**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

**СОДЕРЖАНИЕ**

[ЦЕЛЬ 3](file:///C:\Users\Dmitry\Downloads\Pract3_Юрин_206.docx#__RefHeading___Toc18443921)

[ЗАДАНИЕ 3](file:///C:\Users\Dmitry\Downloads\Pract3_Юрин_206.docx#__RefHeading___Toc18443921)

[МЕТОДЫ КЛАССИФИКАЦИИ 3](file:///C:\Users\Dmitry\Downloads\Pract3_Юрин_206.docx#__RefHeading___Toc18443922)

Наивный байесовский классификатор4

Стохастический градиентный спуск4

K-Ближайших соседей6

Дерево решений9

Случайный лес6

Искусственные нейронные сети9

Метод опорных векторов9

СРАВНЕНИЕ МЕТОДОВ КЛАССИФИКАЦИИ9

ВЫВОД9

[СПИСОК ЛИТЕРАТУРЫ](file:///C:\Users\Dmitry\Downloads\Pract3_Юрин_206.docx#__RefHeading___Toc18443924) 10

[Приложение 1. Матрицы ошибок](file:///C:\Users\Dmitry\Downloads\Pract3_Юрин_206.docx#__RefHeading___Toc18443925) 11

**ЦЕЛЬ**

Целью работы являются изучение методов классификации текстовой информации и получение практических навыков их использования. Классификация является одной из основных задач компьютерной лингвистики. Скорость и качество классификации текстов непосредственно зависят от применяемых методов, поэтому их изучение позволит более эффективно решать задачи анализа данных. Анализ и классификация текстов имеют большое значение в сфере обеспечения общественной безопасности (автоматический поиск противоправной информации) и бизнеса (интеллектуальные помощники, спам-фильтры). Так как компьютерные технологии всё больше и больше проникают в жизнь человека, эта тема будет актуальна ещё долгое время.

**ЗАДАНИЕ**

Для выполнения работы были поставлены следующие задачи:

1. Изучение методов классификации текстовой информации.
2. Написание программы, осуществляющей парсинг комментариев к географическим местам из сервиса Google Maps, а именно к организациям театров. Программа должна получать комментарии из интернета и обрабатывать их определенным образом.
3. Написание программы, осуществляющей классификацию комментариев, используя изученные методы. Для комментариев выделено 2 класса: “позитивные” и “негативные”.
4. Проведение сравнительного анализа результатов классификации и определения лучшего метода классификации для поставленной задачи.

**МЕТОДЫ КЛАССИФИКАЦИИ**

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.

Существует множество методов решения задачи классификации:

* Наивный байесовский классификатор (NB)
* Стохастический градиентный спуск (SGD)
* K-ближайших соседей (KNN)
* Дерево решений (DT)
* Случайный лес (RF)
* Искусственные нейронные сети (FFBP, DAN2, CNN, MLP)
* Метод опорных векторов (SVM)

**Наивный байесовский классификатор**

Наивный метод Байеса - 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:

, где у - метка класса и (x1, …, xn) – вектор зависимых признаков.

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 достижения приемлемой для практических задач точности классификации.

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, то есть в направлении 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 и метка, которая есть у большинства соседей, становится меткой для новой точки. k - количество соседей, которые проверяются.

В нашей работе метод представлен классом sklearn.neighbors.KNeighborsClassifier

**Дерево решений**

Дерево 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). Метод основан на создании модели, предсказывающей значение целевой переменной путём изучения простых правил принятия решений, выведенных 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 заключается в передаче каждому нейрону входных данных. Нейрон применяет к ним нелинейную функцию, 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 — число измерений для ввода и n – число измерений 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, находя скалярное произведение и применяя нелинейную функцию 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 - метод машинного обучения, целью которого является классификация входных наборов 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 разделяет 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 |

