

Волгоградский государственный университет
Институт математики и информационных технологий
Кафедра информационных систем и компьютерного моделирования

Работа допущена к защите

Заведующий кафедрой

_____ А. В. Хоперсков

«_____» _____ 2022 г.

Бут Александр Андреевич

Использование классификаторов в машинном обучении

Отчет по учебной практике, научно-исследовательской работе

Направление: 09.03.04 – Программная инженерия

Студент группы ПРИб-201:

А. А. Бут

подпись

Руководитель практики:

С. С. Хохлова, к.ф.-м.н., доцент каф. ИСКМ

подпись

Ответственный за организацию практики:

В. В. Юданов, к.ф.-м.н., доцент каф. ИСКМ

подпись

Содержание

Введение	4
Глава 1. Теория машинного обучения	6
1.1 Выбор языка программирования	6
1.2 Общие сведения	7
1.3 Построение классификатора	8
1.4 Матрица запутанности	9
1.5 Чувствительность, специфичность и эффективность алгоритма	10
Глава 2. Классификаторы машинного обучения	11
2.1 Метод ближайших соседей	11
2.2 Наивный байесовский классификатор	12
2.3 Деревья решений	13
2.4 Логистическая регрессия	14
2.5 Случайный лес	15
2.6 Метод опорных векторов	16
Глава 3. Практическая часть	18
3.1 Работа с базой данных	18
3.1.1 Подготовка данных	18
3.1.2 Предобработка данных	19
3.2 Разработка приложения	20
3.2.1 Изучение библиотек и основных операторов	20
3.2.2 Написание программного кода	21
3.3 Результаты	27
3.3.1 Статистическая обработка базы данных	27
3.3.2 Результаты работы программы	29

Заключение	32
Приложение А. Листинг разработанной программы	37

Введение

Машинное обучение - одна из самых перспективных областей современной индустрии информационных технологий. Алгоритмы данной сферы программирования способны решать многие задачи, которые человек либо не способен решить, либо хочет оптимизировать и автоматизировать. Компьютерное мышление в настоящие дни внедряется во многих процессах - от таргетированной коммерческой рекламы до прогнозирования прогрессирования раковых опухолей в медицине. Позволяя решать подобные задачи вычислительной технике, общество извлекает из всего этого пользу, несоизмеримо высокую, в сравнении с собственными затраченными ресурсами. Проводя аналогии, обучение искусственного интеллекта на примере многообразия схожих задач напоминает прогресс общества, когда поколение от поколения люди перенимали знания и опыт предков и передавали потомкам, однако протекающий в гораздо более краткие сроки.

Актуальность данной тематики изучения основ и применения на практике классификаторов машинного обучения заключается в том, что достижение определенных целей с помощью машинного обучения возможно огромным количеством разных путей. Именно избрание наиболее подходящего в определенной ситуации, предварительно изучив всю специфику и сферы применения каждого из них является ключом к эффективному решению какой бы то ни было задачи.

Обратимся к более профилированному примеру. Возможность по набору определенных личных предпочтений предсказать направление подготовки при обучении в вузе, определяющее вектор развития для человека, как минимум на последующие четыре года, а зачастую и на значительно более длительный период – означает избавление от личных переживаний, мук выбора, а в глобальном плане – более корректное распределение ресурсов внутри страны, когда грамотные специалисты будут изначально утверждены на позиции,

на которых может быть максимально раскрыт их потенциал.

Целью работы является сравнительный анализ результатов работы нескольких классификаторов для решения задач анализа данных. Задачи:

1. Сбор и изучение литературы по теме исследования.
2. Рассмотрение линейных моделей для классификации данных.
3. Провести сравнительный анализ результатов работы нескольких классификаторов для решения задач анализа данных.

Глава 1

Теория машинного обучения

1.1 Выбор языка программирования

Язык, выбираемый для разработки в сфере искусственного интеллекта и машинного обучения, должен соответствовать нескольким очевидным критериям [1] - быть гибким, стабильным и содержать расширенный спектр инструментов, в том числе, для работы с большим объемом данных.

Язык программирования Python, выбранный для данной научно-исследовательской работы, избирается в настоящее время большим числом специалистов для создания ИИ-систем, этим же широким сообществом пользователей в полной мере поддерживается и дорабатывается, в связи с чем, машинное обучение стало одной из основных отраслей специализации данного языка. Он содержит обширный стек библиотек:

- Keras, TensorFlow, и Scikit-learn - непосредственно для машинного обучения;
- NumPy - для вычислений научного характера;
- SciPy - для вычислений более высокого уровня;
- Pandas - для анализа данных;
- Seaborn - для их визуализации;

В то же время, Python лаконичен и прост в освоении, что позволяет, практически без трат времени на освоение синтаксиса, сразу же акцентировать внимание именно на разработке систем и общей структуре отдельных библиотек.

Таким образом, используя язык программирования Python, можно через минимальный промежуток времени преодолеть порог вхождения при понимании лингвистической специфики, технических аспектов и сосредоточиться

на изучении отдельных алгоритмов, что, как раз, и требуется новичку, желающему начать изучать машинное обучение.

1.2 Общие сведения

Машинное обучение - категория алгоритмов, позволяющая программам предсказывать результаты без явного программирования [2]. Возникло оно в 60-х годах прошлого века, когда исследователи были заинтересованы в создании искусственного интеллекта. Практически сразу от подобных систем требовалось обучение на данных. К этой проблеме пытались подойти с помощью различных методов [3]. Алгоритмы машинного обучения делятся на обучение с учителем и без. Для первого необходимо предоставить качественные данные для ввода, желаемый результат, а также наладить обратную связь о точности предсказаний. Два основных метода обучения с учителем:

- Классификация

Группировка объектов по заранее установленным признакам, самые популярные методы классификации - алгоритм «наивного Байеса» и метод опорных векторов.

- Регрессия

Применяется, когда нужно спрогнозировать результат и записывается в виде уравнения взаимосвязи входных и выходных данных. Часто используемые алгоритмы - линейная, полиномиальная и логистическая регрессия.

