**титульный лист**

Оглавление

[**Введение.** 4](#_Toc137560283)

[**Глава 1. Предварительная обработка текстовых данных и постановка задачи исследования** 7](#_Toc137560284)

[**1.1** **Постановка задачи классификации** 7](#_Toc137560285)

[**1.2 Процедура предварительной обработки теста** 8](#_Toc137560286)

[**1.3 Метрики качества классификации** 10](#_Toc137560287)

[**1.4 Выводы по главе:** 11](#_Toc137560288)

[**Глава 2. Методы классификации** 12](#_Toc137560289)

[**D2.1 Логистическая регрессия** 12](#_Toc137560290)

[**2.2 К-ближайших соседей** 15](#_Toc137560291)

[**2.3 Деревья решений** 18](#_Toc137560292)

[**2.4 Случайный лес** 22](#_Toc137560293)

[**Глава 3. Бинарная классификация** 25](#_Toc137560294)

[**2.2 Сокращение размерности словаря терминов.** 26](#_Toc137560295)

[**2.3 Визуализация** 30](#_Toc137560296)

[**2.4 Классификация** 32](#_Toc137560297)

[**Глава 4. Мультиклассовая классификация** 39](#_Toc137560298)

[**3.1 Описание выборки** 40](#_Toc137560299)

[**3.2 Классификация** 41](#_Toc137560300)

[**Описание используемого программного обеспечения** 49](#_Toc137560301)

[**Список литературы (демо версия) Очень плохо - расширяй** 50](#_Toc137560302)

Синий цвет – мои правки-добавления,

Красный цвет – ошибки-замечания

Текст должен быть нормально оформлен: главы, параграфы, форматирование «по ширине», ссылки на литературу. ГДЕ?

Надо объединить в одном отчете теорию, формирование выборок и результаты экспериментальных исследований. Пока все очень разрозненно.

**Введение.**

Интеллектуальный анализ текстовых данных является важным направлением исследований в области искусственного интеллекта и машинного обучения. Он предназначен для извлечения информации из текстовых данных и позволяет автоматически обрабатывать, классифицировать, анализировать и интерпретировать большие объемы текстовой информации.

Среди основных задач, которые решаются в рамках интеллектуального анализа текстовых данных, можно выделить:

* Классификация текстов: определение категории, к которой относится текст
* Извлечение информации: автоматическое извлечение информации из текстов (например, определение ключевых слов и фраз)
* Кластеризация: разделение текстов на группы по сходству
* Анализ тональности: определение тональности текста (положительная, отрицательная, нейтральная);
* Анализ эмоциональной окраски текста: определение эмоциональной окраски текста (радость, грусть, злость, страх и т.д.);
* Генерация текстов: автоматическая генерация текстов на основе заданных параметров

Таким образом, интеллектуальный анализ текстовых данных имеет множество применений может быть использован в различных сферах, например

* **Разделение web-страниц на разделы**
* **Боты в мессенджерах**
* **Таргетинговая реклама**
* **Сбор статистики**
* **Определение языка текста**
* **Поисковые системы**

Классификацию можно разделить на бинарную и многоклассовую, об особенностях каждой поговорим попозже, сейчас же кратко опишем.

Бинарная классификация – это задача машинного обучения, которая заключается в разделении объектов на две категории на основе определенных признаков объектов. Например, можно классифицировать электронные письма на "спам" и "не спам", определять, является ли пациент больным или здоровым, определять, совершит ли пользователь покупку или нет и т.д.

Для бинарной классификации используются различные алгоритмы машинного обучения, такие как логистическая регрессия, деревья решений, метод опорных векторов и др. Эти алгоритмы строят гиперплоскость, которая разделяет объекты на две категории.

Многоклассовая классификация – это задача машинного обучения, которая заключается в разделении объектов на три и более категорий на основе их признаков. Например, можно классифицировать изображения на несколько категорий, таких как "кошки", "собаки" и "автомобили", определять, какому жанру принадлежит фильм, и т.д.

Для многоклассовой классификации также используются различные алгоритмы машинного обучения, такие как логистическая регрессия, метод опорных векторов, случайный лес, нейронные сети и др. В отличие от бинарной классификации, где модель обучается разделять объекты на две категории, модель многоклассовой классификации должна научиться разделять объекты на три и более категорий.

Работа будет посвящена классификации научных документов – это документы, представляющие собой научную литературу, в них присутствует большое количество специфических научных терминов, как раз они могут служить классообразующими признаками и составить признаковое пространство.

В данной работе исследуется влияние системы признаков на качество классификации. Это означает, что неизвестная целевая функция должна быть аппроксимирована для различного набора терминов.

Для исследования будут использоваться выборки из научных документов, в качестве систем признаков мы рассмотрим термины из названий документов и библиографических описаний. Библиографическое описание – это краткое изложение научного документа, по которому можно примерно понять его содержание. Основное отличие заключается в размерности, которая может отличаться на порядок. Необходимо определить, как сильно влияет используемое количество терминов на качество классификации текстов. Важно отметить, что пространства меньшей размерности позволяют затрачивать меньше вычислительных мощностей, упрощать процесс создания самих признаковых пространств, а также упрощают их интерпретацию.

Целью работы является анализ качества классификации текстовых документов по научным тематикам в зависимости от используемой системы признаков.

В данной работе применяются две системы (множества) признаков. Первое множество включает термины из всех разделов БО. Второе содержит только термины из названий статей.

В качестве показателей качества используются precision, recall, F1-score, accuracy.

Для достижения поставленной цели необходимо решить следующие задачи:

* Предварительная обработка текста
* Векторизация
* Классификация

1. Логистическая регрессия
2. К-ближайших соседей
3. Дерево решений
4. Случайный лес

# **Глава 1. Предварительная обработка текстовых данных и постановка задачи исследования**

## **Постановка задачи классификации**

Задача классификации относится к области машинного обучения и заключается в присвоении объектам определенных меток классов на основе их характеристик, называемых признаками. Дано множество объектов , множество меток классов и обучающая выборка состоящая из пар , где - вектор признаков -го объекта, а - метка класса для этого объекта.

Задача классификации заключается в нахождении алгоритма , который по вектору признаков предсказывает метку класса . Для этого используется обучающая выборка , на которой модель обучается находить закономерности в признаках объектов, связанных с их метками классов.

Функция может быть реализована разными алгоритмами, например, логистической регрессией, методом опорных векторов (SVM), деревьями решений, к-ближайших соседей, случайным лесом и д.р. После обучения модели на обучающей выборке, ее можно применять для классификации новых объектов, то есть предсказания их меток классов.

Оценка качества модели происходит на основе метрик, таких как полнота (recall), точность (precision), аккуратность (accuracy) и д.р. Целью обучения модели является максимизация качества предсказаний на тестовой выборке или в реальном мире.

**1.2 Процедура предварительной обработки теста**

Предварительная обработка текста, используется для очистки текстовых данных. Текстовые данные содержат различный «шум» — например, эмодзи, знаки препинания, чередование строчных и прописных символов. Cтоит заметить, что машины не понимают человеческий язык: текст, звуки, изображения, всю информацию к которой привык человек необходимо обрабатывать, поэтому после предварительной обработки данные представляются в виде чисел, в текстовом анализе для этого используется матрица документ-термин, Tf-idf, а так же другие способы векторизации.

