МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра анализа данных и искусственного интеллекта

КУРСОВАЯ РАБОТА

ЗАДАЧА КЛАССИФИКАЦИИ В МАШИННОМ ОБУЧЕНИИ

Работу выполнила _____ Д.В. Кощавцев
(подпись)

Направление подготовки 01.03.02 Прикладная математика и информатика
курс 3

Направленность (профиль) Прикладная математика

Научный руководитель
канд. физ.-мат. наук, доц. _____ Г.В. Калайдина
(подпись)

Нормоконтролер
канд. физ.-мат. наук, доц. _____ Г.В. Калайдина
(подпись)

Краснодар
2023

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра анализа данных и искусственного интеллекта

КУРСОВАЯ РАБОТА

ЗАДАЧА КЛАССИФИКАЦИИ В МАШИННОМ ОБУЧЕНИИ

Работу выполнила _____ Д.В. Кощавцев
(подпись)

Направление подготовки 01.03.02 Прикладная математика и информатика
курс 3

Направленность (профиль) Прикладная математика

Научный руководитель
канд. физ.-мат. наук, доц. _____ Г.В. Калайдина
(подпись)

Нормоконтролер
канд. физ.-мат. наук, доц. _____ Г.В. Калайдина
(подпись)

Краснодар
2023

РЕФЕРАТ

Курсовая работа 23с., 15 источн.

МЕТОДЫ КЛАССИФИКАЦИИ, МАШИННОЕ ОБУЧЕНИЕ, PYTHON, НЕЙРОННЫЕ СЕТИ, STATISTICA, ЛИНЕЙНЫЕ МОДЕЛИ

Объектом исследования данной курсовой работы являются методы классификации машинного обучения и инструменты для решения задач классификации.

Целью работы является изучение наиболее часто используемых методов классификации для решения задач машинного обучения, инструментов для решения этих задач, а также исследование наиболее оптимального инструмента и метода для решения поставленной задачи.

В курсовой работе были изучены основные методы классификации данных, такие как: метод K-ближайших соседей, метод опорных векторов, наивный байесовский метод, деревья принятия решений и многоклассовая классификация, а также инструменты: язык программирования Python и его библиотеки и пакет STATISTICA.

В результате работы было определенно, что оптимальным инструментом для решения задачи классификации является язык программирования Python, из-за его обширного функционала по обработке данных и их визуализации, а также простоты изучения. Оптимального же метода для решения задач классификации не существует, так как у всех методов есть преимущества и недостатки и для определения того метода, который лучше всего использовать, стоит отталкиваться от постановки задачи и входных данных.

СОДЕРЖАНИЕ

Введение..... 3
1 Метод классификации..... 4
1.1 Метод К-ближайших соседей..... 4
1.2 Наивный байесовский классификатор..... 6
1.3 Деревья принятия решений..... 7
1.4 Многоклассовая классификация..... 10
1.5 Метод опорных векторов..... 11
2 Инструменты для решения задачи..... 13
2.1 Python и библиотеки..... 13
2.1.1 Библиотеки Data Science..... 13
2.1.2 Библиотека Scikit-learn..... 14
2.1.3 Библиотеки TensorFlow и Keras..... 14
2.2 Пакет STATISTICA..... 15
3 Решение задачи классификации..... 18
3.1 Оптимальный инструмент решения..... 18
3.2 Оптимальный метод решения..... 18
Заключение..... 20
Список используемых источников..... 21

ВВЕДЕНИЕ

Машинное обучение – это обширный раздел, связанный с искусственным интеллектом. Он изучает методы построения алгоритмов, которые способны самостоятельно изучаться или воспринимать опыт и знания у экспертов в виде базы знаний.

В наше время машинное обучение всё больше набирает популярность. Эту технологию применяют в разных сферах для автоматизации рутинных задач, обработки больших объёмов неструктурированных данных, оптимизации производства. Она позволяет решать задачи классификации, регрессии и кластеризации.

Задачу классификации в машинном обучении можно сформулировать как поиск отображения из исходного множества объектов в множество возможных классов. То есть требуется найти такое отображение, которое лучше всего приближает истинное соответствие между объектами и таргетами.