При обучении без учителя осуществляется поиск взаимосвязей между объектами входных данных, когда полноценного однозначного ответа на выходе не требуется. Основные методы:

- Кластеризация

При кластеризации происходит деление входных данных на группы по признаку, который система выделяет сама при их анализе. Наиболее распространенные методы - DBSCAN и метод К-средних, применяемые, к примеру, в

работе с изображениями. [4]

- Снижение размерности

Используется для оптимизации скорости и расходования памяти другими методами ML, когда работа ведется с многомерными данными, сложными для визуализации и восприятия, чтобы обобщить все признаки до уровня выше. Примеры - алгоритм T-NSE, сингулярное разложение.

- Нейронные сети

Нейронные сети - особая математическая модель, применяемая в машинном обучении, построенная по аналогии с работой нейронов в человеческом мозге [5].

Персептрон

Математическая модель нейронной сети, состоящего из одного нейрона, который выполняет две последовательные операции [6]

- вычисляет сумму входных сигналов с учетом их весов (проводимости или сопротивления) связи

$$sum = \vec{X}^T \vec{W} + \vec{B} = \sum_{i=1}^n x_i w_i + b \quad (1.1)$$

- применяет активационную функцию к общей сумме воздействия входных сигналов.

1.3 Построение классификатора

Построение классификатора сложено из нескольких шагов [7].

1. Подготовка данных.

Изучается предметная область, касаясь которой будет вестись работа, составляется база данных из характерных ей объектов.

2. Предобработка данных.

Включает в себя отбор значимых для классификации признаков, трансфор-

мацию и очистку данных (исключение дубликатов, противоречивых или ошибочных экземпляров, нормализацию). Результат - получение оптимальной базы примеров, однозначно разделяемой на классы.

3. Разделение.

Весь набор данных делится на два множества: обучающее и тестовое (в отдельных случаях допускается использование третьего, валидационного, множества)

4. Выбор системы кодирования выходных данных. Примеры видов систем:

- «2 на 2»-кодирование
- классическое

5. Конструирование, обучение и анализ результатов. При конструировании выбирается топология сети, функции нейронов и т.д. Затем оценить качество работоспособности алгоритма на тестовой выборке. Убедиться в достижении необходимой точности и результативности. В случае несоответствия вернуться на предыдущий этап и изменить параметры.

1.4 Матрица запутанности

Матрица запутанности - особый объект, представленный в виде ограниченной двумя координатными осями таблицы значений. Используется для подведения итогов классификации и определения ее точности. Внутри сетки значений расположены верно и неверно предсказанные объекты классов[8].

В качестве примера рассмотрим матрицу запутанности, получаемую в результаты работы одного из алгоритмов, который будет описан ниже. На приведенном рисунке (Рис. 1.1) число в верхнем левом квадрате обозначает количество объектов, принадлежность к классу для которых определена верно, в правом - неверно. Для второго класса число в нижнем левом квадрате обозначает количество объектов, для которых класс определен неверно,

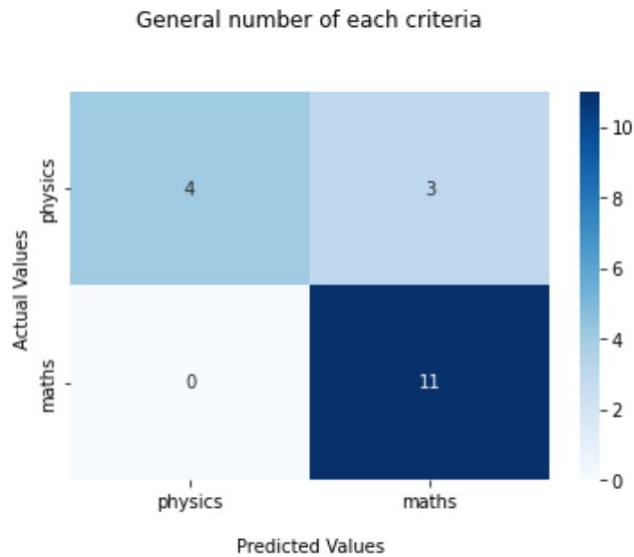


Рисунок 1.1. Матрица запутанности.

в правом - верно.

1.5 Чувствительность, специфичность и эффективность алгоритма

Существуют особые величины для оценки работы алгоритма, вычисляемые на основании матрицы запутанности [9].

- Чувствительность - отношение правильных предсказаний для объектов первого класса к общему количеству предсказаний этого класса.
- Специфичность - отношение правильных предсказаний для объектов второго класса к общему количеству предсказаний этого класса.
- Эффективность - квадратный корень из произведения чувствительности и специфичности.

Глава 2

Классификаторы машинного обучения

Рассмотрим алгоритмы классификации, применённые в данной работе.

2.1 Метод ближайших соседей

Алгоритм устроен следующим образом.

У нас есть тренировочная выборка, мы запоминаем ее объекты, и когда нам поступает новый объект – мы осуществляем поиск наиболее похожих на него объектов выборки среди ближайших к нему и смотрим, какой ответ доминирует – 0/1 и выдадим этот класс в качестве ответа. Другими словами, поступивший объект определяется как доминирующий класс среди объектов, более всего на него похожих. Для определения ближайших к объекту соседей предусмотрены различные метрики, выбирающиеся как параметр [10]:

- Евклидова метрика

$$\rho(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (2.1)$$

- Расстояние Чебышева

$$\rho(x, y) = \max_{i=1, \dots, n} |x_i - y_i| \quad (2.2)$$

- Манхэттенское Расстояние

$$\rho(x, y) = \sum_{i=1}^n |x_i - y_i| \quad (2.3)$$

Поскольку наши данные изначально имеют одинаковую физическую природу, а, следовательно и удельный вес – выборка с помощью евклидова расстояния подходит нам более чем. Число рассматриваемых соседей выбирается самостоятельно и является гиперпараметром алгоритма. Метод

ближайших соседей максимально толерантен к резким границам между классами, т.е. способен предугадывать принадлежность нового объекта к определенному классу с минимальным участием границы, это происходит в связи с тем, что он оперирует максимальным локальным максимальным правдоподобием. Отсюда и проблемы: точность очень сильно зависит от числа соседей, например, если взять все элементы за выборку – метод просто переобучится и возьмет за вероятность долю элементов.

2.2 Наивный байесовский классификатор

Одна из самых простых моделей для решения задач классификации, однако в некоторых простых задачах способна давать достаточно точные и качественные результаты. Например, в задаче фильтрации спам-писем, практически все алгоритмы работают на основе байесовского.

