МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт ИВТИ
Кафедра УИТ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

Направление 27.03.04 Управление в технических системах
(код и наименование)

Образовательная программа Системы и технические средства автоматизации и управления

Форма обучения очная
(очная/очно-заочная/заочная)

Тема: Сравнительный анализ качества классификации в зависимости от используемой системы признаков

Студент А-03-19 Козлов И.А.
группа подпись фамилия и инициалы

Руководитель ВКР д.т.н. профессор Толчеев В.О.
уч. степень должность подпись фамилия и инициалы

организация

«Работа допущена к защите»

Заведующий кафедрой д.т.н. доцент Бобряков А.В.
уч. степень звание подпись фамилия и инициалы

Дата

Москва, 2022

МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт ИВТИ
Кафедра УИТ

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
(бакалаврскую работу)

Направление 27.03.04 «Управление в технических системах»
(код и наименование)

Направленность (профиль) «Системы и технические средства автоматизации и управления»

Форма обучения Очная
(очная/очно-заочная/заочная)

Тема: *«Сравнительный анализ качества классификации в зависимости от используемой системы признаков»*

Студент	А-03-19			Козлов И.А.
	группа		подпись	фамилия и инициалы
Научный руководитель	Д.т.н	Профессор		Толчеев В.О.
Консультант				
	уч. степень	должность	подпись	фамилия и инициалы
Консультант				
	уч. степень	должность	подпись	фамилия и инициалы
Зав. кафедрой	Д.т.н	Доцент		Бобряков А.В.
	уч. степень	звание	подпись	фамилия и инициалы

Место выполнения работы Кафедра Управления и интеллектуальных технологий

СОДЕРЖАНИЕ РАЗДЕЛОВ ЗАДАНИЯ И ИСХОДНЫЕ ДАННЫЕ

1. Рассмотреть модели, методы и критерии качества текстовой классификации.
2. Описать выборку. Провести предварительную обработку текстов, составить матрицу «документ-термин», взвесить термины.
3. Провести бинарную классификацию для различных систем признаков.
4. На основе полученных результатов оценить влияние системы признаков на качество классификации. Выявить лучшие классификаторы.
5. Повторить исследования из п.2-4 для многоклассовой выборки (МКВ).
6. Сделать общие выводы по исследованию.
7. Описать используемое программное обеспечение.

ПЕРЕЧЕНЬ ГРАФИЧЕСКОГО МАТЕРИАЛА

Количество листов _____

Количество слайдов в презентации _____

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Андрей Бурков. Инженерия машинного обучения. М.: ДМК-Пресс, 2022 — 306 с.
2. Андрей Бурков. Машинное обучение без лишних слов. СПб.: Питер, 2020 — 192 с.

Примечания:

1. Задание брошюруется вместе с выпускной работой после титульного листа (страницы задания имеют номера 2, 3).
2. Отзыв руководителя, рецензия(и), отчет о проверке на объем заимствований и согласие студента на размещение работы в открытом доступе вкладываются в конверт (файловую папку) под обложкой работы.

Аннотация

Данная работа представляет собой исследование в области классификации текстовых данных. Основной задачей исследования в работе стоит изучение возможности сокращения признаковых пространств, в качестве примера используются два набора данных для бинарной и многоклассовой классификации, состоящие из научных документов.

В работе рассматриваются алгоритмы машинного обучения: логистическая регрессия, к-ближайших соседей, деревья решений и случайный лес, подробно описываются особенного работы с текстовыми данными, отдельно рассматриваются случаи бинарной и многоклассовой классификации. Рассматриваются метрики качества и используемое программное обеспечение (python).

Оглавление

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. 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

1.1 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 X, множество меток классов Y и обучающая выборка D состоящая из пар (x_i, y_i), где x_i - вектор признаков -го объекта, а y_i - метка класса для этого объекта.

Задача классификации заключается в нахождении алгоритма $f: X \to Y$, который по вектору признаков x предсказывает метку класса y. Для этого используется обучающая выборка D, на которой модель обучается находить закономерности в признаках объектов, связанных с их метками классов.

Функция f 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. Стоит 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:

$$TF - IDF = TF \cdot IDF$$

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.

$$TF(d,t) = \frac{\text{количество вхождений терма } t \text{ в документе } d}{\text{общее количество термов в документе } d}$$

$$IDF(t) = log\left(\frac{\text{общее количество документов в коллекции}}{\text{количество документов, содержащих терм } t}\right)$$

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.

$$Precision = \frac{TP}{TP + FP}$$

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

$$Recall = \frac{TP}{TP + FN}$$

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

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

accuracy – показывает долю правильных классификаций.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

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 = $\frac{1}{n}\sum_{i=1}^{n} x_i$ – среднее арифметическое

weighted avg = $\frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$ – 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

В ходе классификации одним из важнейших пунктов является настройка гиперпараметров алгоритмов, этому будет уделено отдельное внимание во всех пунктах работы.

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.

2. Определение гиперпараметров [3]

   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" лассо регуляризация добавляет штраф к модели, пропорциональный сумме абсолютных значений весовых коэффициентов модели.
   - "l2"(по умолчанию) гребневая регуляризация добавляет штраф к модели, пропорциональный сумме квадратов весовых коэффициентов модели.
   - "elasticnet" является комбинацией l1 и l2 используется для контроля переобучения и улучшения обобщающей способности модели.
   - "none".

- *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".

- 'liblinear' значение по умолчанию для бинарной классификации. Алгоритм использует оптимизацию на основе координатного спуска.
- 'newton-cg' алгоритм использует метод Ньютона-Рафсона для оптимизации функции потерь. Он поддерживает только L2 регуляризацию.
- 'lbfgs' алгоритм также использует метод Ньютона-Рафсона, но использует аппроксимацию Бройдена-Флетчера-Гольдфарба-Шанно. Поддерживает только L2 регуляризацию.
- 'sag' алгоритм стохастического усреднения градиентов. Использует стохастическую оптимизацию градиентного спуска и поддерживает L2 регуляризацию.
- 'saga' улучшенная версия алгоритма 'sag', которая поддерживает как L1, так и L2 регуляризацию.

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.

3. Инициализация весов

   Инициализируем веса случайными значениями из нормального распределения.

4. Обучение модели

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

$$h(x) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

где z – это линейная комбинация признаков и весов:

$z = \sum_{i=0}^{n} \theta_i x_i$

где $x_0 = 1$ - константный признак, а θ - веса, которые мы ищем.

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

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(h\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right)\log\left(1 - h\left(x^{(i)}\right)\right)\right]$$

где m - размер мини-батча, $y^{(i)}$ - истинный класс i-го элемента, а $h\left(x^{(i)}\right)$ - предсказанный класс.

Используя градиентный спуск, обновляем веса на каждой эпохе:

$$\theta_j = \theta_\text{j} - \alpha \frac{\partial}{(\partial \theta_\text{j})} J(\theta)$$

где α - коэффициент обучения.