Целью данной курсовой работы будет рассмотрение некоторых методов классификации и инструментов для их реализации, а также определение наиболее оптимальных методов и инструментов.

1 Методы классификации

1.1 Метод K-ближайших соседей

Метод K-ближайших соседей представляет собой тип контролируемого алгоритма машинного обучения, используемого для классификации, регрессии, поиска обнаружения выбросов. Он использует для обучения все данные новой точки данных или экземпляра. Этот метод непараметрический алгоритм обучения: он не предполагает ничего о базовых данных. Это чрезвычайно полезная функция, поскольку большая часть реальных данных на самом деле не следует никаким теоретическим предположениям, таким как линейная разделимость, равномерное распределение и т.д.

Когда же можно использовать этот метод? Рассмотрим на примере: представим, что мы проводим классификацию объектов на два класса. Нам дана некоторая обучающая выборка и целевой объект. Предположим, что хотим определить, к какому классу относится этот объект. Если этот целевой объект находится в окружении, к примеру, объектов жёлтого класса, то интуитивно понятно, что этот объект тоже доложен принадлежать к жёлтому классу. Эта интуиция и отражает суть метода К-ближайших соседей — классифицировать целевой объект, исходя из того, какие классы у объектов, которые максимально похожи на него.

Когда оценочное значение представляет собой непрерывное число, такое как стоимость арендной платы, метод К-ближайших соседей используется для регрессии. Но мы могли бы также разделить квартиры на категории, например, на основе минимальной и максимальной арендной платы. Если значение дискретно, что делает его категорией, для классификации используется этот метод.

Также возможно оценить, какие соседи сильно отличаются от других. Это то же самое, что определять, какие точки данных находятся настолько далеко, что не вписываются ни в одно значение или категорию, когда это

происходит, метод К-ближайших соседей используется для обнаружения выбросов. Этот метод использует ранее помеченные данные, что делает его алгоритмом обучения под наблюдением.

При добавлении новой точки к набору данных, метод К-ближайших соседей использует только одну часть данных для определения класса добавленной точки. Так как алгоритму не нужно повторно просматривать все точки, это делает его алгоритмом ленивого обучения. Алгоритм также ничего не предполагает о базовых характеристиках данных, он не ожидает, что данные будут соответствовать какому-либо типу распределения, так как это непараметрический алгоритм обучения.

Несмотря на то, что формально фаза обучения отсутствует, алгоритм может легко переобучиться. Из-за этого алгоритму легко подстроиться под конкретные данные.

Метод К-ближайших соседей можно улучшить. У оригинального алгоритма есть один большой недостаток: он никак не учитывает расстояния до соседних объектов, хотя эта информация может быть полезной.

Исправить это можно назначив этим индикаторам веса, которые тем больше, чем ближе объект к целевому. Такой алгоритм называется взвешенным методом К-ближайших соседей. Есть множество вариантов выбора весов для объектов, которые можно поделить на две группы. В первой группе веса зависят лишь от порядкового номера объекта в отсортированном по близости к целевому объекту массива данных. Во второй группе методов вес является некоторой функцией от расстояния. Такая функция должна быть положительной на своей области определения, иначе модель будет поощрять несовпадение с некоторыми ближайшими соседями. Также необходимо, чтобы функция монотонно возрастала, чтобы вес близких соседей был больше, чем далёких. Такая функция называется ядерной функцией (или kernel function):

$$a(u) = arg \max_{y \in Y} \sum_{i=1}^{k} K(\frac{\rho(u,\ x_u^{(i)})}{h}) * \theta[y_u^{(i)} = y]$$

где h – некоторое положительное число, которое называется шириной окна; K – ядро; u – целевой объект.

От выбора ядра K зависит гладкость аппроксимации, но на её качество этот выбор почти не влияет. Чаще всего рассматривают прямоугольное ядро, в силу его простоты, либо гауссовское ядро, в силу его высокой точности.

Ширина окна, в свою очередь, сильно влияет как раз на качество модели. При слишком маленькой ширине модель сильно подстраивается под обучающую выборку и теряет свою обобщающую способность. При слишком большой ширине, напротив, модель становится слишком простой. Универсальной ширины окна не существует, поэтому для каждой задачи её приходится подбирать отдельно.