Томас Байес ввёл понятие вероятности наступления определенного события x , при условии, что событие y уже случилось[11]:

$$P(x|y) = \frac{P(x, y)}{P(y)} \quad (2.4)$$

Из данного соотношения была выведена другая формула:

$$P(x, y) = P(x|y) * P(y) = P(y|x) * P(x) \quad (2.5)$$

Вероятность наступления события x получается путем умножения вероятности наступления события y на вероятность наступления события x , при условии, что y случилось. Отсюда возник финальный вид формулы Байеса:

$$P(x|y) = \frac{P(y|x) * P(x)}{P(y)} \quad (2.6)$$

Таким образом, для каждого объекта высчитывается данная вероятность, определяющая, насколько возможна принадлежность нового объекта к тому или иному классу.

2.3 Деревья решений

Один из самых старых алгоритмов, но до сих пор достаточно популярный. Симулирует процесс, который ранее принимали люди. Например, в прошлом, сотрудник банка задавал клиенту, желающему взять кредит, вопросы, в зависимости от ответа на каждый из которых, выбирался следующий вопрос, исходя из чего принималось решение предоставить займ или нет.

Алгоритм автоматизирует процесс, система учитывает исторические данные прецедентов, работает со всеми предоставленными параметрами по объекту, например, вернувшись к примеру выше, ей были бы предоставлены данные об уровне дохода, наличии постоянной работы и т.д. Затем, она находит по каким данным можно сепарировать объекты на множества. Существует два подхода к делению объектов на группы:

- Метод максимизации энтропии
- Метод хи-квадрата

Машина сама будет выбирать один из признаков, расщеплять данные и приходить к решению. Решения принимаются с некоторой вероятностью, чем она выше - тем лучше точность предсказания.

Достоинства классификатора [12]:

- Интерпретируемость и наглядность
- Производительность

Недостаток классификатора:

- Сложность контроля размера дерева

Алгоритм относится к числу «жадных» - то есть допускающих, что локальные решения на каждом шаге приводят к оптимальному результату. В частности, для деревьев решений это означает, что если на каком-либо шаге по определенному атрибуту уже произошло разделение, повернуть назад и выбрать другой нельзя. Потому невозможно изначально сказать как на итоговый результат повлияет тот или иной атрибут.

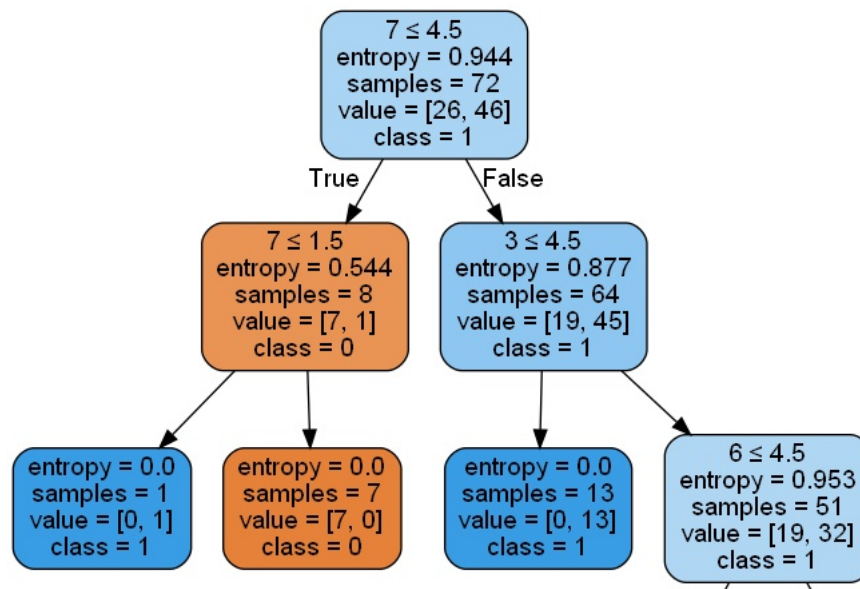


Рисунок 2.1. Кадр дерева решений из программы.

Дерево решений представляет собой древовидную структуру из внутренних и внешних вершин и соединяющих их ребер. Внутренние узлы рассчитывают значение функции решения, на основании которого определяется узел для дальнейшего посещения. Внешние характеризуют входные данные и не имеют дочерних узлов. Таким образом, весь алгоритм состоит в следующем:

- Передаются данные на корневой узел дерева.
- В зависимости от полученного значения осуществляется переход к дочернему узлу.
- Переходы продолжаются до тех пор, пока не будет достигнут конечный узел с меткой класса.

В основе обучения деревьев решений лежит принцип «разделяй и властвуй» [13].

2.4 Логистическая регрессия

Алгоритм является подвидом множественной регрессии. Выходная характеристика объекта может быть

- Категориальной

- Бинарной (избираться из множества, содержащего два значения)

Во втором виде выходной переменной объект может принадлежать определенному классу на выходе - А или Б, каждый из исходов является соответствующим событием. После сравнения вероятностей событий, получается результат, являющийся меткой класса (А или Б), вероятность принадлежности к которому выше.

Преимущество метода - наличие ROC-анализа для определения качества моделей.

2.5 Случайный лес

В основе лежит построение ансамбля деревьев решений, каждое из которых строится путём выборки с возвращением (бутстрепа) [14], примененного к исходной. Число таких деревьев - гиперпараметр и может быть выбрано самостоятельно. Отличия от простого построения дерева решений:

- Используется константное число определенных признаков.
- Строится полное дерево, а не усеченное.
- Оценка регрессии и классификация на основе выводов по всем деревьям.

Общий алгоритм можно представить в следующем виде:

Для каждого из деревьев

1. Формируется бутстреп выборка исходной.
2. Индуцируется неусеченное дерево по следующему подалгоритму:
 - а) из n признаков выбирается случайно p признаков
 - б) из p выбирается обеспечивающий оптимальное расщепление
 - с) выборка расщепляется на две
3. В результате получается ансамбль деревьев решений, по которому осуществляется предсказание.