**ВЫВОД**

###### СПИСОК ЛИТЕРАТУРЫ

1. Naive Bayes // <https://scikit-learn.org/stable/modules/naive_bayes.html> (дата обращения: 03.04.2024).
2. Градиентный спуск // <https://ru.wikipedia.org/wiki/Градиентный_спуск> (дата обращения: 03.04.2024).
3. Stochastic Gradient Descent // <https://scikit-learn.org/stable/modules/sgd.html> (дата обращения: 03.04.2024).
4. Машинное обучение и Data Science (Часть 06): Градиентный спуск // <https://www.mql5.com/ru/articles/11200#:~:text=Согласно.%20Википедии%2C%20градиентный%20спуск%20—,к%20локальному%20максимуму%20этой%20функции> (дата обращения: 04.04.2024).
5. k-nearest neighbors algorithm // <https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm> (дата обращения: 05.04.2024).
6. sklearn.neighbors.KNeighborsClassifier // <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html> (дата обращения: 05.04.2024).
7. Деревья решений // <https://scikit-learn.ru/1-10-decision-trees/> (дата обращения: 05.04.2024).
8. Дерево решений // <https://ru.wikipedia.org/wiki/Дерево_решений> (дата обращения: 05.04.2024).
9. Метод случайного леса // <https://ru.wikipedia.org/wiki/Метод_случайного_леса> (дата обращения: 06.04.2024).
10. Ансамблевые методы // <https://scikit-learn.ru/1-11-ensemble-methods/> (дата обращения: 06.04.2024).
11. Neural network models (supervised) // <https://scikit-learn.org/stable/modules/neural_networks_supervised.html> (дата обращения: 04.04.2024).
12. Support Vector Machines // <https://scikit-learn.org/stable/modules/svm.html> (дата обращения: 04.04.2024).
13. Метод опорных векторов // <https://ru.wikipedia.org/wiki/Метод_опорных_векторов> (дата обращения: 04.04.2024).
14. The Selenium Browser Automation Project // <https://www.selenium.dev/documentation/> (дата обращения: 03.04.2024).
15. XPath Tutorial // [XPath Tutorial (w3schools.com)](https://www.w3schools.com/xml/xpath_intro.asp) (дата обращения: 03.04.2024).