1.2 Наивный байесовский классификатор

Наивный Байес – это самый простой алгоритм, который можно применить к данным. Этот метод – это набор алгоритмов контролируемого обучения, основанный на применении теоремы Байеса с предположением об условной независимости между каждой парой характеристик при заданном значении переменной класса. Теорема Байеса утверждает следующее отношение, учитывая переменную класса y и зависимый вектор признаков x_1через x_n:

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) * P(x_1, \dots, x_n \mid y)}{P(x_1, \dots, x_n)}.$$

Используя наивное предположение об условной независимости,

$$P(x_i \mid y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i \mid y),$$

для всех i, это отношение упрощается до:

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) * \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}.$$

Тогда искомый объект:

$$\hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

Различные наивные байесовские классификаторы различаются в основном предположениями, которые они делают относительно распределения $P(x_i \mid y)$. Несмотря на упрощённые предположения, такие классификаторы хорошо работают во многих реальных ситуациях. Таким классификаторам требуется небольшой объём обучающих данных для оценки параметров.

К плюсам алгоритма можно отнести: быстрое предсказание класса тестового набора; алгоритм справляется с многоклассовым прогнозированием; производительность алгоритма гораздо выше, чем у многих других простых алгоритмов и требует меньшего объёма обучающих данных; метод хорошо работает с категориальными признаками. К минусам относятся: при наличии у переменной категории, которая не наблюдалась в обучающем наборе данных, модель может присвоить ей нулевую вероятность, из-за чего она не сможет сделать предсказание; значения спрогнозированных вероятностей не всегда являются достаточно точными; ограничением данного алгоритма является предположение о независимости признаков.

Так как алгоритм способен производить многоклассовое прогнозирование, он чаще всего применяется для систем рекомендаций, классификации текста, фильтрации спама и подобных задач.

1.3 Деревья принятия решений

Деревья решений – это непараметрический контролируемый метод обучения. Решающее дерево предсказывает значение целевой переменной с помощью применения последовательности простых решающих правил (которые называются предикатами). Этот процесс в некотором смысле

согласуется с естественным для человека процессом принятия решений. Обобщающая способность решающих деревьев невысока, их предсказания вычисляются просто, из-за чего решающие деревья часто используют как части моделей, делающих предсказания на основе агрегации предсказаний других моделей.

Зададим определение решающего дерева. Пусть задано бинарное дерево, в котором:

– каждой внутренней вершине v приписан предикат $B_v : \mathrm{X} \to 0, 1$;

– каждой листовой вершине v приписан прогноз $c_v \in \mathrm{Y}$, где Y – область значений целевой переменной.

В ходе предсказания осуществляется проход по этому дереву к какому-либо листу. Для каждого объекта выборки x движение начинается из корня. В какой-то из вершин v проход продолжится вправо, если $B_v(x) = 1$, и влево, при $B_v(x) = 0$. Проход по дереву продолжается до момента, пока не будет достигнут какой-либо лист. Ответом алгоритма на объекте x считается прогноз c_v, приписанный этому листу.

Предикат B_v может иметь произвольную структуру, но на практике используется просто сравнение с порогом $t \in \mathbb{R}$ по какому-то j-ому признаку. При проходе через узел дерева с данным предикатом объекты будут отправлены в левое поддерево, если значение j-го признака больше t, и в правое поддерево, если значение j-го признака у них меньше или равно t.

У дерева решений есть несколько свойств:

– дерево решений не сможет экстраполировать зависимости за границы области значений обучающей выборки;

– выученная функция является кусочно-постоянной, из-за чего производная равна нулю везде, где задана;

– дерево решений способно идеально приблизить обучающую выборку и ничего не выучить.

Как упоминалось выше, можно построить дерево, запоминающее всю выборку, однако на тестовых данных такой алгоритм не покажет высокого качества. Однако, можно поставить другую задачу: построить оптимальное с точки зрения качества на обучающей выборке дерево минимальной глубины, чтобы снизить переобучение. Проблема в том, что поиск такого дерева пока не имеет решения за полиномиальное время. Решить это можно двумя способами:

– Разрешить искать не оптимальное решение, а просто достаточно хорошее. Начать можно с того, чтобы строить дерево с помощью жадного алгоритма, то есть не искать всю структуру сразу, а строить дерево этаж за этажом. Тогда в каждой внутренней вершине дерева будет решаться задача, схожая с задачей построения решающего пня;

– Заняться оптимизацией с точки зрения computer science — наивную версию алгоритма (перебор наборов возможных предикатов и порогов) можно ускорить.