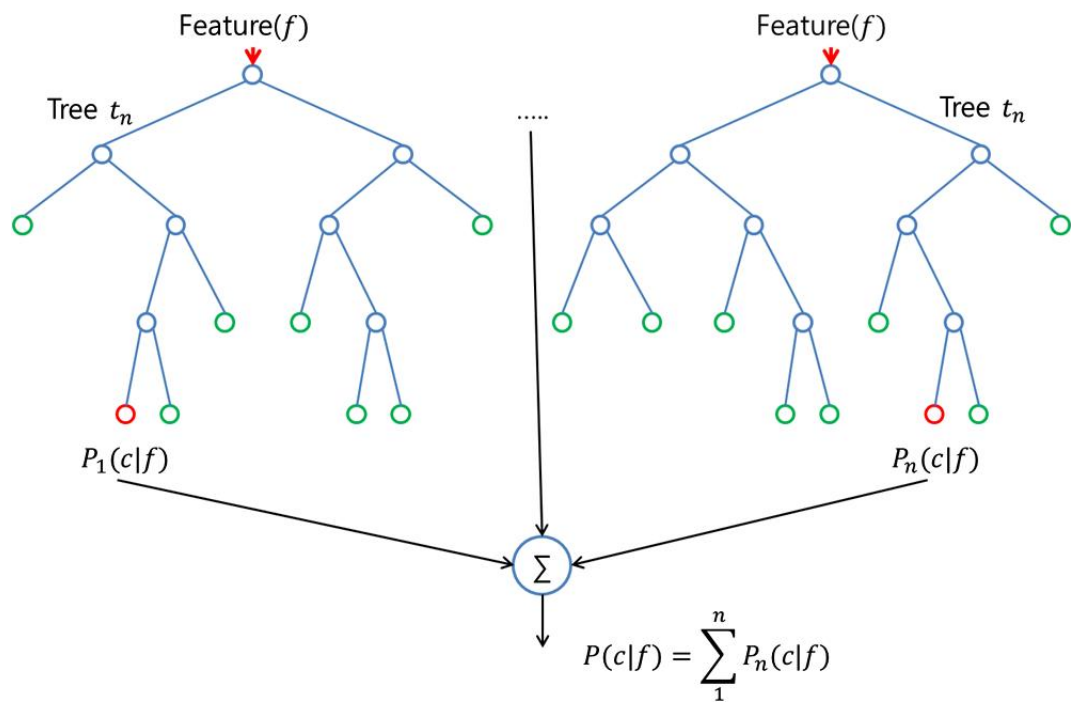


Рисунок 2.2. Пример ансамбля деревьев. [15]

Достоинства метода:

- Нет необходимости в тестовой выборке для оценки точности.
- Проблема усечения дерева, свойственная обычному методу деревьев решений отсутствует.
- Нет проблемы отбора признаков, свойственной другим методам с большим количеством деревьев.
- Простота применения.
- Возможность параллельных вычислений.

2.6 Метод опорных векторов

Главная задача метода - оптимальным путем разделить предоставленный набор данных [11]. Пространство между двумя ближайшими точками называется зазором [16]. Необходимо расположить гиперплоскость с максимально возможным зазором между опорными векторами. Поиск такой гиперплоскости осуществляется по следующему алгоритму:

1. Генерируются возможные гиперплоскости, делящие набор данных с

каким-либо зазором.

2. Из них выбирается такая, что при делении зазор максимален.

Некоторые задачи нельзя решить с помощью линейной гиперплоскости. В такой ситуации используется ядерный трюк для преобразования входного пространства в пространство более высокого измерения. Таким образом, входные данные просто преобразуются в вид, пригодный для разделения по вышеописанному алгоритму. Виды так называемых «ядерных трюков»:

- Линейное ядро
- Полиномиальное ядро
- Ядро радиальной базисной функции

Глава 3

Практическая часть

3.1 Работа с базой данных

3.1.1 Подготовка данных

Для исследования была выбрана база данных, которая состоит из результатов опроса первокурсников, поступивших в 2021 году, и у которых уже сложилось некоторое мнение об обучении в университете на их собственной специальности.

Для привлечения прохождения опроса были привлечены следующие группы обучающихся с первого курса института математики и информационных технологий: программная инженерия, информационные системы и технологии, математическое обеспечение и администрирование информационных систем, прикладная математика и физика, радиотехника, радиофизика, математика и компьютерные науки, прикладная математика и информатика, информатика и вычислительная техника, прикладная информатика, прикладная математика и информатика.

A		B		C	D	E
1	Отметка времени	1. ФИО	2. Ваше направление обучения (полное название)	3. Сумма результатов трёх ЕГЭ	4. Часто ли вы проводите время за компьютером?	
2	11.30.2021 11:52:20	Устарханов Артем Уллубиевич	МОС		213	
3	12.1.2021 15:17:09	Земченко Егор Андреевич	МОС		168	
4	12.10.2021 19:28:14	Васильев Максим Сергеевич	ПМФ		164	
5	11.28.2021 12:03:27	Гувазов Руслан Гуматович	ИСТ		190	
6	11.30.2021 11:09:09	Устиновская Марина Андреевна	ПРИ		264	
7	11.30.2021 13:15:15	Горбаченко Юлия Анатольевна	МОС		181	
8	11.28.2021 12:17:52	Панафидин Михаил Николаевич	ИСТ		219	
9	11.28.2021 12:21:19	Омельченко Олеся	ПРИ		225	
10	11.30.2021 11:43:07	Олейник Алена Александровна	МОС		234	
11	11.30.2021 13:00:24	Зверев Виталий Михайлович	МОС		246	
12	11.30.2021 13:08:13	Пономарев Александр Сергеевич	МОС		240	
13	11.30.2021 13:21:32	Савельев Андрей Леонидович	МОС		196	
14	12.10.2021 18:01:30	Мельников Иван Николаевич	ПМФ		211	
15	11.28.2021 11:05:45	Лиджиев Владимир Минигиевич	ИСТ		220	
16	11.28.2021 12:02:00	Фролова Юлия Алексеевна	ИСТ		265	
17	11.28.2021 12:50:10	Агапченко Олег Андреевич	ИСТ		199	
18	11.30.2021 11:10:03	Резанов Кирилл Владимирович	ПРИ		238	
19	11.30.2021 11:13:55	Астахов Данила Александрович	ПРИ		210	
20	11.30.2021 16:18:16	Ракчеев Никита Сергеевич	ПРИ		246	
21	11.30.2021 11:43:36	Чарнищев Олег Сергеевич	МОС		222	
22	11.30.2021 11:45:47	Крайнев Никита Евгеньевич	МОС		224	
23	11.30.2021 11:51:01	Макарова Елена Вячеславовна	МОС		222	
24	11.30.2021 11:59:42	Кутышенко Кирилл Дмитриевич	МОС		210	
25	11.30.2021 12:15:17	Баркова Анастасия Сергеевна	МОС		265	
26	11.30.2021 13:00:58	Легина Ксения Александровна	МОС		198	
27	12.1.2021 15:18:54	Керимов Артур Рустамович	МОС		174	
28	12.10.2021 17:03:51	Сиваева Артем Алашович	ПМФ		188	
29	12.10.2021 17:08:24	Романов Давид Игоревич	ПМФ		175	
30	11.28.2021 11:20:02	Косиков Александр Сергеевич	ИСТ		207	
31	11.30.2021 11:09:54	Ульянов Анастасия Сергеевна	ПРИ		266	
32	11.30.2021 13:13:12	Живолупов Вадим Дмитриевич	МОС		182	
33	12.1.2021 16:40:56	Звездин Антон Валентинович	МОС		208	
34	12.1.2021 17:11:18	Кариков Климент Владимирович	МОС		235	
35	11.30.2021 11:20:39	Шелепов Матвей Игоревич	ПРИ		230	
36	11.30.2021 11:27:58	Хусаев Виталий Константинович	ПРИ		228	
37	11.30.2021 11:50:06	Полова Анастасия Дмитриевна	МОС		259	
38	11.30.2021 11:53:22	Гуренкова Милана Владиславовна	МОС		202	