5. Прогнозирование класса

Для нового наблюдения вычисляем значение функции $h(x)$с помощью найденных весов и классифицируем его.

$$y_{pred} = \begin{cases} 1, h(x) \geq 0.5 \\ 0, h(x) < 0.5 \end{cases}$$

6. Оценка качества модели:

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

7. Тонкая настройка модели

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

8. 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 $X = (x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, где x_i - вектор признаков объекта, y_i - его класс.
2. Настройка параметров модели [3]

   - *n_neighbors* – количество ближайших соседей, учитываемых при классификации.

   - *weights* – определяет веса, используемые при прогнозировании меток классов.

     - 'uniform' все соседи имеют одинаковый вес,
     - 'distance' веса обратно пропорциональны расстоянию до соседей.

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

   - *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,
     - '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.

3. Для нового объекта z необходимо вычислить расстояние до всех объектов обучающей выборки

$$d(z, x_i)$$

где $d(z, x_i)$ – выбранная метрика расстояния между объектами. Находим k ближайших соседей нового документа среди документов обучающей выборки X по расстояниям, вычисленным на предыдущем шаге.

4. Присваиваем новому документу тот класс, который наиболее часто встречается среди классов k ближайших соседей.
5. Оценка качества модели

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

6. Тонкая настройка модели

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

7. 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:

$$d(x_1, x_2) = \sqrt{\sum_{i=1}^{n} \left(x_{1_i} - x_{2_i}\right)^2},$$

Манхэттенское расстояние:

$$d(x_1, x_2) = \sum_{i=1}^{n} \left|x_{1_i} - x_{2_i}\right|$$

Косинусное расстояние:

$$d(x_1, x_2) = \frac{x_1 \cdot x_2}{||x_1|| \cdot ||x_2||}$$

Расстояние Чебышева:

$$d(x_1, x_2) = \max_{i=1}^{n} |x_{1_i} - x_{2_i}|$$

Метрика Минковского:

$$d(x_1, x_2) = \left(\sum_{i=1}^{n} \left|x_{1_i} - x_{2_i}\right|^p\right)^{1/p}$$

Здесь x_1 и x_2 - 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 [3]

   - *criterion* – критерий разделения при построении дерева решений. Поддерживаются 2 основных критерия
     - «gini» критерий Джини,
     - «entropy» прирост 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.

   - *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 от общего числа признаков,
     - Если значение равно "sqrt", то рассматривается корень от общего числа признаков,
     - Если значение равно "log2", то рассматривается log2 от общего числа признаков,

- Если значение равно None, то рассматриваются все признаки.

- *min_samples_split* – минимальное количество образцов, необходимое для разбиения внутреннего узла, чтобы тот не был терминальным.

- *min_samples_leaf* — определяет минимальное количество образцов, необходимых для формирования листа (терминального узла). Если количество образцов в узле меньше или равно значению min_samples_leaf, то узел становится листом без дальнейшего разделения.

- *max_leaf_nodes* – максимальное количество листьев дерева. Если None, то число листьев не ограничено.

- *random_state* – параметр для контроля случайности процесса обучения, устанавливает начальное значение для генератора случайных чисел. Если мы хотим получать одинаковый результат, то устанавливаем значение int, если хотим, чтобы параметр менялся каждый раз, то None.

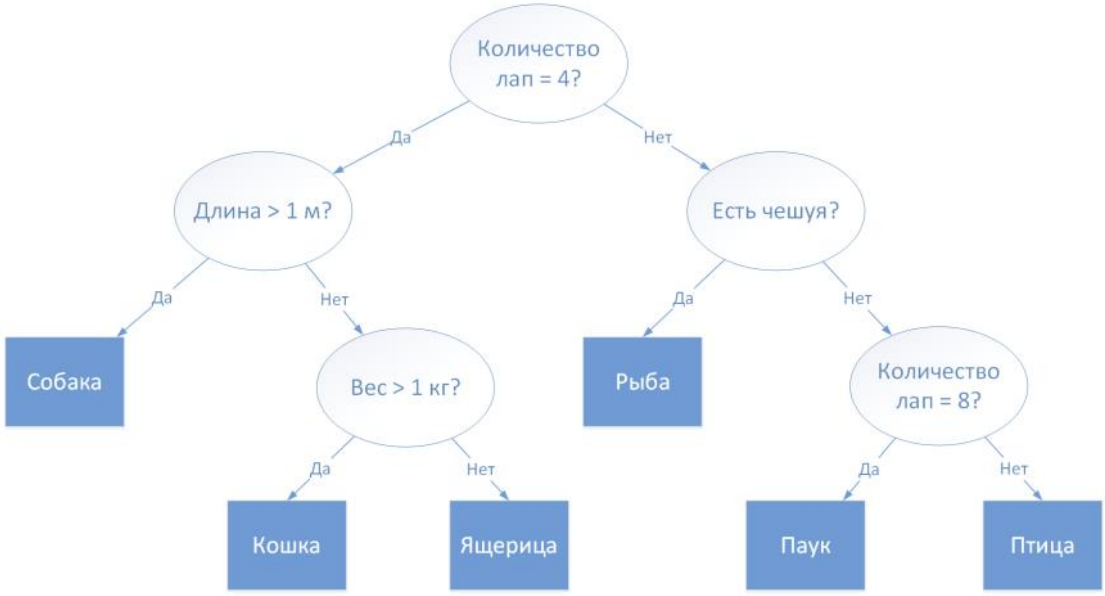
2. Выбор признака для разбиения данных: на первом шаге выбирается признак, который наилучшим образом разделяет выборку на подмножества (обычно используется мера информативности, такая как энтропия Шеннона или критерий Джини).

   $Gini(p) = 1 - \sum_{k=1}^{K} p_k^2$ – критерий Джинни

   $H(p) = -\sum_{k=1}^{K} \log_2(p_j)$ – энтропия Шенона

3. Разбиение выборки по значению выбранного признака: данные делятся на подмножества, соответствующие значениям выбранного признака. Для каждого значения признака создается свой дочерний узел в дереве.
4. 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.
5. 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.



6. Оценка качества модели

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

7. Тонкая настройка модели

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

8. 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 [3]

   Очевидно, многие параметры будут совпадать с параметрами дерева решений

   - *criterion* – критерий разделения при построении дерева решений. Поддерживаются 2 основных критерия
     - «gini» критерий Джини,
     - «entropy» прирост информации Шеннона.

   - *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.

   - *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 от общего числа признаков,
     - Если значение равно "sqrt", то рассматривается корень от общего числа признаков,
     - Если значение равно "log2", то рассматривается log2 от общего числа признаков,
     - Если значение равно None, то рассматриваются все признаки.

   - *min_samples_split* – минимальное количество образцов, необходимое для разбиения внутреннего узла, чтобы тот не был терминальным.

