, y_train, y_test, model_type, preproc)
145
146
147 if __name__ == "__main__":
148     main()
```