Рисунок 3.1. Кадр excel-документа базы данных.

3.1.2 Предобработка данных

Одним из самых важных шагов при построении классификатора является предобработка данных. На этом этапе изучается перечень объектов базы с их характеристиками и основываясь на общих умозаклчениях, сделанных на шаге подготовки, проводится работа по удалению несоответствующих общей логике элементов. Другими словами, мы смотрим на базу данных, ее свойства и предопределяем, каким образом тот или иной ее объект повлияет на обучение системы. Если сразу понятно, что элемент в той или иной степени мешает алгоритму вывести общую закономерность – он исключается. Назовем такой элемент ненадежным, поскольку он потенциально может обучить систему неправильно.

Идеальный результат, к которому должна прийти система в этой работе – исходя из ответов на анкетирование с максимальной точностью определять на какую специальность поступил человек. Если объект из базы будет содержать противоречия, к примеру, в вопросе «Нравится ли вам физика?» респондент выставил баллов меньше, чем в вопросе «Нравится ли вам математика?», но при этом поступил на физическую специальность (Радиофизика, Прикладная математика и физика) – он внесет свой вклад в обучение системы неправильным образом, в соответствии с этими противоречивыми данными. Точно также, анкетлируемые с низкими баллами могут повлиять не на обучение должным образом, поскольку часть из них поступала с доминирующим влиянием не собственных предпочтений, а сложившихся обстоятельств. И наконец, были исключены данные, содержащие в тех же вопросах про симпатию к отдельным школьным предметам одинаковые баллы, поскольку данные элементы содержат не перевес в одну из сторон, а пятидесятипроцентную вероятность.

Таким образом, в изначальной базе с 91 абитуриентом их осталось

61	12.10.2021 18:38:37	Вайт Владимир Витальевич	ПМБ	168	10	10	10	5	10	10	9	10	5	5	10	10	10	10	10
62	11.30.2021 12:04:41	Мамкин Андрей Михайлович	МОС	224	10	3	4	6	8	3	1	10	6	6	8	5	8	9	10
63	12.10.2021 17:20:24	Никитина Татьяна Алексеевна	ПМБ	169	10	8	8	7	9	8	1	10	7	7	6	6	9	9	10
64	12.1.2021 16:55:14	Манкосов Данил Михайлович	МОС	205	9	7	9	8	9	3	8	8	8	9	9	9	8	9	9
65	12.1.2021 17:04:27	Ферина Мария Вячеславовна	МОС	175	8	4	6	8	9	9	5	10	8	7	8	8	8	10	10
66	12.10.2021 17:10:16	Любименко Григорий Геннадьевич	ПМБ	180	8	10	10	8	8	3	10	6	8	8	8	8	3	10	6
67	11.28.2021 11:13:14	Ананьин Клим Вячеславович	ИСТ	216	8	6	9	9	10	3	7	10	9	9	9	7	7	10	8
68	12.10.2021 19:20:06	Федотов Михаил Алексеевич	ПМБ	253	8	8	6	10	3	1	10	5	10	6	5	7	2	7	6
69	12.1.2021 16:30:14	Заварзин Владимир Александрович	Радиотехника	169	10	2	3	7	4	1	3	10	8	10	8	8	10	10	10
70	12.1.2021 16:41:27	Солдатовичев Артем Викторович	Радиотехника	258	7	4	7	8	3	1	7	7	6	8	7	7	7	7	7
71	12.1.2021 16:43:38	Радлова ДЮ	Радиотехника	203	10	6	5	8	10	1	3	7	6	10	10	10	10	10	8
72	12.1.2021 16:45:29	Васильченко Алексей Дмитриевич	Радиотехника	209	8	4	8	8	8	9	1	10	9	5	10	10	9	10	8
73	12.1.2021 16:51:59	Цыпкина Кристина Николаевна	Радиотехника	191	6	4	3	8	4	6	1	7	7	4	5	5	5	5	5
74	12.1.2021 16:36:40	Киселев Семен Алексеевич	Радиотехника	178	10	1	8	9	7	3	5	10	3	10	9	6	9	10	10

Рисунок 3.2. Кадр excel-документа обработанной базы данных.

63. База подготовлена к загрузке в систему. Обучим систему и на изначальной базе, и на отредактированной.

3.2 Разработка приложения

3.2.1 Изучение библиотек и основных операторов

Как было сказано выше, язык программирования Python предоставляет широкий инструментарий для машинного обучения, для работы с данными и их графической визуализацией. Рассмотрим библиотеки, которые были подключены и использовались в данной работе.

- pandas.

Библиотека и непосредственно ее функция `read_excel()` использовалась непосредственно для чтения базы Excel объектов.

- sklearn.

