**Министерство науки и высшего образования**

**Российской Федерации**

**ФЕДЕРАЛЬНОЕ Государственное Автономное**

**образовательное учреждение высшего образования**

**«новосибирский национальный исследовательский государственный университет»**

**Факультет информационных технологий**

**РЕФЕРАТ**

**На тему: «Сравнение методов решения задачи классификации комментариев в Интернете»**

Студенты:

Заляева Александра Вячеславовна

Юрин Марк Антонович

Новосибирск 2024

**Цель работы**: сравнить различные алгоритмы классификации текстовой информации на примере анализа комментариев к театрам в Интернете.

Было решено …

**Теоретическая часть**

Классификация – один из разделов машинного обучения, посвященный задаче разделения множества объектов на несколько заранее определенных классов. Задача классификации решается с помощью различных методов обучения с учителем.

Методы решения поставленной задачи:

* Наивный байесовский классификатор
* Стохастический градиентный спуск
* K-Ближайших соседей
* Дерево решений
* Случайный лес
* Искусственные нейронные сети
* Метод опорных векторов

**Наивный байесовский классификатор**

Наивный метод Байеса — это алгоритм обучения, основанный на применении теоремы Байеса с «наивным» предположением об условной независимости между каждой парой характеристик при заданном значении переменной класса. Теорема Байеса утверждает следующее отношение, учитывая переменную класса y и зависимый вектор признаков (x1, …, xn):

Используя наивное предположение об условной независимости. А так же то, что P(x1,…,xn) является константой с учетом входных данных, мы можем использовать следующее правило классификации:

Несмотря на упрощенные предположения, наивные байесовские классификаторы довольно хорошо работают во многих реальных ситуациях, хорошо проводя классификацию. Им требуется небольшой объем обучающих данных для оценки необходимых параметров.

В нашей работе наивный байесовский метод представлен классом MultinomialNB из библиотеки sklearn.

**Стохастический градиентный спуск**

Метод градиентного спуска — численный [метод](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%B4%D0%B8%D0%B5%D0%BD%D1%82%D0%BD%D1%8B%D0%B5_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D1%8B) нахождения локального минимума или максимума [функции](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D0%B5%D0%B2%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) с помощью движения вдоль [градиента](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%B4%D0%B8%D0%B5%D0%BD%D1%82), один из основных численных методов современной оптимизации.

Алгоритм:

1. Задать начальное приближение (x0, …, xn) = и точность расчета
2. Найти
3. Если , то . Иначе .

Идея состоит в том, чтобы повторять шаги в направлении, противоположном градиенту функции в текущей точке, потому что это будет направлением наискорейшего спуска. Цель градиентного спуска — минимизация функции потерь. Достоинство этого способа обучения состоит в его эффективности и простоте реализуемости.

В нашей работе представлен классом sklearn.linear\_model.SGDClassifier

**K-Ближайших соседей**

Это алгоритм отложенного обучения, который хранит все экземпляры, соответствующие обучающим данным, в n-мерном пространстве. Это алгоритм ленивого обучения**,** так как он не фокусируется на построении общей внутренней модели, вместо этого он работает над хранением экземпляров обучающих данных. Классификация производится простым голосованием k ближайших соседей каждой точки. Чтобы обозначить новую точку, программа просматривает помеченные точки, ближайшие к этой новой точке, также известные как ее ближайшие соседи. Эти соседи голосуют, поэтому любая метка, которая есть у большинства соседей, является меткой для новой точки. k - количество соседей, которые проверяются.