Ставится задача построить решающее дерево, наилучшим образом представляющее целевую зависимость. Однако, оптимизировать структуру дерева с помощью градиентного спуска невозможно, из-за обнуления производных.

Алгоритм построения решающего дерева жадным алгоритмом:

1) Создаём вершину v;

2) Если выполнен критерий остановки, то останавливаемся, объявляем эту вершину листом и ставим ей в соответствие ответ, после чего возвращаем её;

3) Иначе: находим предикат, который определит наилучшее разбиение текущего множества объектов X_m на две подвыборки X_l и X_r, максимизируя критерий ветвления;

4) Для X_l и X_r рекурсивно повторим процедуру.

1.4 Многоклассовая классификация

Многоклассовая классификация – это способность модели классифицировать входные данные на более чем два класса. Этот метод похож на бинарную классификацию, но позволяет расширить модель для более общих сценариев. Это актуально в сферах, где категории или классы могут быть множественными. Многоклассовая классификация – это задача машинного обучения, в которой модель обучается различать входные данные на несколько классов. Это значит, что у нас есть более двух классов (категорий) для классификации. В данном контексте, "класс" – это категория, в которую модель пытается отнести входные данные. Многоклассовая классификация имеет множество приложений, начиная от компьютерного зрения и распознавания образов до анализа текстовых данных. Важно отметить, что многоклассовая классификация является обобщением задачи бинарной классификации, где есть только два класса.

Существует несколько методов и подходов для решения задач многоклассовой классификации:

– One-vs-All: этот метод заключается в создании бинарных классификаторов для каждого класса;

– Softmax-регрессия: это обобщение логистической регрессии на многоклассовый случай. Он использует функцию softmax для преобразования выходов модели в вероятности принадлежности к разным классам;

– Деревья принятия решений и случайный лес: Деревья принятия решений разделяют данные на классы с помощью древовидной структуры, а случайный лес комбинирует несколько деревьев для улучшения точности;

– Нейронные сети: Глубокие нейронные сети, такие как сверточные нейронные сети (CNN) и рекуррентные нейронные сети (RNN), часто используются для многоклассовой классификации изображений и текстовых данных. Эти сети могут изучать сложные зависимости между данными и классами.

При данном методе модель обучается решать несколько задач одновременно. Вместо того, чтобы создавать отдельные модели для каждой задачи, многозадачные модели обучаются с учетом всех задач одновременно.

Преимущества данного метода:

– Многозадачные модели могут использовать общую информацию, извлеченную из данных, что может привести к более эффективному обучению и повышению обобщающей способности модели;

– Вместо обучения и поддержки нескольких отдельных моделей, многозадачные модели обучаются одновременно, что может быть более эффективным с точки зрения вычислительных ресурсов;

– Многозадачные модели могут лучше справляться с обобщением к новым данным и решать задачи, которые ранее не встречались;

– При добавлении новых задач или данных можно легко расширить многозадачную модель, в то время как для отдельных моделей потребуется отдельное обучение.

1.5 Метод опорных векторов

Метод опорных векторов (Support Vector Machines — SVM) — это набор контролируемых методов обучения, используемых для классификации, регрессии и обнаружения выбросов. С точки зрения классификации, это дискриминантный классификатор, который используется для поиска оптимальной гиперплоскости с целью классификации данных по различным классам. В двумерном пространстве эту оптимальную гиперплоскость можно представить как линию, разделяющую пространство на две части: где одна часть пространства содержит точки данных, которые принадлежат одному классу, а другая часть пространства содержит точки данных, принадлежащие другому. класс. Концепция линий, действующих как классификатор, верна только в том случае, если точки данных линейно разделимы. SVM также можно использовать для поиска оптимальной кривой, которая может

использоваться для классификации точек данных, которые нельзя разделить линейно.