Основная библиотека, содержащая классификаторы, такие как `KNeighborsClassifier` (классификатор ближайших соседей), `DecisionTreeClassifier` (классификатор деревьев решений), `GuassainNB` (наивный байесовский классификатор), `RandomForestClassifier` (классификатор случайного леса), `LogisticRegression` (логистическая регрессия) и `svm`, в частности класс `SVC` (метод опорных векторов). Также содержит оператор `confusion_matrix()`, на основании параметров `y_true` и `y_pred` (каноничный и предсказанный результаты) вычисляющий матрицу запутанности, показывающую, насколько метод точен и оператор `train_test_split()`, осуществляющий деление входных данных на тестовую выборку и обучающую.

Для визуализации результирующего графа метода деревьев решений эта библиотека содержит функцию `export_graphviz()`.

- `seaborn`

Функция `heatmap` модуля `seaborn` использовалась для создания цветной матрицы запутанности.

- `matplotlib`

Функция `plt.show()` использовалась для вывода отрисованной с помощью `heatmap`-функции матрицы на экран.

3.2.2 Написание программного кода

Процесс написания программы был начат с подключения основных библиотек для прочтения `excel`-файла с данными, функции которых были описаны выше.

Листинг 3.1. Подключение библиотек

```
1 import math
2 import pandas as pd
3 import numpy as np
4 import seaborn as sns
```

Далее были прочитаны и, для проверки корректности чтения, были выведены на экран данные файла.

Листинг 3.2. Чтение `excel`-файла

```
1 df = pd.read_excel('preliminary.xlsm')
2 print(df)
```

Были созданы необходимые для работы переменные, такие как переменная количества итераций, матрица для результатов работы всех алгоритмов, массив, содержащий ответы студентов на каждый из вопросов (для последующей постройки по нему графика «ящик с усами»), в который соответствующие данные были сразу записаны.

Листинг 3.3. Задание глобальных переменных

```
1 n = 5000
2 characteristic_table = [['KNN'], [0],[0],[0]],
3                        [['DT'], [0],[0],[0]],
4                        [['NB'], [0],[0],[0]],
5                        [['RF'], [0],[0],[0]],
6                        [['LR'], [0],[0],[0]],
7                        [['SVC'], [0],[0],[0]]
8 method_id = 0
9 every_question_base = []
10 j = 0
11 for i in range (4, 34):
12     every_question_base.append(np.array(df.iloc[:, i]))
13     print(every_question_base[j])
14     print('\n')
15     j+1
```

Отдельно была проведена работа по подключению библиотек, а затем созданию алгоритма, отрисовывающего матрицу запутанности в цвете. В нем графику был задан цвет градиента, подписаны оси.

Листинг 3.4. Создание алгоритма для матрицы запутанности

```
1 from sklearn.metrics import confusion_matrix
2 import matplotlib.pyplot as plt
3
4 def confusion_matrix_visualization(cm):
5     matrix = sns.heatmap(cm, annot=True, cmap='Blues')
6
7     matrix.set_title('General number of each criteria\n\n')
8     matrix.set_xlabel('\nPredicted Values')
9     matrix.set_ylabel('Actual Values ')
10
11     matrix.xaxis.set_ticklabels(['physics','maths'])
12     matrix.yaxis.set_ticklabels(['physics','maths'])
13
```

```
14 plt.show(matrix)
```

Далее была начата работа по написанию алгоритмов, работающих непосредственно с выходными данными, на результате работы которых строится вся данная исследовательская работа. Функция «асс»(accuracy) - на основании матрицы запутанности (cm) подсчитывает точность классификатора. Аналогично «sens»(sensitivity), «spec»(specificity) и «effect»(effectivity) - подсчитывают чувствительность, специфичность и эффективность алгоритмов соответственно. Функция «count_values» вызывает каждую из перечисленных последовательно и записывает результаты в массив, который и возвращает.

Листинг 3.5. Написание функций

```
1 def acc(cm):
2     total=sum(sum(cm))
3     accuracy=(cm[0,0]+cm[1,1])/total
4     return accuracy
5
6 def sens(cm):
7     sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
8     return sensitivity
9
10 def spec(cm):
11     specificity = cm[1,1]/(cm[1,0]+cm[1,1])
12     return specificity
13
14 def effect(cm):
15     effectivity = math.sqrt(sens(cm)*spec(cm))
16     return effectivity
17
18
19 def count_values(true, predicted):
20     cm = confusion_matrix(true, predicted)
21     print("ACCURACY: ", acc(cm))
22     values = [0, 0, 0]
```

```

23     values[0] = sens(cm)
24     values[1] = spec(cm)
25     values[2] = effect(cm)
26     return (values)

```

Следующим шагом, была проведена настройка входных данных, переменным «x» и «y» были переданы соответственно массив ответов студентов на вопросы и результирующий класс, к которому каждый студент относится. Здесь же был подключен модуль `train_test_split`, необходимый, как было описано выше, для разделения входных данных на тестовую и обучающую выборку.

Листинг 3.6. Предварительное разделение входной группы по переменным

```

1 x = df.values[:, 4:32]
2 x=x.astype('int')
3 y = df.values[:, 34]
4 y=y.astype('int')
5 print(x)
6 print(y)
7 from sklearn.model_selection import train_test_split

```

После всех перечисленных выше шагов входные данные были приведены в необходимый вид. Была начата работа непосредственно с классификаторами. Были предприняты следующие шаги для каждого из них:

1. Подключение.
2. Присвоение классификатору уникального идентификатора, необходимого в последующем для вывода его результирующих значений.
3. Создание локальных массивов результирующих значений.
4. Задание критериев классификатора.
5. В цикле на 5000 итераций в процессе каждой из них
 - . Входная группа делилась на тестовую и обучающую выборку в соотношении 20% к 80%.
 - . Классификатор обучался на обучающей выборке.
 - . Предсказывался результат для тестовой выборки.

. Фиксировалась чувствительность, специфичность и эффективность.

Далее, уже вне цикла

6. Выводились на экран средние чувствительность, специфичность и эффективность.