Обработка текстовых данных для машинного обучения может быть довольно сложной задачей, поскольку текст является неструктурированным и содержит много различных типов информации, таких как слова, фразы, контекст, тональность, чувства, сленг, опечатки, аббревиатуры и т.д.

Частью обработки естественного языка является токенизация, в результате чего мы получаем большое число уникальных терминов, такие методы как стемминг и лемматизация помогают сократить число уникальных терминов, но оно так же остается большим.

Стемминг представляет собой процесс обрезания слова до его основы путем удаления окончаний и аффиксов. Основная цель стемминга - сократить слово до его базовой формы, чтобы учитывать только его смысловое значение, игнорируя грамматические изменения. В результате этого процесса могут получаться некорректные или несуществующие слова.

Пример стемминга: «бегающий» - «бега», «дома» - «дом»

Лемматизация, в отличие от стемминга, приводит слова к их леммам или базовым словарным формам, учитывая грамматические правила и контекст. Лемматизация основывается на знании о частях речи и использует словарь или базу данных для определения правильной леммы. Это позволяет сохранить семантическое значение слова и производить корректные словоформы.

Пример: «бегающий» - «бегать», «дома» - «дом»

В работе предварительная обработка состоит из следующих шагов:

• преобразование в нижний регистр

• удаление знаков препинания

• удаление стоп-слов

• токенизация (разделение текста на слова и словосочетания)

• лемматизация

• векторное представление слов с использованием CountVectorizer и TfidfVectorizer (в ходе исследования было принято решение использовать TF-IDF)

Взвешивание Tf-idf (Term Frequency-Inverse Document Frequency) используется для взвешивания значимости термов (слов) в документе относительно коллекции документов. Она вычисляется как произведение двух компонентов:

Tf-idf позволяет отличить важные термы, которые часто встречаются в конкретном документе (высокий TF) и редкие термы, которые встречаются в небольшом количестве документов в коллекции (высокий IDF). Комбинирование TF и IDF позволяет получить числовое значение, отражающее важность терма в контексте конкретного документа и всей коллекции.

## **1.3 Метрики качества классификации**

Матрица неточностей (Confusion matrix) является инструментом для оценки качества алгоритма машинного обучения в задачах классификации. Она позволяет визуализировать, насколько успешно алгоритм классифицирует объекты каждого класса.

Матрица неточностей (ошибок) состоит из четырех значений: true positives (TP), false positives (FP), true negatives (TN) и false negatives (FN).

True positives (TP) - это количество объектов, которые были правильно отнесены к положительному классу.

False positives (FP) - это количество объектов, которые были неправильно отнесены к положительному классу (ложные срабатывания).

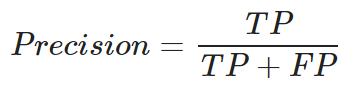
True negatives (TN) - это количество объектов, которые были правильно отнесены к отрицательному классу.

False negatives (FN) - это количество объектов, которые были неправильно отнесены к отрицательному классу (ложные пропуски).

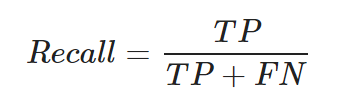
Матрица неточностей имеет следующий вид:

|  |  |  |
| --- | --- | --- |
|  | Оценка эксперта | |
| Оценка классификатора | Положительная | Отрицательная |
| Положительная | TP | FP |
| Отрицательная | FN | TN |

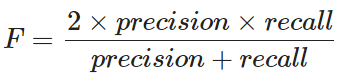
*precision (точность)* – показывает долю верно классифицированных объектов среди всех объектов, которые к этому классу отнес классификатор.



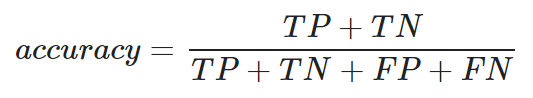
*recall (полнота)* – показывает отношение верно классифицированных объектов класса к общему числу элементов этого класса.



*f1-score (F1-мера) –* гармоническое среднее, между *precision* и *recall*. Является наиболее точным усреднением.



accuracy – показывает долю правильных классификаций.



В случае многоклассовой классификации, как и для бинарной, строится матрица неточностей и на основе неё вычисляются метрики, однако в этом случае чаще используются средние значения для отображения более полной картины.

macro avg = – среднее арифметическое

weighted avg = – среднее взвешенное

## **1.4 Выводы по главе:**

**По ходу главы сформулировали постановку задачи классификации, рассмотрели основные термины и особенности текстовой классификации. Подробно описали Tf-Idf взвешивание, метрики качества классификации, такие как: precision, recall, F1-score и accuracy.**

# **Глава 2. Методы классификации**

### **2.1 Логистическая регрессия**

**Логистическая регрессия – это алгоритм машинного обучения, который используется для классификации. Он относится к семейству линейных моделей и основан на принципе максимизации правдоподобия.**

**Алгоритм логистической регрессии можно представить следующим образом:**

1. **Подготовка данных:**

**Подготавливаем данные, представляем их в числовом формате, нормализуем их при необходимости, и разбиваем на обучающую и тестовую выборки.**

1. **Определение гиперпараметров:**

**Определяем гиперпараметры модели, учитывая то, что в работе реализация метода была с использованием библиотеки Scikit-learn:**

***- penalty* – тип регуляризации для контроля переобучения. Может принимать значения "l1", "l2", "elasticnet" и "none". По умолчанию используется "l2".**

***- C* – обратная сила регуляризации. Большие значения C указывают на меньшую регуляризацию, что может привести к переобучению модели, а маленькие значения C указывают на большую регуляризацию, что может привести к недообучению. Значение по умолчанию для C равно 1.0.**

***- max\_iter* – максимальное число итераций для оптимизации функции потерь. Значение по умолчанию равно 100.**

***- solver* – алгоритм для оптимизации функции потерь. Может принимать значения "newton-cg", "lbfgs", "liblinear", "sag", и "saga". По умолчанию используется "lbfgs" –** алгоритм Бройдена-Флетчера**.**

***- multi\_class* – способ обработки многоклассовых проблем. Может принимать значения "ovr" (один против всех) и "multinomial" (многовариантная логистическая регрессия). По умолчанию используется "ovr".**

***- class\_weight* – веса классов для контроля дисбаланса классов. Может быть задано значение "balanced", чтобы автоматически вычислить веса, основываясь на доле объектов в каждом классе.**