   - *min_samples_leaf* — определяет минимальное количество образцов, необходимых для формирования листа (терминального узла). Если

количество образцов в узле меньше или равно значению min_samples_leaf, то узел становится листом без дальнейшего разделения.

- *max_leaf_nodes* – максимальное количество листьев дерева. Если None, то число листьев не ограничено.

- *random_state* – параметр для контроля случайности процесса обучения, устанавливает начальное значение для генератора случайных чисел. Если мы хотим получать одинаковый результат, то устанавливаем значение int, если хотим, чтобы параметр менялся каждый раз, то None.

Рассмотрим параметры всего леса, а не отдельного дерева

- *n_estimators* – количество деревьев в лесу. По умолчанию - 100.

- *bootstrap* – принимает значения True или False (По умолчанию - True). Используется метод бутстрэпа для построения деревьев или нет.

Бутстрэп заключается в том, что при построении каждого дерева случайного леса выбирается случайный поднабор данных из обучающего набора с возвращением. Это означает, что одно и то же наблюдение может быть выбрано несколько раз, а другие наблюдения могут быть исключены из выборки. Применение бутстрэп-выборки позволяет создавать различные подвыборки для каждого дерева, что вносит вариацию в данные, улучшая обобщающую способность модели и снижая переобучение.

- *oob_score* – принимает значения True или False (По умолчанию - False).

При использовании бустрэпа, данные, которые не попали в выборку для построения дерева, называются out-of-bag (OOB) наблюдениями.

OOB наблюдения, используются для оценки точности каждого решающего дерева. Это позволяет более эффективно использовать имеющиеся данные и получать более точные оценки качества модели.

2. Выберите N случайных записей из набора данных.
3. Постройте дерево решений на основе этих N записей. Подробнее построение дерева рассмотрено в параграфе 2.3.

4. Выберите количество деревьев в вашем алгоритме и повторите шаги 1 и 2.
5. 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.
6. Оценка качества модели:

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

7. Тонкая настройка модели:

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

8. 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,

- *Дополнительно будет приведено сравнение с классификацией на признаковом пространстве ключевых слов, однако мы не будем рассматривать это исследование подробно, так как оно не является основным*.

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.

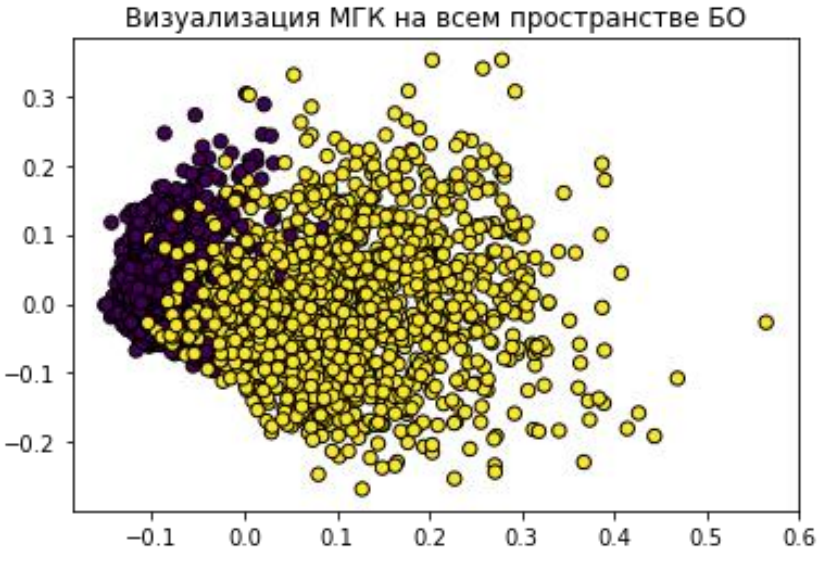
Словарь терминов, составленный по БО (названия, аннотации, ключевые слова), имеет размерность 25124, словарь, составленный по названиям 6394 а, по ключевым словам, 12186.

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

Визуализация с исходными метками

Здесь желтые точки – ИАД, фиолетовые точки – не ИАД

Для визуализации требуется сократить размерность данных, для этого в работе используется ***метод главных компонент*** рисунок 1**.**



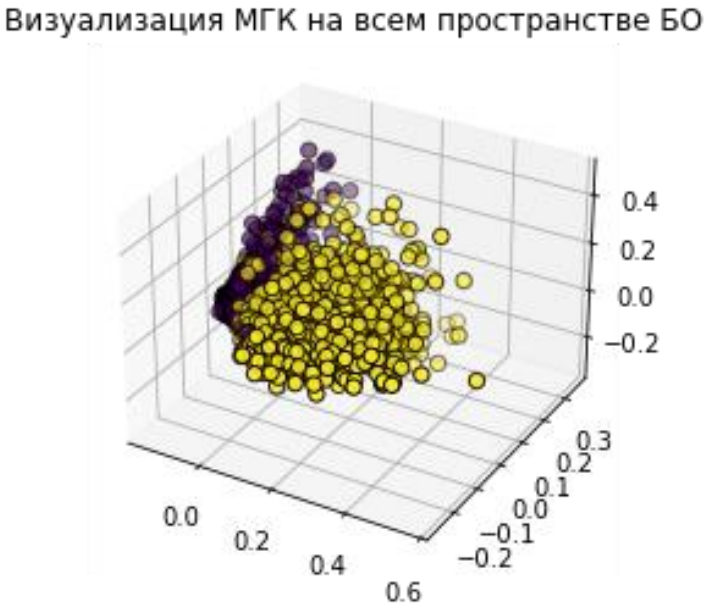


Рис 1. Визуализация с исходными метками.

2.3 Классификация

В таблицах используются обозначения, все метрики описаны в пункте 1.3

- precision – точность
- recall – полнота
- f1-score – F1-мера
- accuracy – аккуратность
- macro avg – среднее арифметическое
- weighted avg – среднее взвешенное
- support – количество наблюдений в каждом классе из тестового набора данных

Логистическая регрессия представлена в таблице 1

Таблица 1 – результат метода логистической регрессии

	По названиям				По библиографическому описанию			
	precision	recall	f1-score	support	precision	recall	f1-score	support
False	0.85	0.93	0.89	1228	0.95	0.98	0.96	1228
True	0.91	0.81	0.85	1059	0.98	0.94	0.96	1059
accuracy			0.87	2287			0.96	2287
macro avg	0.88	0.87	0.87	2287	0.96	0.96	0.96	2287
weighted avg	0.88	0.87	0.87	2287	0.96	0.96	0.96	2287

Оптимальные параметры при использовании только названий документов

- Гребневая регуляризация 'penalty': 'l2',
- Обратная сила регуляризации 'C': 50,
- Веса классов не учитываются 'class_weight': None,
- Максимальное число итераций для оптимизации функции потерь 'max_iter': 15,
- Стратегия обработки многоклассовых проблем один против всех, значение по умолчанию 'multi_class': 'ovr',
- Алгоритм оптимизации функции потерь методом Ньютона-Рафсона 'solver': 'newton-cg'.