В нашей работе метод представлен классом sklearn.neighbors.KNeighborsClassifier

**Дерево решений**

Деревья решений — это непараметрический контролируемый метод обучения, используемый для [классификации](https://scikit-learn.ru/1-10-decision-trees/#tree-classification) и [регрессии](https://scikit-learn.ru/1-10-decision-trees/#tree-regression) . Цель состоит в том, чтобы создать модель, которая предсказывает значение целевой переменной, изучая простые правила принятия решений, выведенные из характеристик данных. На рёбрах («ветках») дерева решения записаны признаки, от которых зависит целевая функция, в «листьях» записаны значения [целевой функции](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D0%B5%D0%B2%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F), а в остальных узлах — признаки, по которым различаются случаи. Чтобы классифицировать новый случай, надо спуститься по дереву до листа и выдать соответствующее значение.

В нашей работе метод представлен классом sklearn.tree.DecisionTreeClassifier

**Случайный лес**

Метод случайного леса — алгоритм [машинного обучения](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), заключающийся в использовании ансамбля [решающих деревьев](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D0%BF%D1%80%D0%B8%D0%BD%D1%8F%D1%82%D0%B8%D1%8F_%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B9). Алгоритм применяется для задач классификации, регрессии и кластеризации. Основная идея заключается в использовании большого [ансамбля](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D1%81%D0%B0%D0%BC%D0%B1%D0%BB%D1%8C_(%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)) [решающих деревьев](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D0%BF%D1%80%D0%B8%D0%BD%D1%8F%D1%82%D0%B8%D1%8F_%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B9), каждое из которых само по себе даёт очень невысокое качество классификации, но за счёт их большого количества результат получается хорошим.

Классификация объектов проводится путём голосования: каждое дерево комитета относит классифицируемый объект к одному из классов, а побеждает класс, за который проголосовало наибольшее число деревьев.

В нашей работе этот метод не представлен, но его можно найти в sklearn.tree.DecisionTreeClassifier

**Искусственные нейронные сети**

Нейронная сеть состоит из нейронов, которые расположены слоями, они принимают некоторый входной вектор и преобразуют его в выходной. Процесс включает в себя то, что каждый нейрон принимает входные данные и применяет к ним функцию, которая часто является нелинейной функцией, а затем передает выходные данные на следующий уровень.

В представленной работе в качестве нейронной сети использовался многослойный перцептрон. Многослойный персептрон (MLP) — это алгоритм обучения с учителем, который применяет функцию  к набору данных, где m — количество измерений для ввода и o - количество размеров для вывода.  Между входным и выходным слоями может быть один или несколько нелинейных слоев, называемых скрытыми слоями. Каждый нейрон в скрытом слое преобразует значения из предыдущего слоя взвешенным линейным суммированием , за которой следует нелинейная функция активации. Выходной слой получает значения из последнего скрытого слоя и преобразует их в выходные значения.

В нашей работе MLP представлен классом sklearn.neural\_network.MLPClassifier

**Метод опорных векторов**

Метод опорных векторов - это метод машинного обучения, целью которого является попытка классифицировать входные наборы данных в один из двух классов. Для построения модели метода опорных векторов нужно взять обучающие входные данные, отобразить их в многомерное пространство, а затем использовать регрессию, чтобы найти [гиперплоскость](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%BF%D0%B5%D1%80%D0%BF%D0%BB%D0%BE%D1%81%D0%BA%D0%BE%D1%81%D1%82%D1%8C) с наибольшим зазором в этом пространстве, которая лучше всего разделяла бы два класса входных данных. После обучения модели она способна классифицировать новые входные данные в один из классов при помощи разделяющей гиперплоскости. Метод опорных векторов возможно применять только для линейно разделимых классов.