7. Эти же параметры записывались в общую характеристическую таблицу

Ниже часть программы, в которой велась работа с одним из классификаторов

Листинг 3.7. Работа с классификатором ближайших соседей

```
1 from sklearn.neighbors import KNeighborsClassifier
2 method_id = 0
3 values = [0, 0, 0]
4 sensitivity = []
5 specifity = []
6 effectivity = []
7 knn = KNeighborsClassifier(n_neighbors=9, weights="uniform",
    metric="euclidean")
8 for i in range(0, n):
9     x_train, x_test, y_train, y_test = train_test_split(x, y,
        test_size=0.20, stratify=y)
10    knn.fit(x_train, y_train)
11    predicted = knn.predict(x_test)
12    values = count_values(y_test, predicted)
13    print(values)
14    sensitivity.append(values[0]), specifity.append(values[1]),
        effectivity.append(values[2])
15 print(np.mean(sensitivity), np.mean(specifity), np.mean(
    effectivity))
16 characteristic_table[method_id][1] = np.mean(sensitivity)
17 characteristic_table[method_id][2] = np.mean(specifity)
18 characteristic_table[method_id][3] = np.mean(effectivity)
```

Отличия синтаксиса программы остальных классификаторов от приведенного заключаются в их номере идентификатора, а также самом названии как

классификатора, так и ответственной за его вызов переменной.

Листинг 3.8. Подключение классификатора деревьев решений

```
1 from sklearn.tree import DecisionTreeClassifier
```

Листинг 3.9. Присвоение переменной значения модуля классификатора

```
1 dtc = DecisionTreeClassifier(criterion = "entropy", max_depth =  
    5)
```

Листинг 3.10. Обучение классификатора

```
1 dtc.fit(x_train, y_train)
```

На основании хода предсказания классификатора деревьев решений был построен полный граф, визуализирующий процесс принятия решения, кадр которого был представлен в главе 2. Для этого использовались модули `display.Image` библиотеки `IPython`, `StringIO` библиотеки `six` и библиотека `pydotplus`.

Листинг 3.11. Построение графа деревьев решений

```
1 from sklearn.tree import export_graphviz  
2 from six import StringIO  
3 from IPython.display import Image  
4 import pydotplus  
5  
6 answer_cols = df.iloc[-1, 4:32]  
7  
8 dot_data = StringIO()  
9 export_graphviz(dtc, out_file=dot_data, filled=True, rounded =  
    True, special_characters=True, feature_names=answer_cols,  
    class_names = ['0', '1'])  
10 graph = pydotplus.graph_from_dot_data(dot_data.getvalue())  
11 graph.write_png('Students.png')  
12 Image(graph.create_png())
```

И также на основании данных об ответах на вопросы «Нравится ли вам математика?» и «Нравится ли вам физика» были построены графики «ящик с усами», визуализирующий распределение ответов на вопросы в зависимости от класса.

Листинг 3.12. Создание графика с усами

```
1 for i in range (0 , 2):  
2     ax = sns.boxplot(x=df.iloc[:, 34], y=every_question_base[i])  
3     plt.show()
```

Расширенный листинг программы представлен в приложении А.

3.3 Результаты

3.3.1 Статистическая обработка базы данных

Рассмотрим графики «Ящик с усами», получившиеся в конце программы, чтобы проанализировать входные данные.

Была создана визуализация ответов на шесть вопросов, которые нагляднее всего показывают различия между элементами двух разных классов (Рис. 3.3). Подписанные буквы обозначают определенный вопрос.

А) Вопрос 7. «Нравится ли вам математика?»

Б) Вопрос 12. «Нравится ли вам физика?»

В) Вопрос 14. «Нравится ли вам разбираться в принципе работы электронных устройств?»

Г) Вопрос 19. «Нравится ли вам решать математические задачи?»

Д) Вопрос 32. «Нравится ли вам паять схемы?»

Е) Вопрос 6. «Нравится ли вам решать головоломки?»