Оптимальные параметры при использовании БО

- Гребневая регуляризация 'penalty': 'l2',
- Обратная сила регуляризации 'C': 50,
- Веса классов учитываются 'class_weight': 'balanced'
- Максимальное число итераций для оптимизации функции потерь 'max_iter': 15,
- Стратегия обработки многоклассовых проблем многовариантная логистическая регрессия 'multi_class': 'multinomial',
- Алгоритм оптимизации функции потерь методом Ньютона-Рафсона 'solver': 'newton-cg'.

К-ближайших соседей представлен в таблице 2

Таблица 2 – результат метода к-ближайших соседей

	По названиям				По библиографическому описанию			
	precision	recall	f1-score	support	precision	recall	f1-score	support
False	0.88	0.66	0.76	1228	0.79	1.00	0.88	1228
True	0.70	0.90	0.78	1059	1.00	0.69	0.81	1059
accuracy			0.77	2287			0.85	2287
macro avg	0.79	0.78	0.77	2287	0.89	0.84	0.85	2287
weighted avg	0.80	0.77	0.77	2287	0.88	0.85	0.85	2287

Оптимальные параметры при использовании только названий документов

- Число ближайших соседей 'n_neighbors': 10,
- Алгоритм поиска ближайших соседей - KD-дерево 'algorithm': 'kd_tree',
- Метрика для вычисления расстояния до соседей 'metric': 'euclidean',
- Веса для прогнозирования меток классов, все соседи имею одинаковый вес 'weights': 'uniform'.

Оптимальные параметры при использовании БО

- Число ближайших соседей 'n_neighbors': 10,
- Алгоритм поиска ближайших соседей - перебор всех точек в обучающем наборе данных 'algorithm': 'brute',
- Метрика для вычисления расстояния до соседей 'metric': 'manhattan',
- Веса для прогнозирования меток классов, все соседи имею одинаковый вес 'weights': 'uniform'.

Дерево решений представлено в таблице 3

Таблица 3 – результат метода дерева решений

	По названиям				По библиографическому описанию			
	precision	recall	f1-score	support	precision	recall	f1-score	support
False	0.81	0.83	0.82	1228	0.99	0.99	0.99	1228
True	0.80	0.77	0.79	1059	0.98	0.99	0.99	1059
accuracy			0.80	2287			0.99	2287
macro avg	0.80	0.80	0.80	2287	0.99	0.99	0.99	2287
weighted avg	0.80	0.80	0.80	2287	0.99	0.99	0.99	2287

Оптимальные параметры при использовании только названий документов

- Критерий энтропии Шеннона разделения при построении дерева решений 'criterion': 'entropy',
- Максимальная глубина дерева 'max_depth': 80,
- Количество признаков, рассматриваемых для разбиения 'max_features': 0.3,
- Максимальное количество листьев дерева не ограничено 'max_leaf_nodes': None,
- Минимальное количество образцов, необходимых для формирования терминального узла 'min_samples_leaf': 1,
- Минимальное количество образцов, необходимое для разбиения внутреннего узла, чтобы тот не был терминальным 'min_samples_split': 5,
- Стратегия для выбора разбиения в каждом узле «лучшее разбиение» 'splitter': 'best'.

Оптимальные параметры при использовании БО

- Критерий Джинни для разделения при построении дерева решений 'criterion': 'gini',
- Максимальная глубина дерева 'max_depth': 10,

- Количество признаков, рассматриваемых для разбиения не ограничено 'max_features': None,
- Максимальное количество листьев дерева не ограничено 'max_leaf_nodes': None,
- Минимальное количество образцов, необходимых для формирования терминального узла 'min_samples_leaf': 5,
- Минимальное количество образцов, необходимое для разбиения внутреннего узла, чтобы тот не был терминальным 'min_samples_split': 2,
- Стратегия для выбора разбиения в каждом узле «случайное разбиение» 'splitter': 'random'.

Случайный лес представлен в таблице 4

Таблица 4 – результат метода случайный лес

	По названиям				По библиографическому описанию			
	precision	recall	f1-score	support	precision	recall	f1-score	support
False	0.86	0.90	0.88	1228	0.99	0.99	0.99	1228
True	0.88	0.83	0.86	1059	0.99	0.99	0.99	1059
accuracy			0.87	2287			0.99	2287
macro avg	0.87	0.87	0.87	2287	0.99	0.99	0.99	2287
weighted avg	0.87	0.87	0.87	2287	0.99	0.99	0.99	2287

Оптимальные параметры при использовании только названий документов

- Количество деревьев в лесу 'n_estimators': 400,
- Максимальная глубина дерева 'max_depth': 300,
- Критерий Джинни для разделения при построении дерева решений 'criterion': 'gini',
- Количество признаков, которые рассматриваются при поиске лучшего разбиения log2(число признаков) 'max_features': 'log2',

- Максимальное количество листьев дерева не ограничено 'max_leaf_nodes': None,
- Минимальное количество образцов, необходимых для формирования терминального узла 'min_samples_leaf': 1,
- Минимальное количество образцов, необходимое для разбиения внутреннего узла, чтобы тот не был терминальным 'min_samples_split': 10,
- Используется метод бустрэпа 'bootstrap': True,
- OOB наблюдения, используются для оценки точности каждого решающего дерева 'oob_score': True

Оптимальные параметры при использовании БО

- Количество деревьев в лесу 'n_estimators': 700,
- Максимальная глубина дерева 'max_depth': 100,
- Критерий энтропии Шеннона для разделения при построении дерева решений 'criterion': 'entropy',
- Для от общего числа признаков, которые рассматриваются при поиске лучшего разбиения 'max_features': 0.1,
- Максимальное количество листьев дерева не ограничено 'max_leaf_nodes': None,
- Минимальное количество образцов, необходимых для формирования терминального узла 'min_samples_leaf': 1,
- Минимальное количество образцов, необходимое для разбиения внутреннего узла, чтобы тот не был терминальным 'min_samples_split': 2,
- Используется метод бустрэпа 'bootstrap': True,
- OOB наблюдения, используются для оценки точности каждого решающего дерева 'oob_score': True

Сравнение методов и признаковых пространств представлено в таблице 5

Таблица 5 – сравнение методов и признаковых пространств

	Названия	Библиографическое описание	Ключевые слова
Метрика / Метод	Accuracy	Accuracy	Accuracy
Логистическая регрессия	0.87	0.96	0.96
КБС	0.77	0.85	0.84
Дерево решений	0.80	0.99	0.96
Случайный лес	0.87	0.99	0.97