Преимуществами данного метода являются:

– Эффективен в пространствах больших размеров;

– По-прежнему эффективен в случаях, когда количество измерений превышает количество образцов;

– Использует подмножество обучающих точек в функции принятия решений (называемых опорными векторами), поэтому это также эффективно с точки зрения памяти;

– Универсальность: для функции принятия решения могут быть указаны различные функции ядра. Предоставляются общие ядра, но также можно указать собственные ядра.

К недостаткам данного метода можно отнести:

– Если количество функций намного превышает количество выборок, избегайте чрезмерной подгонки при выборе функций ядра, и термин регуляризации имеет решающее значение;

– SVM не предоставляют напрямую оценки вероятностей, они рассчитываются с использованием дорогостоящей пятикратной перекрестной проверки;

2 Инструменты для задачи классификации

2.1 Python и библиотеки

2.1.1 Библиотеки Data Science

Решить задачу классификации можно с помощью науки о данных. К библиотекам Data Science относятся:

– NumPy: библиотека линейной алгебры, разработанная на Python. В библиотеку входят функции для работы со сложными математическими операциями линейной алгебры, алгоритмы преобразования Фурье и генерации случайных чисел, методы для работы с матрицами и n-мерными массивами;

– Pandas: библиотека с открытым исходным кодом, которая предлагает широкий спектр инструментов для обработки и анализа данных. С ее помощью вы можете читать данные из широкого спектра источников, таких как CSV, базы данных SQL, файлы JSON и Excel;

– Seaborn: библиотека визуализации, основанная на Matplotlib. С помощью этой библиотеки машинного обучения легко создавать определенные типы графиков, такие как временные ряды, тепловые карты (heat map) и графики «скрипками» (violin plot);

– SciPy: это набор основанных на NumPy пакетов, предоставляющих инструменты для научных расчетов через компьютер. Она включает подмодули, которые занимаются оптимизацией, преобразованиями Фурье, обработкой сигналов и изображений, линейной алгеброй и, среди прочего, статистикой.

С помощью этих библиотек можно провести классификацию данных и визуализировать результат. Однако такой способ классификации затрачивает тем больше времени, чем больше данных нужно исследовать.

2.1.2 Библиотека Scikit-learn

Scikit-learn – еще одна известная библиотека машинного обучения Python, с широким спектром алгоритмов кластеризации, регрессии и классификации. Она также отлично взаимодействует с другими научными библиотеками Python, такими как NumPy и SciPy. Эта библиотека поддерживает алгоритмы обучения как с учителем, так и без учителя. Вот список основных преимуществ данной библиотеки, делающих ее одной из самых предпочтительных библиотек Python для ML:

– снижение размерности;

– алгоритмы, построенные на решающих деревьях;

– построение решающих поверхностей;

– анализ и выбор признаков;

– обнаружение и удаление выбросов;

– продвинутое вероятностное моделирование;

– классификация и кластеризация без учителя.

Эта библиотека содержит множество готовых простых моделей машинного обучения, которые можно будет использовать для задачи классификации.

2.1.3 Библиотеки TensorFlow и Keras

TensorFlow – библиотека сквозного машинного обучения Python для выполнения высококачественных численных вычислений. С помощью TensorFlow можно построить глубокие нейронные сети для распознавания образов и рукописного текста и рекуррентные нейронные сети для NLP (обработки естественных языков). Также есть модули для векторизации слов (embedding) и решения дифференциальных уравнений в частных производных (PDE). Этот фреймворк имеет отличную архитектурную поддержку, позволяющую с легкостью производить вычисления на самых разных

платформах, в том числе на десктопах, серверах и мобильных устройствах. Основной козырь TensorFlow это абстракции. Они позволяют разработчикам сфокусироваться на общей логике приложения, а не на мелких деталях реализации тех или иных алгоритмов. С помощью этой библиотеки разработчики Python могут легко использовать AI и ML для создания уникальных адаптивных приложений, гибко реагирующих на пользовательские данные, например на выражение лица или интонацию голоса.