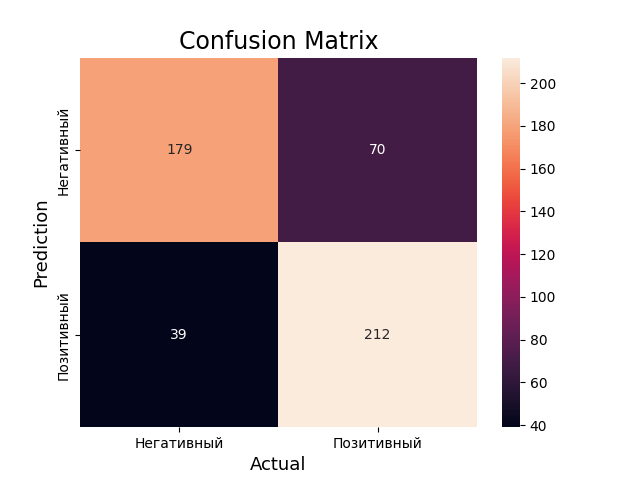
В нашей работе метод представлен классом sklearn.svm.LinearSVC

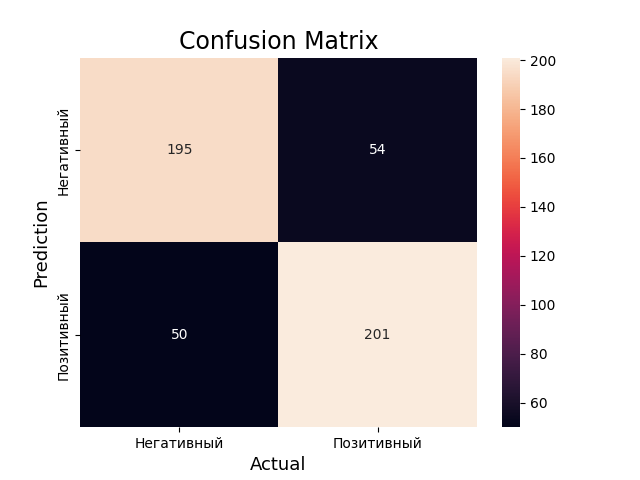
**Сравнение методов классификации**

Обучение проводилось на выборке размером 1500 комментариев, тестирование – на выборке размером 500. Выборки состоят из комментариев о различных театрах России, взятых из Google.

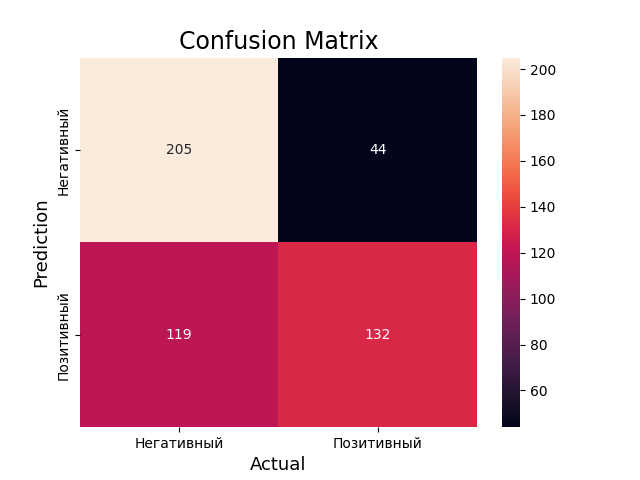
|  |  |  |
| --- | --- | --- |
| Название метода | Время обучения, сек | Точность на обучающей выборке, % |
| Наивный байесовский классификатор | 0.006023 | 78.2 |
| Стохастический градиентный спуск | 0.025934 | 79.2 |
| K-Ближайших соседей | 0.001042 | 67.8 |
| Дерево решений | 0.071842 | 71 |
| Искусственные нейронные сети | 5.207116 | 76.8 |
| Метод опорных векторов | 0.031917 | 80.6 |