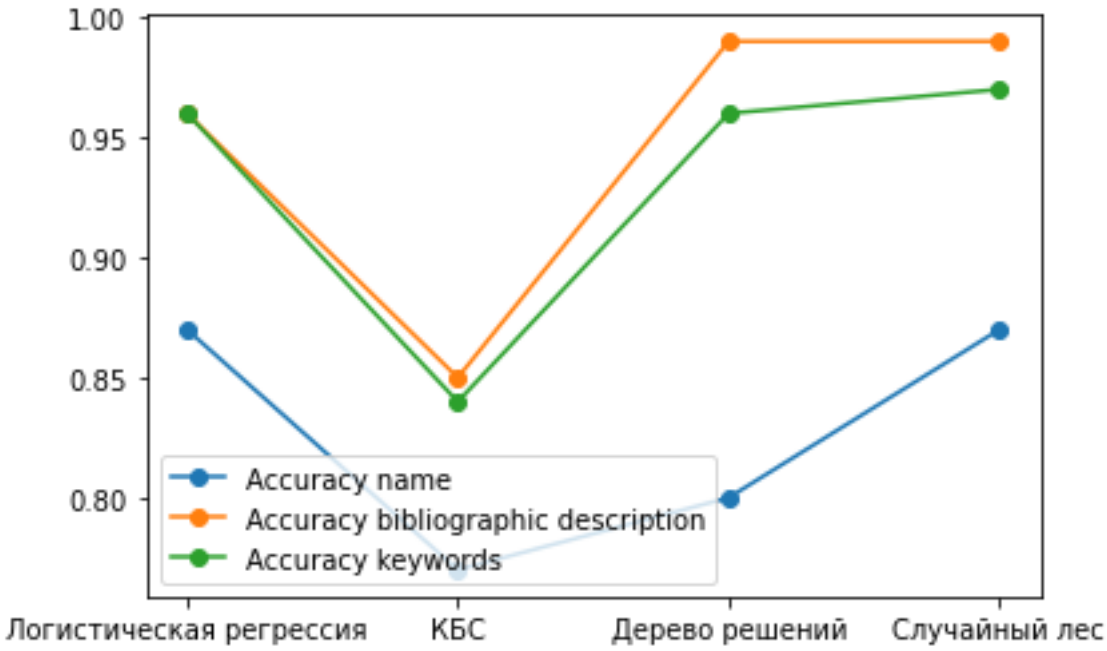


Рис 2. Сравнение accuracy по различным методам

2.4 Выводы по главе

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

В ходе исследований лучшего результата удалось добиться с помощью алгоритма случайного леса. Для всех методов были найдены и описаны

оптимальные гиперпараметры, использование меньшего признакового пространства требует более точной настройки методов.

Метод деревьев решений показал наибольшую чувствительность к признаковому пространству, тогда как логистическая регрессия наименьшую. В самом деле алгоритм к-ближайших соседей потерял в качестве меньше всего, однако он в целом показал худшие результаты.

Так как в выборке имелся хорошо сформированный набор ключевых слов, было приведено сравнение с ним, набор терминов вдвое меньше, чем набор по БО, но вдвое больше, чем набор по названиям, то есть представляет собой среднюю размерность.

Можно сказать, что такое сокращение было успешно, так как мы почти не потеряли в качестве классификации на многих методах и не потеряли вовсе, используя логистическую регрессию.

Однако стоит отметить, что список ключевых слов может отсутствовать в литературе, а его формирование является отдельной задачей, что усложняет работу, название же есть всегда.

Итог

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.)

Если система требует скорости, то использование названий документов при бинарной классификации целесообразно, такая система будет быстрой и достаточно точной, до 87%, что тем более является отличным результатом, если возможности собрать больше данных нет. Однако если скорости не требуется, то при классификации научных документов, лучше будет остановится на использовании библиографических описаний, а при возможности ключевых слов.

Глава 4. МУЛЬТИКЛАССОВАЯ КЛАССИФИКАЦИЯ

Подход к многоклассовой классификации несколько отличается от бинарной, такая классификация имеет свою специфику, некоторое из перечисленного может встречаться и в бинарной классификации, однако чаще встречается в мультиклассовой:

- Наличие большого количества классов. С увеличением количества классов возрастает сложность 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. Могут быть случаи, когда классификатор не уверен к какому относить объект. Так же 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".
- Подходы «жесткой» и «мягкой» классификации, при мягкой классификации классификатор дает степень уверенности (вероятность), что объект действительно относится к данному классу.
- 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.

4.1 Описание выборки

В этой главе мы рассмотрим выборку из 10 классов, сформированную А.В. Кононенко и состоящую из документов научно-технической литературы. 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

Пример БО:

Название:

Combining ensemble methods and social network metrics for improving accuracy of OCSVM on intrusion detection in SCADA systems

Аннотация:

Modern Supervisory Control and Data Acquisition SCADA systems used by the electric utility industry to monitor and control electric power generation, transmission and distribution are recognized today as critical components of the electric power delivery infrastructure. SCADA systems are large, complex and incorporate increasing numbers of widely distributed components. The presence of a real time intrusion detection mechanism, which can cope with different types of attacks, is of great importance, in order to defend a system against cyber attacks This defense mechanism must be distributed, cheap and above all accurate, since false positive

alarms, or mistakes regarding the origin of the intrusion mean severe costs for the system. Recently an integrated detection mechanism, namely IT-OCSVM was proposed, which is distributed in a SCADA network as a part of a distributed intrusion detection system (IDS), providing accurate data about the origin and the time of an intrusion. In this paper we also analyze the architecture of the integrated detection mechanism and we perform extensive simulations based on real cyber attacks in a small SCADA testbed in order to evaluate the performance of the proposed mechanism.

Ключевые слова:

Cryptography and Security

Словарь терминов, составленный по БО (названия, аннотации, ключевые слова), имеет размерность 42969, словарь, составленный по названиям 13880. В сравнении с выборкой, используемой для бинарной классификации, текущая имеет большую размерность, при этом словарь ключевых слов имеет всего 416, что не даст так же эффективно его использовать. Словарь терминов из названий в 3 раза меньше, чем словарь терминов из БО.