1. **Инициализация весов:**

**Инициализируем веса случайными значениями из нормального распределения.**

1. **Обучение модели:**

**На каждой эпохе для каждого элемента мини-батча, вычисляем значение гипотезы h(x) с помощью линейной функции:**

**где z – это линейная комбинация признаков и весов:**

**z =**

**где - константный признак, а - веса, которые мы ищем.**

**Далее, вычисляем функцию потерь (ошибку) для каждого элемента мини-батча:**

**где - размер мини-батча, - истинный класс -го элемента, а - предсказанный класс.**

**Используя градиентный спуск, обновляем веса на каждой эпохе:**

**где - коэффициент обучения.**

1. **Прогнозирование класса:**

**Для нового наблюдения вычисляем значение функции с помощью найденных весов и классифицируем его.**

1. **Оценка качества модели:**

**Вычисляем метрики качества модели, такие как точность, полнота, F1-мера, ROC-кривая и AUC-ROC, на тестовой выборке.**

1. **Тонкая настройка модели:**

**Изменяем гиперпараметры и повторяя шаги 3-6, можно улучшить качество модели.**

1. **Использование модели:**

**Используем обученную модель для прогнозирования классов новых наблюдений.**

### **2.2 К-ближайших соседей**

**Алгоритм k-ближайших соседей (k-NN) является методом классификации и регрессии. При классификации объекта алгоритм находит k ближайших к нему объектов из обучающей выборки и присваивает объекту тот класс, который наиболее часто встречается среди этих k ближайших соседей.**

**Алгоритм k-NN:**

1. **Пусть дана обучающая выборка , где - вектор признаков объекта, - его класс.**
2. **Настройка параметров модели:**

***- n\_neighbors* –количество соседей, учитываемых при классификации.**

***- weights* – определяет веса, используемые при прогнозировании меток классов. Варианты весов могут быть 'uniform' (все соседи имеют одинаковый вес) или 'distance' (веса обратно пропорциональны расстоянию до соседей).**

***algorithm*: определяет алгоритм, используемый для нахождения ближайших соседей. Варианты могут быть 'brute' (перебор всех точек в обучающем наборе данных), 'kd\_tree' (использует KD-дерево для эффективного поиска ближайших соседей), 'ball\_tree' (использует сферические разбиения для поиска ближайших соседей) или 'auto' (выбирает наиболее эффективный алгоритм на основе количества точек и размерности данных).**

***- leaf\_size* – размер листьев дерева, используемого при использовании kd\_tree или ball\_tree. Этот параметр влияет на скорость поиска ближайших соседей.**

***- p* – параметр Minkowski метрики, используемой для измерения расстояния между точками. По умолчанию p=2, что соответствует евклидовой метрике. Другие возможные значения могут быть 1 (манхэттенская метрика) или любое другое положительное число.**

***- metric*: функция расстояния, используемая для вычисления расстояния между точками. Варианты могут быть 'euclidean', 'manhattan', 'minkowski' или любая другая пользовательская функция расстояния.**

1. **Для нового объекта *z* необходимо вычислить расстояние до всех объектов обучающей выборки:**

**где – выбранная метрика расстояния между объектами. Находим k ближайших соседей нового документа среди документов обучающей выборки X по расстояниям, вычисленным на предыдущем шаге.**

1. **Присваиваем новому документу тот класс, который наиболее часто встречается среди классов k ближайших соседей.**
2. **Оценка качества модели:**

**Вычисляем метрики качества модели, такие как точность, полнота, F1-мера, accuracy.**

1. **Тонкая настройка модели:**

**Изменяя гиперпараметры и повторяя шаги 2-5, в ряде случаев можно улучшить качество модели.**

1. **Использование модели:**

**Используем обученную модель для прогнозирования классов новых наблюдений.**

**Для регрессии, вместо присвоения новому объекту того класса, который наиболее часто встречается среди k ближайших соседей, в качестве ответа используется среднее значение целевой переменной для k ближайших соседей.**

**Алгоритм k-NN прост в реализации и может быть использован для решения различных задач классификации и регрессии. Однако, он имеет несколько недостатков, таких как чувствительность к выбору метрики расстояния и числа ближайших соседей, а также высокую вычислительную сложность при работе с большими объемами данных.**

**Рассмотрим метрики расстояния, наиболее часто используемые для k-NN:**

**Евклидово расстояние:**

**Манхэттенское расстояние:**

**Косинусное расстояние:**

**Расстояние Чебышева:**

**Метрика Минковского:**

**Здесь и - вектора признаков объектов, n - количество признаков, а p - параметр метрики.**

### **2.3 Деревья решений**

Деревья решений — это алгоритм машинного обучения, который используется в задачах классификации и регрессии. Он представляет собой древовидную структуру, в которой каждый узел представляет собой проверку значения одного из признаков объекта, а каждое ребро, исходящее из узла, соответствует одному из возможных значений этого признака. Листья дерева содержат метки классов или числовые значения для задачи регрессии.

Алгоритм построения дерева решений:

1. Настройка параметров:

*- criterion* **–** функция для измерения качества разбиения. Поддерживаются критерии “gini” для неопределенности Джини, “log\_loss” и “entropy” для прироста информации Шеннона.

*- splitter* **–** стратегия для выбора разбиения в каждом узле. Поддерживаются стратегии “best” для выбора лучшего разбиения и “random” для выбора лучшего случайного разбиения.

*- max\_depth* **–** максимальная глубина дерева. Если None, то узлы расширяются до тех пор, пока все листья не будут чистыми или пока все листья не будут содержать меньше min\_samples\_split образцов.

*- min\_samples\_split* **–** минимальное количество образцов, необходимое для разбиения внутреннего узла. Если int, то рассматривается min\_sample*s\_split* как минимальное число. Если float, то *min\_samples\_split*: это доля и ceil (min\_samples\_split \* n\_samples) - минимальное количество образцов для каждого разбиения.

*- min\_samples\_leaf* **––** минимальное количество образцов, необходимое для того, чтобы узел был листовым. Точка разбиения на любой глубине будет рассматриваться только в том случае, если она оставляет не менее min\_samples\_leaf обучающих образцов в каждой из левой и правой ветвей. Это может сгладить модель, особенно в регрессии.

*- min\_weight\_fraction\_leaf* **–** минимальная взвешенная доля суммарного веса (всех входных образцов), необходимая для того, чтобы узел был листовым. Образцы имеют равный вес, когда sample\_weight не предоставлен.

*- max\_features* **–** количество признаков, которые рассматриваются при поиске лучшего разбиения.

* Если int, то рассматривается max\_features признаков при каждом разделении.
* Если float, то max\_features является долей и рассматривается max(1, int(max\_features \* n\_features)) признаков при каждом разделении.
* Если значение равно "sqrt", то рассматривается sqrt(n\_features) признаков при каждом разделении.
* Если значение равно "log2", то рассматривается log2(n\_features) признаков при каждом разделении.
* Если значение равно None, то рассматриваются все признаки.

*-random\_state* **–** параметр для контроля случайности процесса обучения. Если int, то random\_state является начальным числом генератора случайных чисел. Если RandomState instance или None, то генератор случайных чисел используется по умолчанию.