Keras — одна из основных библиотек Python с открытым исходным кодом, написанная для построения нейронных сетей и проектов машинного обучения. В этой библиотеке реализованы практически все автономные модули нейронной сети, включая оптимизаторы, нейронные слои, функции активации слоев, схемы инициализации, функции затрат и модели регуляризации. Это позволяет строить новые модули нейросети, просто добавляя функции или классы. И поскольку модель уже определена в коде, разработчику не приходится создавать для нее отдельные конфигурационные файлы. Также Keras можно использовать при работе со сверточными нейронными сетями. В нем реализованы алгоритмы нормализации, оптимизации и активации слоев. Keras не является ML-библиотекой полного цикла (то есть, исчерпывающей все возможные варианты построения нейронных сетей). Вместо этого она функционирует как очень дружелюбный, расширяемый интерфейс, увеличивающий модульность и выразительность (в том числе других библиотек).

Эти библиотеки позволяют строить нейронные сети, которые также эффективны для задачи классификации.

2.2 Пакет STATISTICA

STATISTICA – это универсальная компьютерная интегрированная система, предназначенная для статистического анализа и визуализации

данных, а также разработки пользовательских приложений, содержащая широкий набор процедур анализа для применения в научных исследованиях, технике и бизнесе.

Данные в STATISTICA организованы в виде электронной таблицы. Таблица с исходными данными является одним из типов документа в системе STATISTICA. Каждый тип документа выводится в своём окне в рабочей области системы. Как только это окно становится активным, изменяется панель инструментов и меню. В нём появляются команды, доступные для этого типа документов.

Отличительной чертой системы STATISTICA является то, что в ней реализован так называемый графически-ориентированный подход к анализу данных. Смысл подхода состоит в том, чтобы получать всестороннее визуальное представление данных на всех этапах статистической обработки и на основе этого представления выбирать следующий шаг анализа. Для отображения результатов статистической обработки используются разнообразные графики: гистограммы, диаграммы рассеяния, круговые диаграммы, вероятностные графики, графики поверхностей и многие другие. Помимо общих статистических и графических средств, в системе имеются специализированные модули, например, для решения инженерно-технических и, в частности, промышленных задач: карты контроля качества, анализ процессов и планирование эксперимента. Работа со всеми модулями происходит в рамках единого программного пакета, для которого можно выбирать один из нескольких предложенных интерфейсов пользователя. С помощью реализованных в системе STATISTICA мощных языков программирования и языка макрокоманд, снабженных специальными средствами поддержки, пользователь может создать законченные статистические модули (процедуры) и встраивать их в различные другие приложения или вычислительные среды.

В этой программе реализованы многие статистические методы анализа данных, что позволяет решить простую задачу классификации, однако большое количестве исходных данных затрудняет эту задачу.

3 Решение задачи классификации

3.1 Оптимальный инструмент

Поставленную задачу классификации можно решить разными способами, однако наиболее оптимальным будет построение модели и её обучение с помощью языка Python. В силу большого объёма начальных данных обрабатывать их вручную будет гораздо тяжелее, чем если бы эту работу выполнила обученная модель или нейронная сеть. Также в Python есть возможность удобно работать со строками, что значительно упростит задачу обработки текстовых данных. Плюсом будут библиотеки вроде Pandas, которая позволит работать с csv-таблицами, и Seaborn, позволяющая визуализировать обработанные данные.

3.3 Оптимальный метод

Разные постановки задач требуют разных методов решения и единого метода, подходящего для всех задач, нет. Для задач классификации текста наиболее часто используют метод опорных векторов, метод K-ближайших соседей и наивный байесовский метод. Для задач классификации изображений обычно используют свёрточные нейросети. Наиболее точные результаты даёт градиентный бустинг, однако не всегда его можно использовать.

Таким образом можно прийти к выводу, что, чтобы определить оптимальные методы для конкретной задачи, нужно исходить из заданных начальных данных и постановки самой задачи, так как не все методы будут одинаково эффективны при определённых условиях. Так, например метод K-ближайших соседей чувствителен к масштабу данных и к неинформативным признакам, а наивный байесовский алгоритм, если категория какой-либо категориальной переменной не видна в наборе обучающих данных,

присваивает этой категории нулевую вероятность, и тогда прогноз не может быть сделан.