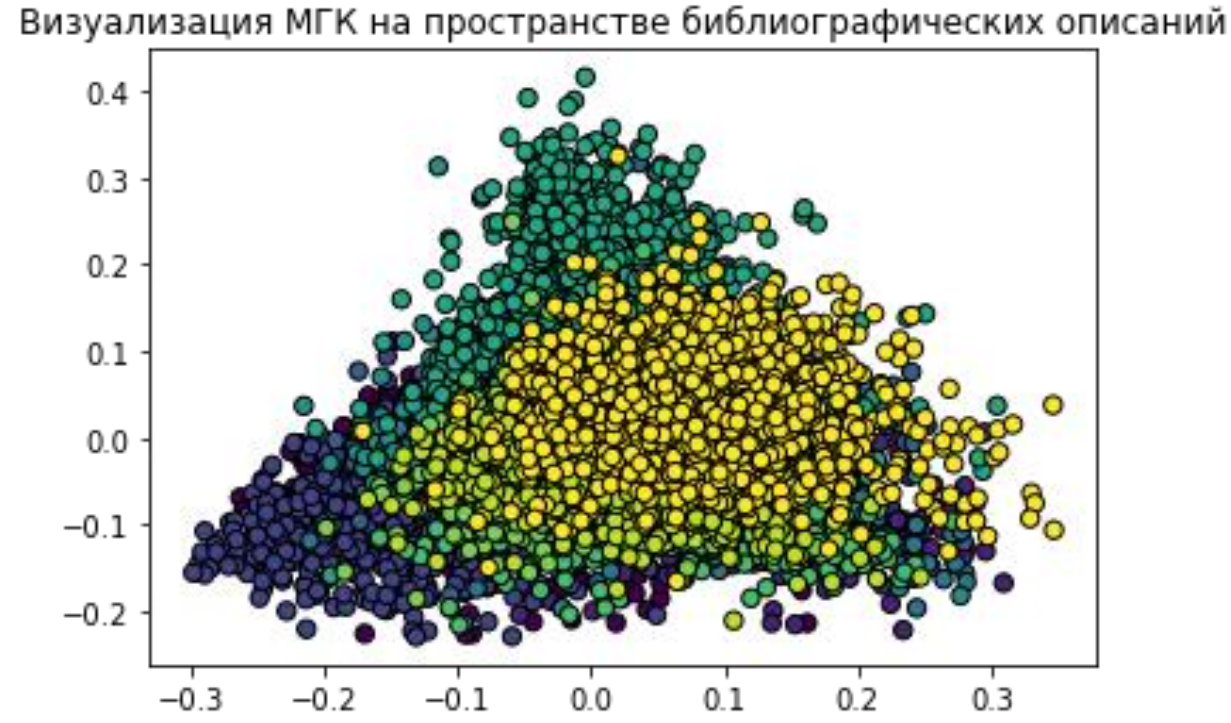
4.2 Предобработка и визуализация.

Предобработка признаковых пространств состояла из следующих пунктов

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

Визуализация с исходными метками

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



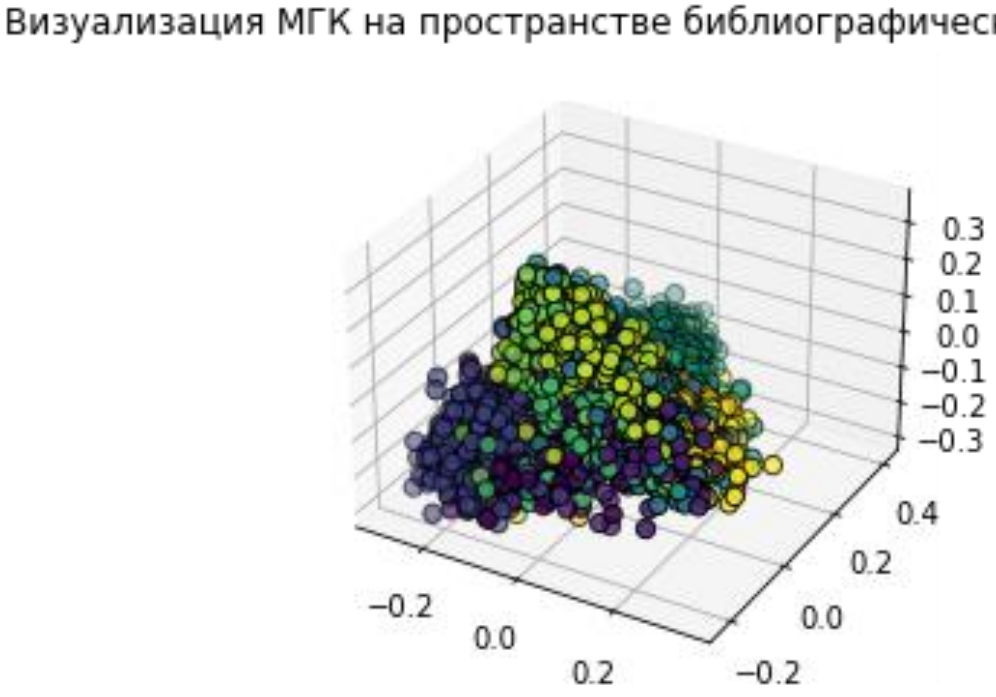


Рис 3. Визуализация с исходными метками.

4.3 Классификация

Сначала рассмотрим результаты классификации по признаковому пространству, построенному на названиях.

Логистическая регрессия представлена в таблице 6

Таблица 6 – результат метода логистической регрессии

	precision	recall	f1-score	support
computer vision	0.55	0.59	0.57	1193
control systems	0.45	0.44	0.44	1245
cyber security	0.65	0.69	0.67	822
database	0.34	0.32	0.33	1330
expert	0.38	0.32	0.35	1332
fuzzy	0.87	0.67	0.76	1232
information retrieval	0.62	0.55	0.58	1398
neural nets	0.46	0.55	0.50	1313
robotic	0.48	0.52	0.50	1031
text mining	0.59	0.59	0.59	1068
accuracy	0.55	0.65	0.60	1355
macro avg			0.53	13319
weighted avg	0.54	0.54	0.54	13319

Оптимальные параметры

- Гребневая регуляризация 'penalty': 'l2',
- Обратная сила регуляризации 'C': 1,
- Веса классов не учитываются 'class_weight': 'balanced',
- Максимальное число итераций для оптимизации функции потерь 'max_iter': 50,
- Стратегия обработки многоклассовых проблем один против всех, значение по умолчанию 'multi_class': 'ovr',
- Алгоритм оптимизации функции потерь на основе координатного спуска 'solver': 'liblinear'.

К-ближайших соседей представлен в таблице 7

Таблица 7 – результат метода к-ближайших соседей

	precision	recall	f1-score	support
computer vision	0.41	0.45	0.43	1245
control systems	0.48	0.35	0.41	1031
cyber security	0.51	0.85	0.64	2015
database	0.40	0.24	0.30	1330
expert	0.47	0.18	0.26	1332
fuzzy	0.66	0.63	0.64	1232
information retrieval	0.59	0.51	0.54	1398
neural nets	0.47	0.43	0.45	1313
robotic	0.57	0.43	0.49	1068
text mining	0.43	0.66	0.52	1355
accuracy			0.50	13319
macro avg	0.50	0.47	0.47	13319
weighted avg	0.50	0.50	0.48	13319

Оптимальные параметры

- Число ближайших соседей 'n_neighbors': 50,
- Алгоритм поиска ближайших соседей – перебор всех точек в обучающем наборе 'algorithm': 'brute',
- Метрика для вычисления расстояния до соседей 'metric': 'cosine',
- Веса для прогнозирования меток классов обратно пропорциональны расстоянию до соседей 'weights': 'distance'.