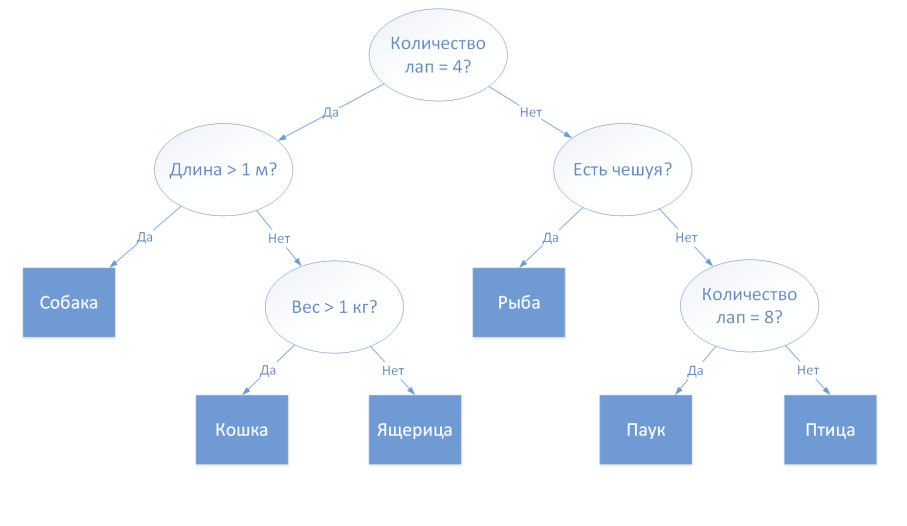
*- max\_leaf\_nodes* **–** максимальное количество листьев дерева. Если None, то число листьев не ограничено.

1. Выбор признака для разбиения данных: на первом шаге выбирается признак, который наилучшим образом разделяет выборку на подмножества (обычно используется мера информативности, такая как энтропия Шеннона или критерий Джини).

– критерий Джинни

– энтропия Шенона

1. Разбиение выборки по значению выбранного признака: данные делятся на подмножества, соответствующие значениям выбранного признака. Для каждого значения признака создается свой дочерний узел в дереве.
2. Рекурсивное применение шагов 2 и 3 для каждого дочернего узла: процесс разбиения данных продолжается рекурсивно, пока не будет достигнут критерий остановки, например, минимальное количество объектов в листьях дерева или достижение заданной глубины дерева.
3. Назначение меток классов или числовых значений для листьев: в каждый лист дерева записывается метка класса (для задач классификации) или числовое значение (для задач регрессии), соответствующее объектам из данного листа.



1. **Оценка качества модели:**

**Вычисляем метрики качества модели, такие как точность, полнота, F1-мера, accuracy.**

1. **Тонкая настройка модели:**

**Изменяем гиперпараметры и повторяя шаги 3-6, можно улучшить качество модели.**

1. **Использование модели:**

**Используем обученную модель для прогнозирования классов новых наблюдений.**

**2.4 Случайный лес**

**Алгоритм случайного леса – это алгоритм машинного обучения, который использует ансамбль решающих деревьев для решения задач классификации и регрессии.**

**Алгоритм случайного леса:**

1. **Выбор параметров случайного леса:**

***- n\_estimators* – количество деревьев в лесу. По умолчанию - 100.**

***- criterion* – критерий, используемый в деревьях для оценки качества разбиения. Доступны два варианта: "gini" и "entropy". По умолчанию используется "gini".**

***- max\_depth* – максимальная глубина деревьев. Если значение равно None, то узлы будут разбиты до тех пор, пока каждый лист содержит не менее min\_samples\_split элементов или пока все листья не будут содержать одинаковые метки. По умолчанию - None.**

***- min\_samples\_split* – минимальное количество элементов, необходимых для разделения узла. По умолчанию - 2.**

***- min\_samples\_leaf* – минимальное количество элементов в листе. По умолчанию - 1.**

***- max\_features* –** количество признаков, которые рассматриваются при поиске лучшего разбиения.

* Если int, то рассматривается max\_features признаков при каждом разделении.
* Если float, то max\_features является долей и рассматривается max(1, int(max\_features \* n\_features)) признаков при каждом разделении.
* Если значение равно "sqrt", то рассматривается sqrt(n\_features) признаков при каждом разделении.
* Если значение равно "log2", то рассматривается log2(n\_features) признаков при каждом разделении.
* Если значение равно None, то рассматриваются все признаки.

***- bootstrap* – если True, то используется метод бутстрэпа для построения деревьев, это значит, что на каждом шаге будет случайно выбираться часть выборки, иначе будет использоваться весь набор данных. По умолчанию - True.**

***- oob\_score* – если True, то используется out-of-bag (oob) оценка для каждого дерева. OOB-оценка (Out-of-Bag) – это метод оценки качества алгоритма случайного леса, основанный на использовании данных, которые не попали в выборку для построения данного дерева. OOB-оценка позволяет оценить качество случайного леса без необходимости использовать отдельную выборку для валидации модели. Это позволяет более эффективно использовать имеющиеся данные и получать более точные оценки качества модели. По умолчанию - False.**

1. **Выберите N случайных записей из набора данных.**
2. **Постройте дерево решений на основе этих N записей. Подробнее построение дерева рассмотрено в параграфе 2.3.**
3. **Выберите количество деревьев в вашем алгоритме и повторите шаги 1 и 2.**
4. **В случае регрессионной задачи для новой записи каждое дерево в лесу предсказывает значение Y (выход). Конечное значение можно вычислить, взяв среднее значение всех значений, предсказанных всеми деревьями в лесу. В случае классификации выбирается класс, которому отдало предпочтение большинство деревьев.**
5. **Оценка качества модели:**

**Вычисляем метрики качества модели, такие как точность, полнота, F1-мера, accuracy.**

1. **Тонкая настройка модели:**

**Изменяем гиперпараметры и повторяя шаги 2-5, можно улучшить качество модели.**

1. **Использование модели:**

**Используем обученную модель для прогнозирования классов новых наблюдений.**

**Алгоритм случайного леса имеет ряд преимуществ, таких как:**

* **Высокая точность и устойчивость к переобучению за счет использования множества независимых деревьев.**
* **Способность обрабатывать большие и высокоразмерные данные за счет выбора случайных подпространств признаков.**
* **Возможность оценивать важность признаков за счет анализа их вклада в уменьшение ошибки или неопределенности.**
* **Простота реализации и интерпретации за счет использования стандартных методов построения и оценки деревьев.**

**Алгоритм случайного леса также имеет некоторые недостатки, такие как:**

* **Большой объем памяти и вычислительных ресурсов, необходимых для хранения и обучения большого числа деревьев.**
* **Сложность объяснения логики принятия решений за счет использования ансамбля деревьев вместо одного дерева.**
* **Низкая эффективность для задач с нелинейными и сложными зависимостями между признаками и целевой переменной за счет использования пороговых правил разбиения узлов.**

## **2.5 Выводы по главе**

**В главе подробно рассмотрены с описанием алгоритма методы классификации, используемые в работе: логистическая регрессия, к-ближайших соседей, деревья решений и случайный лес. Подробно описаны настраиваемые гиперпараметры, используемые в библиотеке scikit-learn.**

**Глава 3. Бинарная классификация**

**В данной главе мы рассмотрим классификацию текстовых коллекций на примере бинарной выборки научных текстов. Будет произведен сравнительный анализ качества классификации в зависимости от системы признаков и сравнение различных методов классификации.**