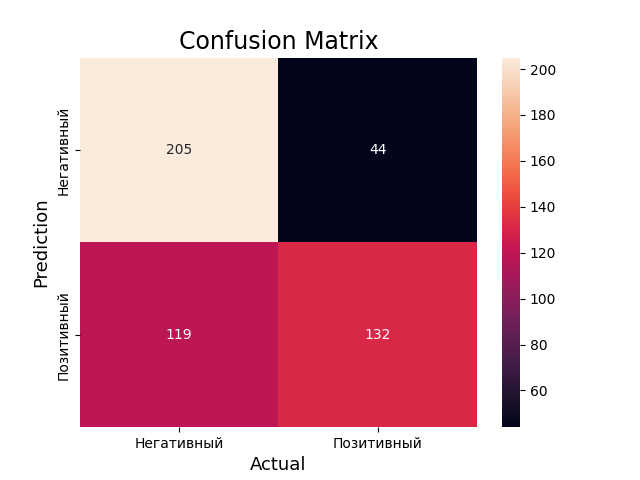
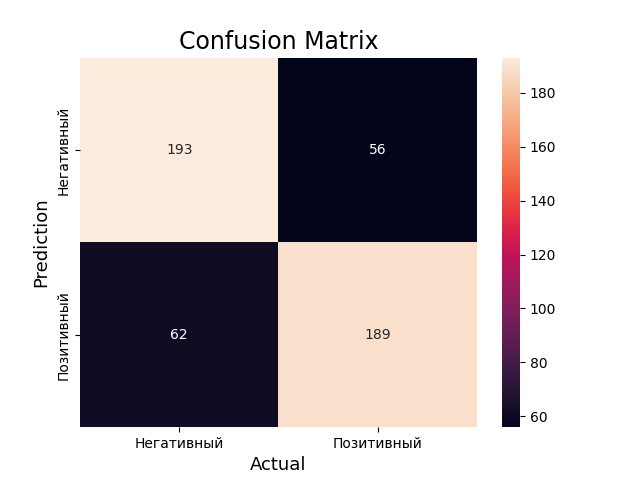
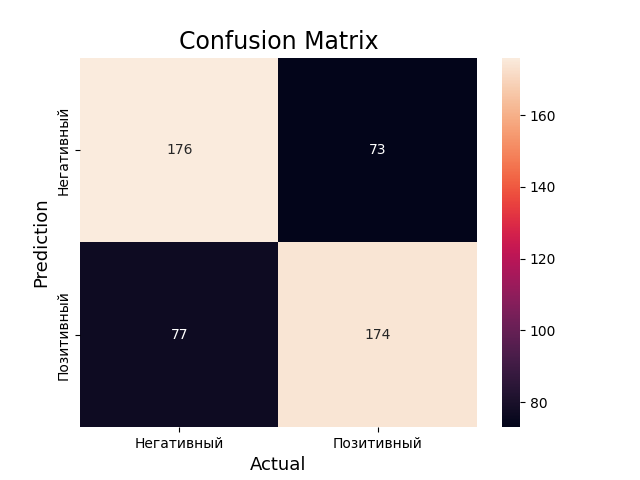
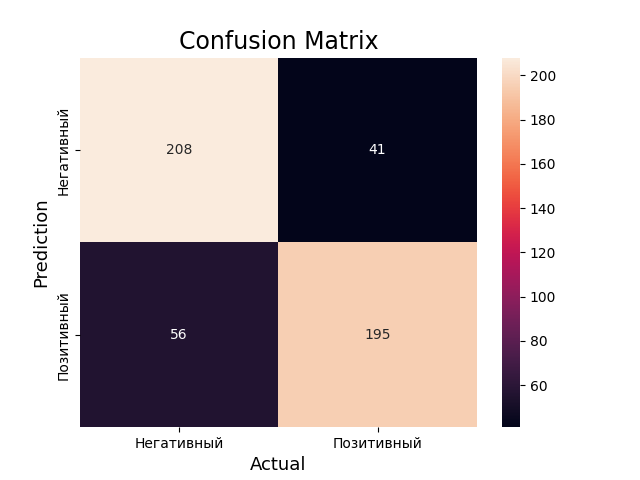
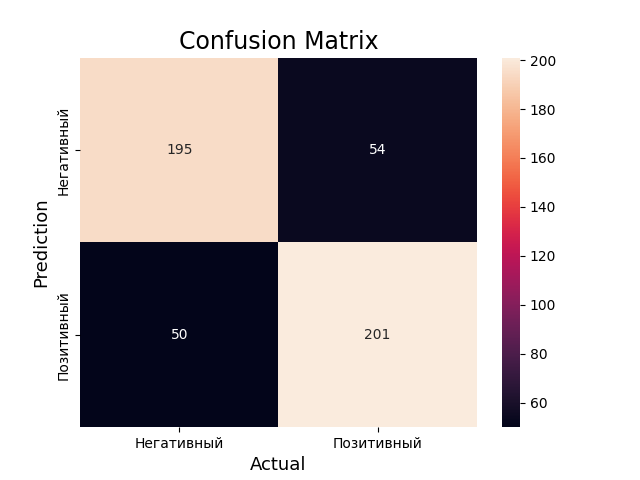
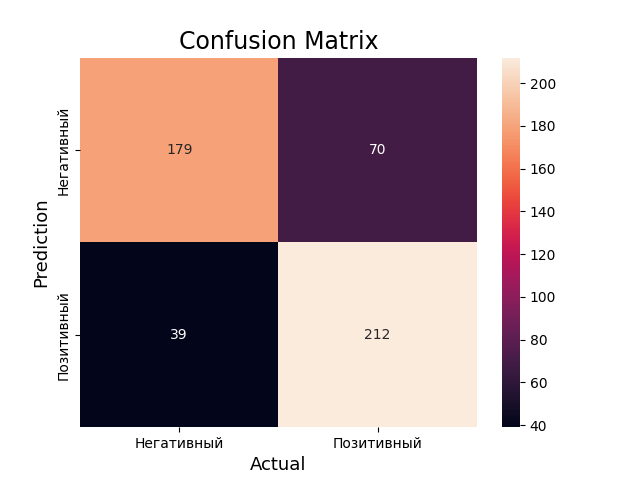
******Приложение 1. Матрицы ошибок**

Рис. 5. Матрица ошибок для метода многослойного перцептрона

Рис. 4. Матрица ошибок для метода дерева решений

Рис. 2. Матрица ошибок для стохастического градиентного спуска

Рис. 1. Матрица ошибок для метода наивного байеса

Рис. 3. Матрица ошибок для метода k-ближайших соседей

Рис. 6. Матрица ошибок для метода опорных векторов