Дерево решений представлено в таблице 8

Таблица 8 – результат алгоритма дерева решений

	precision	recall	f1-score	support
computer vision	0.38	0.38	0.38	1245
control systems	0.50	0.41	0.45	1031
cyber security	0.38	0.81	0.52	2015
database	0.40	0.22	0.29	1330
expert	0.29	0.20	0.24	1332
fuzzy	0.96	0.62	0.75	1232
information retrieval	0.68	0.44	0.54	1398
neural nets	0.41	0.53	0.47	1313
robotic	0.69	0.40	0.50	1068
text mining	0.56	0.52	0.54	1355
accuracy			0.47	13319
macro avg	0.53	0.45	0.47	13319
weighted avg	0.51	0.47	0.47	13319

Оптимальные параметры

- Критерий Джинни для разделения при построении дерева решений 'criterion': 'gini',
- Максимальная глубина дерева 'max_depth': 200,
- Количество признаков, рассматриваемых для разбиения 'max_features': 0.7,
- Максимальное количество листьев дерева не ограничено 'max_leaf_nodes': 100,
- Минимальное количество образцов, необходимых для формирования терминального узла 'min_samples_leaf': 5,
- Минимальное количество образцов, необходимое для разбиения внутреннего узла, чтобы тот не был терминальным 'min_samples_split': 10,
- Стратегия для выбора разбиения в каждом узле «случайное разбиение» 'splitter': 'random'.

Случайный лес представлен в таблице 9

Таблица 9 – результат алгоритма случайный лес

	precision	recall	f1-score	support
computer vision	0.37	0.43	0.40	1245
control systems	0.45	0.39	0.42	1031
cyber security	0.63	0.70	0.66	2015
database	0.25	0.37	0.30	1330
expert	0.35	0.23	0.28	1332
fuzzy	0.87	0.65	0.74	1232
information retrieval	0.58	0.46	0.51	1398
neural nets	0.45	0.50	0.47	1313
robotic	0.52	0.47	0.49	1068
text mining	0.51	0.53	0.52	1355
accuracy			0.49	13319
macro avg	0.50	0.47	0.48	13319
weighted avg	0.50	0.49	0.49	13319

Оптимальные параметры

- Количество деревьев в лесу 'n_estimators': 400,
- Максимальная глубина дерева 'max_depth': 150,
- Критерий Джинни для разделения при построении дерева решений 'criterion': 'gini',
- Для от общего числа признаков, которые рассматриваются при поиске лучшего разбиения 'max_features': 0.3,
- Максимальное количество листьев дерева не ограничено 'max_leaf_nodes': None,
- Минимальное количество образцов, необходимых для формирования терминального узла 'min_samples_leaf': 1,
- Минимальное количество образцов, необходимое для разбиения внутреннего узла, чтобы тот не был терминальным 'min_samples_split': 20,
- Не используется метод бустрэпа 'bootstrap': False.

Теперь рассмотрим классификацию по признаковому пространству, построенному на всем библиографическом описании.

Логистическая регрессия представлена в таблице 10

Таблица 10 – результат метода логистической регрессии

	precision	recall	f1-score	support
computer vision	0.76	0.88	0.82	1245
control systems	0.77	0.77	0.77	1031
cyber security	0.91	0.89	0.90	2015
database	0.89	0.85	0.87	1330
expert	0.90	0.89	0.90	1332
fuzzy	0.99	0.95	0.97	1232
information retrieval	0.87	0.77	0.81	1398
neural nets	0.82	0.85	0.83	1313
robotic	0.82	0.80	0.81	1068
text mining	0.84	0.88	0.86	1355
accuracy			0.86	13319
macro avg	0.85	0.85	0.85	13319
weighted avg	0.86	0.86	0.86	13319

Оптимальные параметры

- Лассо регуляризация 'penalty': 'l1',
- Обратная сила регуляризации 'C': 1,
- Веса классов не учитываются 'class_weight': 'balanced',
- Максимальное число итераций для оптимизации функции потерь 'max_iter': 20,
- Стратегия обработки многоклассовых проблем один против всех, значение по умолчанию 'multi_class': 'ovr',
- Алгоритм оптимизации функции потерь на основе координатного спуска 'solver': 'liblinear'.

К-ближайших соседей представлен в таблице 11

Таблица 11 – результат метода к-ближайших соседей

	precision	recall	f1-score	support
computer vision	0.42	0.78	0.55	1245
control systems	0.68	0.50	0.58	1031
cyber security	0.61	0.94	0.74	2015
database	0.87	0.18	0.30	1330
expert	0.81	0.25	0.38	1332
fuzzy	0.91	0.75	0.82	1232
information retrieval	0.74	0.67	0.71	1398
neural nets	0.69	0.57	0.63	1313
robotic	0.71	0.68	0.69	1068
text mining	0.56	0.86	0.68	1355
accuracy			0.64	13319
macro avg	0.70	0.62	0.61	13319
weighted avg	0.70	0.64	0.61	13319

Оптимальные параметры

- Число ближайших соседей 'n_neighbors': 500,
- Алгоритм поиска ближайших соседей - перебор всех точек в обучающем наборе данных 'algorithm': 'brute',
- Метрика для вычисления расстояния до соседей 'metric': 'cosine',
- Веса для прогнозирования меток классов обратно пропорциональны расстоянию до соседей 'weights': 'distance'.

Дерево решений представлено в таблице 12

Таблица 12 – результат алгоритма дерева решений

	precision	recall	f1-score	support
computer vision	0.88	0.87	0.88	1245
control systems	0.89	0.74	0.81	1031
cyber security	0.88	0.92	0.90	2015
database	0.84	0.91	0.88	1330
expert	0.93	0.87	0.90	1332
fuzzy	0.99	0.98	0.98	1232
information retrieval	0.80	0.81	0.80	1398
neural nets	0.90	0.86	0.88	1313
robotic	0.85	0.89	0.87	1068
text mining	0.85	0.90	0.88	1355
accuracy			0.88	13319
macro avg	0.88	0.88	0.88	13319
weighted avg	0.88	0.88	0.88	13319

Оптимальные параметры

- Критерий энтропии Шеннона для разделения при построении дерева решений 'criterion': 'entropy',
- Максимальная глубина дерева 'max_depth': 50,
- Количество признаков, рассматриваемых для разбиения не ограничено 'max_features': None,
- Максимальное количество листьев дерева не ограничено 'max_leaf_nodes': 100,
- Минимальное количество образцов, необходимых для формирования терминального узла 'min_samples_leaf': 5,
- Минимальное количество образцов, необходимое для разбиения внутреннего узла, чтобы тот не был терминальным 'min_samples_split': 2,
- Стратегия для выбора разбиения в каждом узле «лучшее разбиение» 'splitter': 'best'.

Случайный лес представлен в таблице 13

Таблица 13 – результат алгоритма 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

Оптимальные параметры

- Количество деревьев в лесу 'n_estimators': 1500,
- Максимальная глубина дерева 'max_depth': 150,
- Критерий Джинни для разделения при построении дерева решений 'criterion': 'gini',
- Для от общего числа признаков, которые рассматриваются при поиске лучшего разбиения 'max_features': 0.1,
- Максимальное количество листьев дерева не ограничено 'max_leaf_nodes': None,
- Минимальное количество образцов, необходимых для формирования терминального узла 'min_samples_leaf': 1,
- Минимальное количество образцов, необходимое для разбиения внутреннего узла, чтобы тот не был терминальным 'min_samples_split': 2,
- Не используется метод бустрэпа 'bootstrap': False.