**2.1 Описание выборки**

**Для исследования используется выборка текстов с сайта elibrary.ru, состоящая из библиографических описаний (БО) текстов различной тематики. В выборке содержится 3267 элементов, каждый из которых** относится к классу **«Интеллектуальный анализ данных» (ИАД) или «не ИАД». БО состоят из названия, аннотации, года выпуска, автора и ключевых слов. В данной работе для классификации используются (и сравниваются):**

**- все БО (название, аннотации и ключевые слова),**

**- только названия.**

**- только ключевые слова**

**Предварительно можно сказать, что классы сбалансированы**

Количество текстов по теме ИАД 1528

Количество текстов по теме не ИАД 1739

**Пример БО:**

**Название:**

**ПЕРСПЕКТИВЫ ВНЕДРЕНИЯ ТЕХНОЛОГИЙ DATA MINING В ТАМОЖЕННУЮ ДЕЯТЕЛЬНОСТЬ**

**ЖУРНАЛ:**

**АКАДЕМИЧЕСКИЙ ВЕСТНИК РОСТОВСКОГО ФИЛИАЛА РОССИЙСКОЙ ТАМОЖЕННОЙ АКАДЕМИИ**

**Аннотация:**

**В статье проведен анализ перспективных направлений внедрения технологий Data Mining в деятельность таможенных органов. Рассмотрены классификационные методы машинного обучения с учителем и без учителя, применение которых может автоматизировать решение сложных задач по отнесению поставок товаров к рисковым или выявлению потенциальных рисков. Особое внимание уделено кластерному анализу и программным платформам, которые поддерживают его реализацию.**

**Ключевые слова:**

**ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ МАШИННОЕ ОБУЧЕНИЕ ТАМОЖЕННЫЕ РИСКИ ТАМОЖЕННАЯ ДЕЯТЕЛЬНОСТЬ DATA MINING MACHINE LEARNING CUSTOMS RISK CUSTOMS ACTIVITY**

**Автор: Кудрявцев О. Е.**

**2.2 Предобработка и визуализация.**

Предобработка всех признаковых пространств состояла из следующих пунктов

* приведение к нижнему регистру и токенизация
* удаление стоп-слов
* лемматизация
* TF-IDF взвешивание

В ходе программного и визуального изучения выборки, выявилось наличие небольшого процента англоязычных текстов, так же некоторые БО содержали в себе 2 языка, было принято решение не удалять из выборки, а обработать вместе с русскоязычными, поэтому пришлось расширить словарь стоп-слов и использовать 2 словаря для лемматизации.

Словарь терминов, составленный по БО (названия, аннотации, ключевые слова), имеет размерность 25124, словарь, составленный по названиям 6394 а, по ключевым словам, 12186.

Таким образом, словарь терминов из названий почти в 4 раза меньше, чем словарь терминов из БО.

**Проводим классификацию с помощью логистической регрессии – для чего из описания не ясно – хотим показать, что можно отбросить редко встречающиеся слова? Пиши понятно**

Сравнение результатов классификации по названиям

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | С удалением малозначимых терминов | | | | Без удаления малозначимых терминов | | | |
|  | precision | recall | f1-score | support | precision | recall | f1-score | support |
| False | 0.85 | 0.93 | 0.89 | 1262 | 0.83 | 0.96 | 0.89 | 1262 |
| True | 0.91 | 0.81 | 0.86 | 1084 | 0.94 | 0.77 | 0.84 | 1084 |
| accuracy |  |  | 0.88 | 2346 |  |  | 0.87 | 2346 |
| macro avg | 0.88 | 0.87 | 0.88 | 2346 | 0.88 | 0.86 | 0.87 | 2346 |
| weighted avg | 0.88 | 0.88 | 0.88 | 2346 | 0.88 | 0.87 | 0.87 | 2346 |

В таблице использованы следующие обозначения:……….

Оптимальные параметры с удалением малозначимых терминов

'C': 100'max\_iter': 100, 'penalty': 'l2', 'solver': 'liblinear'; 'class\_weight': None

Оптимальные параметры без удаления малозначимых терминов

'penalty': 'l2'; 'C': 100; 'max\_iter': 100; 'solver': "newton-cg"; 'class\_weight': None

Сравнение результатов классификации по библиографическому описанию

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | С удалением малозначимых терминов | | | | Без удаления малозначимых терминов | | | |
|  | precision | recall | f1-score | support | precision | recall | f1-score | support |
| False | 0.88 | 0.97 | 0.92 | 1262 | 0.83 | 0.98 | 0.90 | 1262 |
| True | 0.96 | 0.85 | 0.90 | 1084 | 0.98 | 0.77 | 0.86 | 1084 |
| accuracy |  |  | 0.91 | 2346 |  |  | 0.88 | 2346 |
| macro avg | 0.92 | 0.91 | 0.91 | 2346 | 0.90 | 0.87 | 0.88 | 2346 |
| weighted avg | 0.92 | 0.91 | 0.91 | 2346 | 0.90 | 0.88 | 0.88 | 2346 |

Оптимальные параметры с удалением малозначимых терминов

'penalty': 'l2' – регуляризация Тихонова

'C': 100

'max\_iter': 100

'solver': 'newton-cg'

'class\_weight': None

Оптимальные параметры без удаления малозначимых терминов

'penalty': 'l2'

'C': (100,)

'max\_iter': 100

'solver': 'liblinear'

'class\_weight': None

**Вывод:**

С помощью алгоритма логистической регрессии, получили достаточно высокое качество ~~точные результаты~~ классификации. Можно сказать, что уменьшение размерности признакового пространства не привело к существенному ~~дало~~ ухудшению классификации, в частности значение f1-меры отличается……….. ~~и можно продолжать исследование.~~

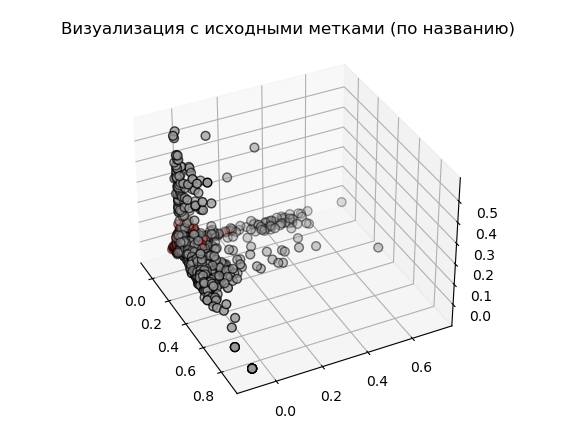
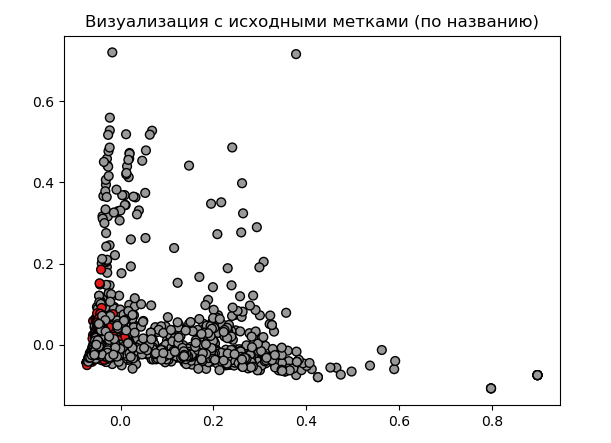
## **2.3 Визуализация**

Почему здесь визуализация? Ведь мы изучаем выборку до классификации, хотим понять, как она расположена в признаковом пространстве. В п.2.1?