ЗАКЛЮЧЕНИЕ

Наиболее часто используемыми методами машинного обучения для решения задач классификации являются метод K-ближайших соседей, наивный байесовский метод, деревья принятия решений (метод случайного леса), многоклассовая классификация, метод опорных векторов, также можно включить сюда градиентный бустинг. Однако, как удалось установить, для решения задач классификации нет единого метода, который будет давать наилучший результат при разных постановках и входящих данных. У каждого метода есть свои плюсы и минусы, которые стоит учитывать при принятии решения о том, какой алгоритм использовать.

Инструментами для решения задачи классификации являются программирование на языке Python и использование программы Statistica. Несмотря на то, что Statistica позволяет решать задачи классификации и даже визуализировать эти решения, она имеет проблемы с большим объёмом данных. Язык Python, в свою очередь, предоставляет возможности обработки входных данных, построении и обучении модели машинного обучения и визуализации полученных результатов, поэтому он лучше подойдёт для решения задачи классификации с большим объёмом входных данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Берман, К. Основы Python для Data Science : учебное пособие / К. Берман. – СПб. : Питер, 2023. — 272 с. – ISBN 978-5-4461-2251-6

2. Будума, Н. Основы глубокого обучения. Создание алгоритмов для искусственного интеллекта следующего поколения : учебное пособие / Н. Будума, Н. Локашо. – Москва : Манн, Иванов и Фербер, 2020. — 304 с. – ISBN 978-5-00146-472-3

3. Бурков, А. Машинное обучение без лишних слов : учебное пособие / А. Бурков. – СПб. : Питер, 2021. – 192с. ISBN 978-5-4461-1560-0

4. Вейдман, С. Глубокое обучение: легкая разработка проектов на Python : учебное пособие / С. Вейдман. – СПб. : Питер, 2021. — 272 с. – ISBN 978-5-4461-1675-1

5. Вьюгин, В.В. Математические основы машинного обучения ипрогнозирования : учебное пособие / В.В. Вьюгин – Москва : МЦНМО, 2022. – 484 с. – ISBN 978-5-4439-2014-6

6. Мюллер, А.П. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными : учебное пособие / А.П. Мюллер, С. Гвидо. – Москва : Вильямс, 2022. – 480с. – ISBN 978-5-9908910-8-1

7. Орельен, Ж. Прикладное машинное обучение с помощью Scikit-Learn, Keras и TensorFlow. Концепции, инструменты : учебное пособие / Ж. Орельен. – Москва : Диалектика, 2020г. – 1040с. – ISBN 978-5-907203-33-4

8. Постолит, А. В. Основы искусственного интеллекта в примерах на Python : учебное пособие / А.В. Постолит. – СПб. : БХВ-Петербург, 2021. – 448 с. – ISBN 978-5-9775-6765-7

9. Протодьяконов, А. В. Алгоритмы Data Science и их практическая реализация на Python : учебное пособие / А.В. Протодьяконов, П.А. Пылов, В.Е. Садовников – Москва : Инфра-Инженерия, 2022. – 392 с. – ISBN 978-5-9729-1006-9

10. Рашид, Т. Создаём нейронную сеть : учебное пособие / Т. Рашид – СПб. : “Альфа-книга”, 2017. – 272 с. – ISBN 978-5-9909445-7-2

11. Рашка, С. Python и машинное обучение : учебное пособие / С. Рашка, В. Мирджалили. – Москва : ДМК-Пресс, 2020. – 420 с. – ISBN 978-5-97060-409-0

12. Саттон, Р.С. Обучение с подкреплением : учебное пособие / Р.С. Саттон, Э.Г. Барто. – Москва : ДМК-Пресс, 2020. – 552с. – ISBN 978-5-00101-456-0

13. Траск, Э. Грокаем глубокое обучение : учебное пособие / Э. Траск — СПб. : Питер, 2019. — 352 с. – ISBN 978-5-4461-1334-7

14. Флах, П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных : учебное пособие / П. Флах. – Москва : ДМК-Пресс, 2021. – 402с. – ISBN 978-5-97060-273-7

15. Michie, D. Machine Learning, Neural and Statistical Classification / D. Michie, D. J. Spiegelhalter. – New York : Ellis Horwood, 2020. – 289с. – ISBN 9780131063600