Сравнение методов и признаковых пространств представлено в таблице 14

Таблица 14 – сравнение методов и признаковых пространств

	Названия	Библиографическое описание
Метрика / Метод	Accuracy	Accuracy
Логистическая регрессия	0.60	0.86
КБС	0.50	0.64
Дерево решений	0.47	0.88
Случайный лес	0.49	0.91

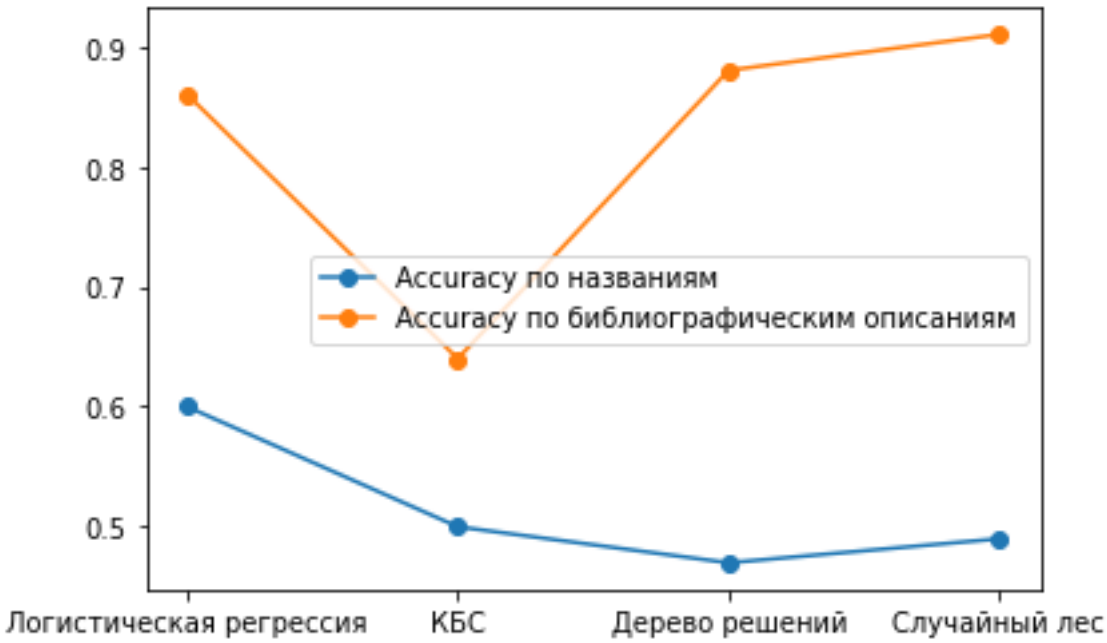


Рис 4. Сравнение accuracy по различным методам

4.4 Выводы по главе

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

В ходе исследований лучшего результата удалось добиться с помощью алгоритма случайного леса. Для всех методов были найдены и описаны

оптимальные гиперпараметры, использование меньшего признакового пространства требует более точной настройки методов.

Выяснилось, что одни названия дают плохой результат, при том, что на бинарной выборке различия в качестве наблюдалось не так ярко. Сказались проблемы многоклассовой классификации, размерность признакового пространства сыграло роль, в следствии чего можно сказать, что в данном случае нельзя использовать признаковое пространство сокращенной размерности, не смотря на большие вычислительные затраты на обработку библиографических описаний.

Выбор лучшего метода схож с бинарной классификацией, лучшие результаты показал случайный лес на пространстве БО, однако на сокращенном признаковом пространстве лучше всего справилось логистическая регрессия, по результатам исследований именно этот метод меньше всего теряет в качестве при сокращении признакового пространства.

ОПИСАНИЕ ИСПОЛЬЗУЕМОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Для исследований используются высокоуровневый язык программирования Python, основные библиотеки, используемые в работе: scikit-learn, matplotlib, NumPy и Pandas.

Python – это высокоуровневый язык программирования, который широко применяется в области анализа данных и машинного обучения. Он обладает простым синтаксисом, мощными библиотеками и поддерживает различные парадигмы программирования [6].

Scikit-learn – это одна из наиболее популярных библиотек машинного обучения написанная на Python. Она предоставляет широкий спектр алгоритмов классификации, регрессии, кластеризации, а также инструменты для предобработки данных, выбора моделей, оценки качества и валидации моделей [6].

Matplotlib – это библиотека визуализации данных для языка Python. Она предоставляет широкие возможности для создания различных типов графиков, диаграмм и визуализации результатов анализа данных [6].

NumPy – это библиотека для работы с массивами и матрицами в Python. Она предоставляет эффективные структуры данных и функции для выполнения математических операций на массивах. NumPy широко используется в анализе данных и машинном обучении для обработки и представления числовых данных [6].

Pandas – это библиотека, предоставляющая высокоуровневые структуры данных и инструменты для анализа данных. Она позволяет удобно и эффективно работать с табличными данными, включая чтение и запись данных, фильтрацию, сортировку, группировку, агрегацию и многое другое [6].

ЗАКЛЮЧЕНИЕ

Целью работы было рассмотрение возможности использования сокращенного признакового пространства. Подводя итог, можно сказать, что цель выполнена, признаковые пространства небольшой размерности можно рассматривать и использовать для классификации текстовых коллекций, однако это требует большее точных настроек параметров исследований, особенно в случае многоклассовой классификации.

В работе упоминаются методы сокращения размерности, такие как метод главных компонент, обычно он используется для визуализации, но также стоит рассмотреть и для классификации данных, это явно сократит время обучения алгоритмов, но может и сказаться на качестве классификации.

Подробно описаны все методы классификации, а также используемое программное обеспечение для их реализации. Изучены отличие бинарной и многоклассовой классификации, рассмотрены особенности классификации текстовых данных.

Список литературы

1) Хабр – русскоязычный IT-портал https://habr.com/ru/all/
2) Документация Python https://docs.python.org/3/
3) Документация Python библиотеки scikit-learn https://scikit-learn.org/stable/
4) Документация Python библиотеки Matplotlib https://matplotlib.org/
5) Stack Overflow – форум для разработчиков https://stackoverflow.com/
6) Википедия - https://ru.wikipedia.org/wiki/
7) Методы выявления закономерностей из эмпирических данных. В.О. Толчеев. М.:Издательсикй дом МЭИ, 2010. – 88 с.
8) Машинное обучение без лишних слов. Андрей Бурков. СПб.: Питер, 2020 — 192 с.
9) Инженерия машинного обучения. Андрей Бурков. М.: ДМК-Пресс, 2022 — 306 с.