Визуализация с исходными метками

Здесь в дальнейшем, серые точки – ИАД, красные точки – не ИАД

Для визуализации требуется сократить размерность данных, для этого в работе используется ***метод главных компонент.***

****

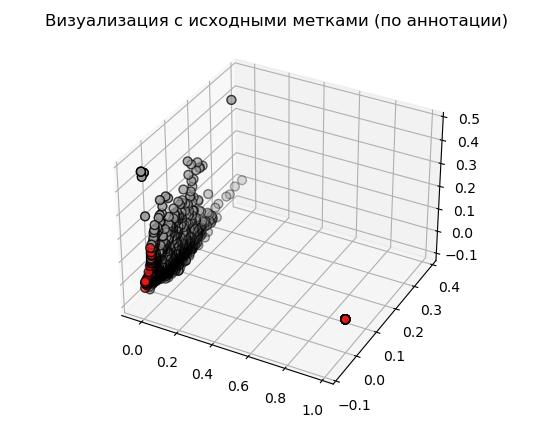
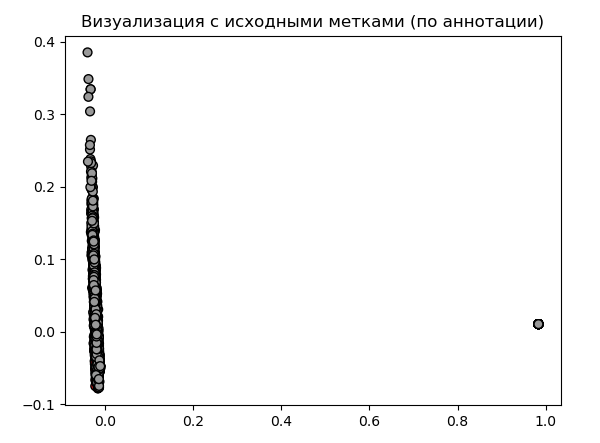
****

Рис 1. Визуализация с исходными метками.

Главные компоненты получились так себе. Делал ли ты t-SNE?

Делал, получилось не очень

## **2.4 Классификация**

Таблицы-графики не могут идти друг за другом без пояснений. Провел исследование – поясни, что получил, затем переходи к другому эксперименту. Обязательно говорим о значениях настраиваемых параметров у методов.

Это «прошлое» замечание – ничего не изменилось! Улучшай

**Логистическая регрессия**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | По названиям | | | | По библиографическому описанию | | | |
|  | precision | recall | f1-score | support | precision | recall | f1-score | support |
| False | 0.85 | 0.93 | 0.89 | 1262 | 0.88 | 0.97 | 0.92 | 1262 |
| True | 0.91 | 0.81 | 0.86 | 1084 | 0.96 | 0.85 | 0.90 | 1084 |
| accuracy |  |  | 0.88 | 2346 |  |  | 0.91 | 2346 |
| macro avg | 0.88 | 0.87 | 0.88 | 2346 | 0.92 | 0.91 | 0.91 | 2346 |
| weighted avg | 0.88 | 0.88 | 0.88 | 2346 | 0.92 | 0.91 | 0.91 | 2346 |

Оптимальные параметры с удалением малозначимых терминов

'C': 100, 'max\_iter': 100, 'penalty': 'l2', 'solver': 'liblinear'; 'class\_weight': None

Оптимальные параметры с удалением малозначимых терминов

'penalty': 'l2'

'C': (100,),

'max\_iter': 100

'solver': 'newton-cg'

'class\_weight': None

**К-ближайших соседей**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | По названиям | | | | По библиографическому описанию | | | |
|  | precision | recall | f1-score | support | precision | recall | f1-score | support |
| False | 0.85 | 0.89 | 0.87 | 1262 | 0.94 | 0.87 | 0.90 | 1262 |
| True | 0.86 | 0.81 | 0.84 | 1084 | 0.86 | 0.93 | 0.90 | 1084 |
| accuracy |  |  | 0.85 | 2346 |  |  | 0.90 | 2346 |
| macro avg | 0.86 | 0.85 | 0.85 | 2346 | 0.90 | 0.90 | 0.90 | 2346 |
| weighted avg | 0.86 | 0.85 | 0.85 | 2346 | 0.90 | 0.90 | 0.90 | 2346 |

Оптимальные параметры по названиям

'metric': 'cosine', 'n\_neighbors': 10, 'weights': 'uniform' 'algorithm': 'brute' 'p': 1

Оптимальные параметры по аннотациям

'metric': 'cosine', 'n\_neighbors': 10, 'weights': 'uniform' 'algorithm': 'brute' 'p': 1

**Деревья решений**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | По названиям | | | | По библиографическому описанию | | | |
|  | precision | recall | f1-score | support | precision | recall | f1-score | support |
| False | 0.80 | 0.89 | 0.84 | 1262 | 0.84 | 0.90 | 0.87 | 1262 |
| True | 0.85 | 0.75 | 0.79 | 1084 | 0.87 | 0.81 | 0.84 | 1084 |
| accuracy |  |  | 0.82 | 2346 |  |  | 0.86 | 2346 |
| macro avg | 0.83 | 0.82 | 0.82 | 2346 | 0.86 | 0.85 | 0.85 | 2346 |
| weighted avg | 0.83 | 0.82 | 0.82 | 2346 | 0.86 | 0.86 | 0.86 | 2346 |

Оптимальные параметры по названиям

'criterion': 'entropy',

'max\_depth': 100,

'max\_features': 0.5,

'min\_samples\_leaf': 5,

'min\_samples\_split': 2,

'random\_state': 0,

'splitter': 'random'

Оптимальные параметры по аннотациям

'criterion': 'gini',

'max\_depth': 10,

'max\_features': None,

'min\_samples\_leaf': 10,

'min\_samples\_split': 10,

'random\_state': 0,

'splitter': 'best'**Случайный лес**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | По названиям | | | | По библиографическому описанию | | | |
|  | precision | recall | f1-score | support | precision | recall | f1-score | support |
| False | 0.82 | 0.96 | 0.88 | 1262 | 0.89 | 0.96 | 0.92 | 1299 |
| True | 0.94 | 0.76 | 0.84 | 1084 | 0.95 | 0.85 | 0.90 | 1095 |
| accuracy |  |  | 0.87 | 2346 |  |  | 0.91 | 2394 |
| macro avg | 0.88 | 0.86 | 0.86 | 2346 | 0.92 | 0.91 | 0.91 | 2394 |
| weighted avg | 0.88 | 0.87 | 0.86 | 2346 | 0.92 | 0.91 | 0.91 | 2394 |