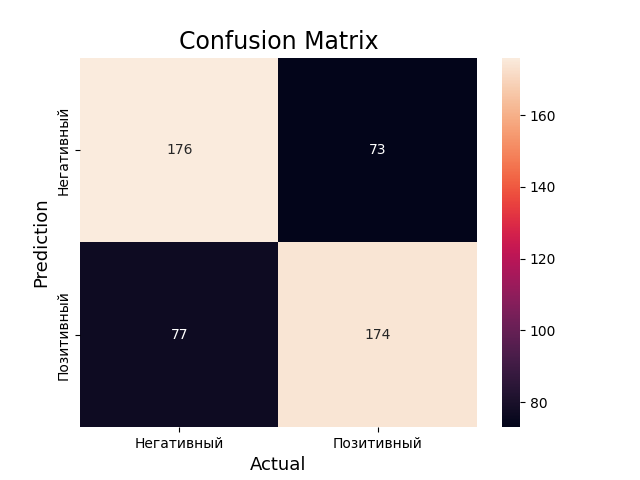
**Приложения**

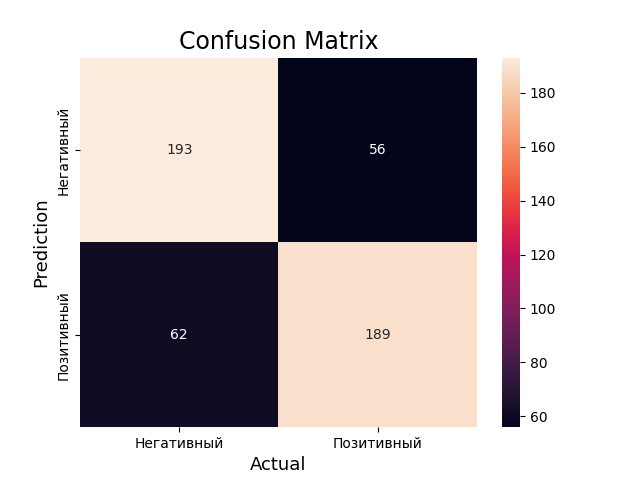
Рис. матрица ошибок для наивного байесовского метода

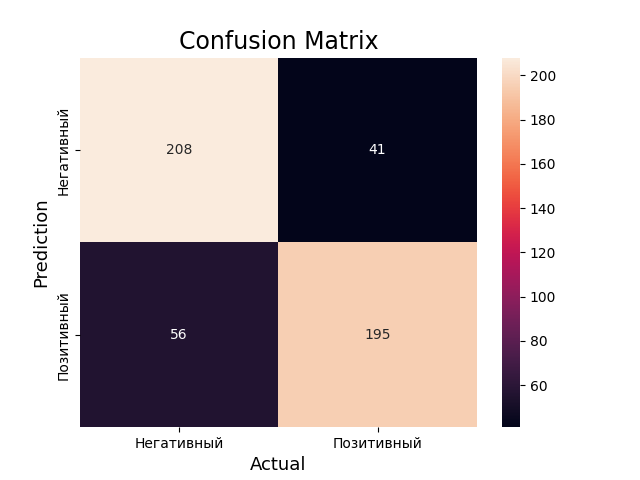
****Рис. матрица ошибок для стохастического градиентного

спуска

Рис. матрица ошибок для метода k-ближайших соседей

Рис. матрица ошибок для дерева решений

Рис. матрица ошибок для многослойного перцептрона

Рис. матрица ошибок для метода опорных векторов

**Список литературы**

<https://scikit-learn.org/stable/modules/naive_bayes.html>

<https://ru.wikipedia.org/wiki/Градиентный_спуск>

<https://scikit-learn.org/stable/modules/sgd.html>

<https://www.mql5.com/ru/articles/11200#:~:text=Согласно.%20Википедии%2C%20градиентный%20спуск%20—,к%20локальному%20максимуму%20этой%20функции>

<https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm>

<https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>

<https://scikit-learn.ru/1-10-decision-trees/>

<https://ru.wikipedia.org/wiki/Дерево_решений>

<https://ru.wikipedia.org/wiki/Метод_случайного_леса>

<https://scikit-learn.ru/1-11-ensemble-methods/>

<https://scikit-learn.org/stable/modules/neural_networks_supervised.html>

<https://scikit-learn.org/stable/modules/svm.html>

<https://ru.wikipedia.org/wiki/Метод_опорных_векторов>