Оптимальные параметры по названиям

'bootstrap': True,

'criterion': 'gini',

'max\_depth': 200,

'max\_features': log2,

'min\_samples\_leaf': 1,

'min\_samples\_split': 2,

'n\_estimators': 400,

'oob\_score': True

Оптимальные параметры по аннотациям

'bootstrap': False,

'criterion': 'gini',

'max\_depth': 100,

'max\_features': 'sqrt',

'min\_samples\_leaf': 1,

'min\_samples\_split': 2,

'n\_estimators': 300,

'oob\_score': False

**Сравнение методов:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | По названиям | | По библиографическому описанию | |
| Метрика  Метод | Accuracy | Macro avg F1 | Accuracy | Macro avg F1 |
| Логистическая регрессия | 0.88 | 0.88 | 0.91 | 0.91 |
| КБС | 0.85 | 0.85 | 0.90 | 0.90 |
| Дерево решений | 0.82 | 0.82 | 0.86 | 0.85 |
| Случайный лес | 0.87 | 0.86 | 0.91 | 0.91 |

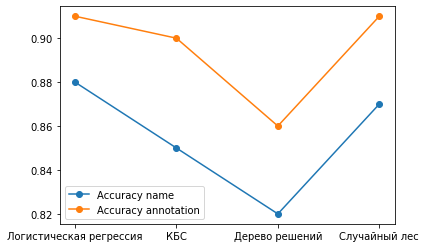
****

Рис 2. Сравнение accuracy по различным методам

**На рисунке оранжевый цвет – правильность на аннотациях, раньше в главе писалось, что сравниваем БО и названия. Аннотации это не БО. Как на самом деле?**

**Выводы по главе:**

В ходе исследований лучшего результата удалось добиться с помощью алгоритма логистической регрессии, наравне с алгоритмом случайного леса. В ходе исследования были получены и описаны параметры для каждого из методов… (пока ещё не описаны)

Метод К-ближайших соседей, показал наибольшую чувствительность к настройке параметров, а также к выбору признакового пространства. Наиболее оптимальным методом для текущий задачи считаю логистическую регрессию.

Точность методов определялась на основе метрик, описанных в пункте 1.3 «метрики качества классификации», наиболее показательными метриками считаются *f1-метрика* и *accuracy*.

Целью главы была проверка возможности сокращения признакового пространства и возможности классификации текстовых данных при небольшой размерности признаков. Мы показали, что классификация на небольшом признаковом пространстве возможна и имеет свои плюсы, такие как:

* требуется меньше вычислительных мощностей
* простота сбора и работы с данными небольшой размерности
* возможность визуально оценить элементы выборки
* возможность работы с данными, где нельзя собрать пространство большой размерности (сообщения в мессенджерах, социальные сети, почта и др.)

Благодаря сокращению размерности признакового пространства удалось снизить время обучения алгоритмов примерно в 4-5 раз.

**Глава 4. Мультиклассовая классификация**

Подход к многоклассовой классификации несколько отличается от бинарной, такая классификация имеет свою специфику ~~проблемы~~:

* Наличие большого количества классов. С увеличением количества классов возрастает сложность ~~задачи~~ классификации и требуются более крупные выборки.
* Несбалансированность классов. Если размеры классов существенно отличаются друг от друга, то алгоритм классификации может давать смещенные результаты в пользу больших классов.
* Неоднозначность классификации. Некоторые объекты могут быть неоднозначно отнесены к нескольким классам. Что имеется в виду – «мягкая» классификация?
* Необходимость выбора оптимального способа голосования «каждый-против каждого» или «каждый-против всех»
* Более сложная оценка качества классификации. В многоклассовой классификации необходимо использовать специальные методы для оценки качества классификации, такие как микро- и макро-усреднение метрик, которые учитывают не только точность классификации каждого класса, но и сбалансированность классов.
* ~~Недостаточное количество данных для обучения. Если у нас недостаточно данных для обучения многоклассового классификатора, то это может привести к переобучению или недообучению~~. Включил в первый пункт
* Наличие скрытых зависимостей между классами. В некоторых случаях классы могут быть связаны между собой скрытыми зависимостями, что усложняет задачу классификации. Например, если мы классифицируем объекты по виду животных, то классы "кошки" и "собаки" могут быть связаны между собой, потому что они относятся к одной категории "домашних животных". Можно объединить с «Неоднозначностью классификации»

## **3.1 Описание выборки**

В этой главе мы рассмотрим выборку из 10 классов, сформированную А.В. Кононенко и состоящую из документов, которые извлечены из …………научно-технической литературы сформированную студентом кафедры Кононенко Александром Вячеславовичем.

Классы можно считать сбалансированными, число элементов и наименования указаны ниже

1. computer vision - 1797
2. text mining - 1933
3. control systems - 1452
4. cyber security - 2965
5. information retrieval - 1944
6. fuzzy - 1776
7. neural nets - 1840
8. database - 1936
9. robotic - 1550
10. expert – 1834

Визуализация ???

## **3.2 Классификация**

Нужно дать внятные комментарии по проведенным исследованиям и расширить (переписать) выводы.

В качестве предобработки текста использовали лемматизацию и TF-IDF взвешивание. Для сравнение приведем таблиц метрик классификации различными методами при использовании разных признаковых пространств.

Опять сплошные таблицы – описываем нормально.

**Далее рассмотрим признаковое пространство аннотации**

**Логистическая регрессия**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| computer vision | 0.83 | 0.87 | 0.85 | 1245 |
| control systems | 0.77 | 0.76 | 0.77 | 1031 |
| cyber security | 0.88 | 0.90 | 0.89 | 2015 |
| database | 0.85 | 0.85 | 0.85 | 1330 |
| expert | 0.89 | 0.89 | 0.89 | 1332 |
| fuzzy | 0.98 | 0.95 | 0.96 | 1232 |
| information retrieval | 0.85 | 0.77 | 0.81 | 1398 |
| neural nets | 0.83 | 0.87 | 0.85 | 1313 |
| robotic | 0.79 | 0.77 | 0.78 | 1068 |
| text mining | 0.82 | 0.85 | 0.83 | 1355 |
| accuracy |  |  | 0.85 | 13319 |
| macro avg | 0.85 | 0.85 | 0.85 | 13319 |
| weighted avg | 0.85 | 0.85 | 0.85 | 13319 |

{'clf\_\_C': 1200,

'clf\_\_class\_weight': 'balanced',

'clf\_\_max\_iter': 80,

'clf\_\_multi\_class': 'ovr',

'clf\_\_penalty': 'l1',

'clf\_\_solver': 'liblinear'}

**К-ближайших соседей**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| computer vision | 0.50 | 0.78 | 0.61 | 1245 |
| control systems | 0.69 | 0.51 | 0.59 | 1031 |
| cyber security | 0.59 | 0.95 | 0.73 | 2015 |
| database | 0.90 | 0.18 | 0.30 | 1330 |
| expert | 0.85 | 0.30 | 0.44 | 1332 |
| fuzzy | 0.91 | 0.77 | 0.83 | 1232 |
| information retrieval | 0.75 | 0.66 | 0.70 | 1398 |
| neural nets | 0.72 | 0.66 | 0.69 | 1313 |
| robotic | 0.71 | 0.65 | 0.68 | 1068 |
| text mining | 0.55 | 0.88 | 0.68 | 1355 |
| accuracy |  |  | 0.65 | 13319 |
| macro avg | 0.72 | 0.63 | 0.62 | 13319 |
| weighted avg | 0.71 | 0.65 | 0.63 | 13319 |

{'clf\_\_algorithm': 'brute',

'clf\_\_metric': 'euclidean',

'clf\_\_n\_neighbors': 500,

'clf\_\_weights': 'distance'}

**Случайный лес**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| computer vision | 0.94 | 0.93 | 0.94 | 1245 |
| control systems | 0.89 | 0.83 | 0.86 | 1031 |
| cyber security | 0.92 | 0.95 | 0.94 | 2015 |
| database | 0.91 | 0.94 | 0.92 | 1330 |
| expert | 0.93 | 0.94 | 0.93 | 1332 |
| fuzzy | 0.98 | 0.98 | 0.98 | 1232 |
| information retrieval | 0.89 | 0.86 | 0.88 | 1398 |
| neural nets | 0.96 | 0.93 | 0.94 | 1313 |
| robotic | 0.91 | 0.92 | 0.92 | 1068 |
| text mining | 0.90 | 0.94 | 0.92 | 1355 |
| accuracy |  |  | 0.92 | 13319 |
| macro avg | 0.92 | 0.92 | 0.92 | 13319 |
| weighted avg | 0.92 | 0.92 | 0.92 | 13319 |

{'clf\_\_bootstrap': False,

'clf\_\_criterion': 'gini',

'clf\_\_max\_depth': 150,

'clf\_\_max\_features': 0.3,

'clf\_\_min\_samples\_leaf': 1,

'clf\_\_min\_samples\_split': 20,

'clf\_\_n\_estimators': 400,

'clf\_\_oob\_score': False}

**Третьим признаковым пространством будет объединение названий, аннотаций и ключевых слов.**

**Логистическая регрессия**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| computer vision | 0.76 | 0.88 | 0.81 | 1245 |
| control systems | 0.77 | 0.76 | 0.77 | 1031 |
| cyber security | 0.91 | 0.90 | 0.90 | 2015 |
| database | 0.88 | 0.83 | 0.86 | 1330 |
| expert | 0.90 | 0.89 | 0.90 | 1332 |
| fuzzy | 0.99 | 0.95 | 0.97 | 1232 |
| information retrieval | 0.86 | 0.76 | 0.81 | 1398 |
| neural nets | 0.81 | 0.85 | 0.83 | 1313 |
| robotic | 0.81 | 0.79 | 0.80 | 1068 |
| text mining | 0.83 | 0.88 | 0.86 | 1355 |
| accuracy |  |  | 0.85 | 13319 |
| macro avg | 0.85 | 0.85 | 0.85 | 13319 |
| weighted avg | 0.86 | 0.85 | 0.85 | 13319 |

{'clf\_\_C': 1,

'clf\_\_class\_weight': 'balanced',

'clf\_\_max\_iter': 20,

'clf\_\_multi\_class': 'ovr',

'clf\_\_penalty': 'l1',

'clf\_\_solver': 'liblinear'}

**К-ближайших соседей**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| computer vision | 0.48 | 0.74 | 0.58 | 1245 |
| control systems | 0.64 | 0.52 | 0.57 | 1031 |
| cyber security | 0.62 | 0.93 | 0.75 | 2015 |
| database | 0.80 | 0.22 | 0.35 | 1330 |
| expert | 0.76 | 0.25 | 0.37 | 1332 |
| fuzzy | 0.83 | 0.77 | 0.80 | 1232 |
| information retrieval | 0.73 | 0.69 | 0.71 | 1398 |
| neural nets | 0.66 | 0.60 | 0.63 | 1313 |
| robotic | 0.71 | 0.66 | 0.68 | 1068 |
| text mining | 0.56 | 0.85 | 0.68 | 1355 |
| accuracy |  |  | 0.64 | 13319 |
| macro avg | 0.68 | 0.62 | 0.61 | 13319 |
| weighted avg | 0.68 | 0.64 | 0.62 | 13319 |

{'clf\_\_algorithm': 'brute',

'clf\_\_metric': 'cosine',

'clf\_\_n\_neighbors': 200,

'clf\_\_weights': 'distance'}

**Случайный лес**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| computer vision | 0.89 | 0.92 | 0.90 | 1245 |
| control systems | 0.91 | 0.77 | 0.83 | 1031 |
| cyber security | 0.90 | 0.96 | 0.93 | 2015 |
| database | 0.91 | 0.92 | 0.91 | 1330 |
| expert | 0.93 | 0.94 | 0.93 | 1332 |
| fuzzy | 0.98 | 0.98 | 0.98 | 1232 |
| information retrieval | 0.86 | 0.84 | 0.85 | 1398 |
| neural nets | 0.92 | 0.85 | 0.88 | 1313 |
| robotic | 0.87 | 0.91 | 0.89 | 1068 |
| text mining | 0.90 | 0.94 | 0.92 | 1355 |
| accuracy |  |  | 0.91 | 13319 |
| macro avg | 0.91 | 0.90 | 0.90 | 13319 |
| weighted avg | 0.91 | 0.91 | 0.91 | 13319 |

{'clf\_\_bootstrap': False,

'clf\_\_criterion': 'gini',

'clf\_\_max\_depth': 150,

'clf\_\_max\_features': 0.1,

'clf\_\_min\_samples\_leaf': 1,

'clf\_\_min\_samples\_split': 2,

'clf\_\_n\_estimators': 1500,

'clf\_\_oob\_score': False}

Сравним классификацию по аннотациям и по всем признакам, везде лучшие результаты показал случайный лес, поэтому сморим на него

|  |  |  |
| --- | --- | --- |
|  | По аннотациям | По названиям аннотациям и ключевым словам |
| accuracy | 0.86 | 0.83 |
| macro avg f1-score | 0.86 | 0.82 |
| weighted avg f1-score | 0.86 | 0.83 |

**Вывод:** в ходе исследований были настроены лучшие гиперпараметры для каждого метода. Выяснилось, что одни названия дают неутешительный результат, при том, что на бинарной выборке различия в качестве почти не наблюдалось. Одновременное использование всех признаков не дало улучшения в сравнении с аннотациями, наоборот качество немного просело. Поэтому на данном этапе предпочтительнее использовать аннотации.

# **Описание используемого программного обеспечения**

**Заключение**

# **Список литературы (демо версия) Очень плохо - расширяй**

1. хабр
2. sklearn
3. stack overflow
4. Андрей Бурков. Инженерия машинного обучения. М.: ДМК-Пресс, 2022 — 306 с.
5. Андрей Бурков. Машинное обучение без лишних слов. СПб.: Питер, 2020 — 192 с.