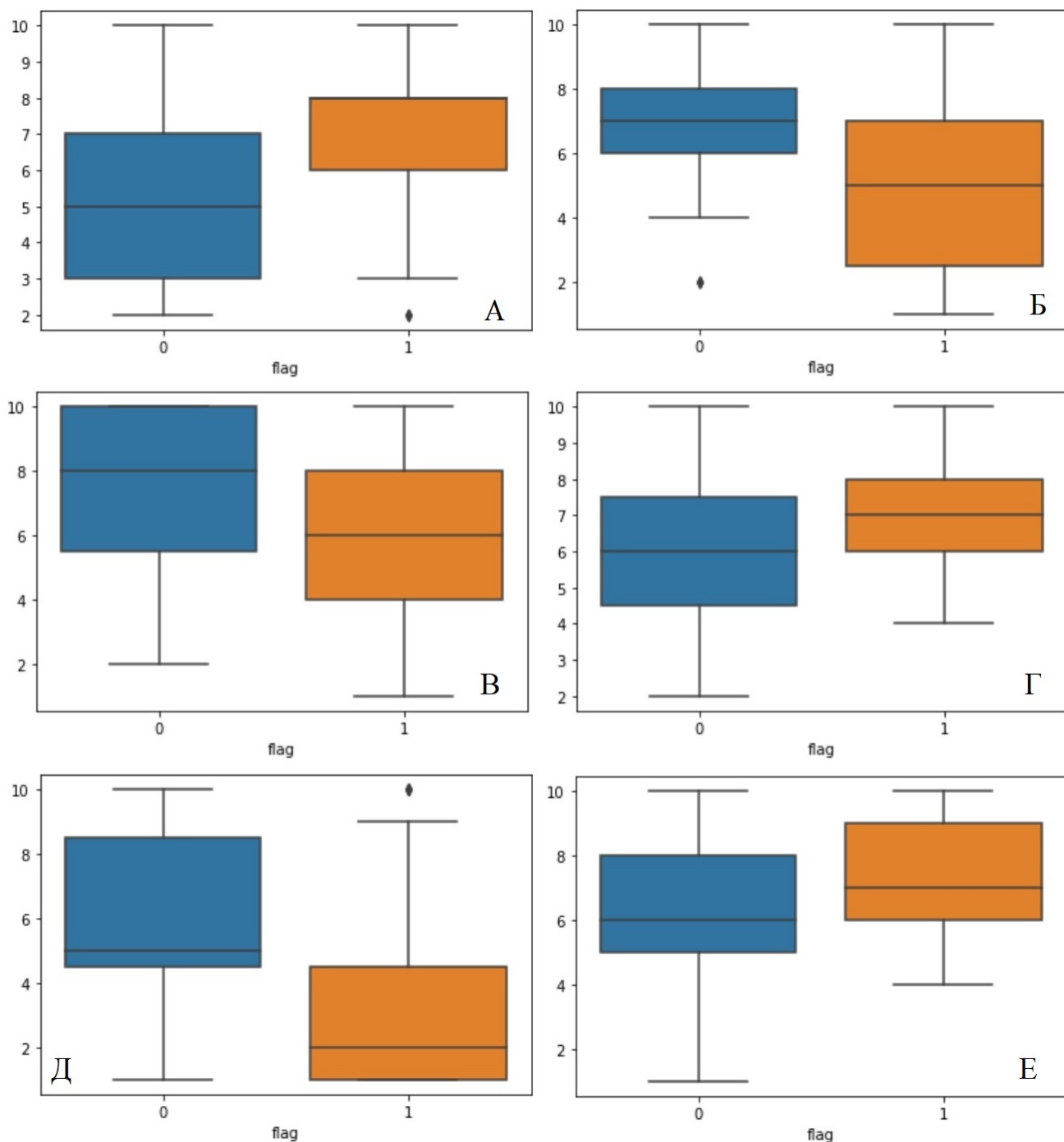


Рисунок 3.3. «Графики для шести вопросов»

Общие выводы

1. Наиболее оптимально сказаться на обучении должны были вопросы А, Б и Д, поскольку область схожести ответов на них минимальна.
2. Вопросы В, Г и Е сказались на обучении негативным образом, поскольку в них существовала высокая схожесть ответов между представителями двух классов.

3. Ответы студентов, отстоящие от общей массы ответов респондентов (артефакты), сказались на обучении негативно, поскольку потенциально могут обучить систему воспринимать подобные нестандартные ответы как определяющие для того или иного класса.

3.3.2 Результаты работы программы

На выходе получилась следующая таблица значений чувствительности, специфичности и эффективности.

	KNN	DT	RF	NB	LOGREG	SV
чувствительность	0.2998	0.6268	0.3185	0.7142	0.5428	0.5664
специфичность	0.7107	0.8737	0.7808	0.7272	0.7763	0.7146
эффективность	0.4269	0.7322	0.4647	0.7207	0.6295	0.6210

Таблица 3.1

Выводы на основании таблицы

1. Сравнив строки «чувствительность» и «специфичность» можно прийти к выводу, что класс «1» всеми алгоритмами предсказывается с большей точностью, чем класс «0».

2. Эффективнее всего сработали алгоритмы деревьев решений и наивного Байеса (DT и NB в таблице). Эффективность первого метода составила 0.7322, а эффективность второго - 0.7207, и остальные показатели выше, чем показатели других алгоритмов.

3. Для классификатора деревьев решений, чувствительность которого на выходе 0.6268, а специфичность 0.8737, наличие в базе ненадежных элементов, большинство из которых относится к классу «0», означает снижение точности определения принадлежности к этому же классу «0», тогда как для классификатора наивного Байеса присутствие этих элементов сказывается в

равной степени как на предсказании класса «0», так и на предсказании класса «1», поскольку его чувствительность 0.7142, а специфичность 0.7272.

4. Метод ближайших соседей (KNN в таблице) и метод случайного леса (RF в таблице) хуже, чем другие методы, определяют принадлежность к классу «0». Можно сделать вывод, что они более чувствительны к ненадежным элементам, в частности, к их классу.

Обоснованием подобной точности алгоритмов и общих проблем с отнесением объектов к классу «0» является недостаточное количество элементов этой группы, а также содержание среди них ошибочных. На этапе 3.1.2 «Предобработка данных» была проведена работа по исключению несоответствующих требованиям объектов, однако в последствии они были добавлены обратно, и алгоритмы обучались на целостной базе, потому процент ошибочных объектов класса «0» слишком высок и дает слишком высокую погрешность, поскольку классификаторы обучались на частично некорректной базе.

Обучим алгоритмы на базе, из которой исключены ненадежные элементы (выбросы). Результат отражен в таблице 3.2.

	KNN	DT	RF	NB	LOGREG	SV
чувствительность	0.2646	0.4723	0.3262	0.5176	0.5314	0.4003
специфичность	0.7984	0.6298	0.7798	0.6761	0.7672	0.7780
эффективность	0.4136	0.5230	0.4739	0.5770	0.6257	0.4977

Таблица 3.2

Выводы на основании второй таблицы

1. Рассмотрев детально каждую строку, можно прийти к выводу, что заметные изменения произошли с чувствительностью, для метода деревьев решений произошло ее снижение с 0.6268 до 0.4723, для Наивного Байесовского классификатора с 0.7142 до 0.5176, что говорит о повысившейся сложности определения принадлежности к классу «0».

2. Произошло недообучение в силу недостаточного наличия элементов класса «0» в базе.

3. Часть алгоритмов (случайный лес, логистическая регрессия, метод опорных векторов) выдала схожие с первыми результаты.

4. Наивный Байесовский классификатор и метод деревьев решений на выходе оказались более чувствительными к качеству входной группы данных.

Общие выводы

1. Все методы хорошо справляются с классификацией объектов класса «1».

2. Объекты класса «0» характеризуются хуже.

3. Достигнутая после двух запусков программы эффективность показывает, что алгоритм деревьев решений и алгоритм наивного Байеса способны осуществлять поиск закономерностей даже среди неидеальной базы данных, что выделяет их на фоне других и позволяет получить лучшие результаты.

4. Для получения близких к идеальным результатов необходимо, чтобы база данных содержала развернутую информацию о представителях каждого класса, а также большое количество объектов, относящихся к каждому из них.

Заключение

В ходе работы были решены следующие задачи:

1. Были изучены основные положения машинного обучения.
2. Было разобрано несколько классификаторов машинного обучения, их пути развития, определены наиболее результативные для приведенной базы.
3. Был проведен сравнительный анализ результатов работы выбранных классификаторов.
4. На данных из базы классификаторы были обучены определять принадлежность элементов к определенным классам, проведена оценка эффективности для каждого из алгоритмов. Для метода деревьев решений была достигнута эффективность 73%, для алгоритма наивного байеса 72%.

Все цели, поставленные в данной научной работе, выполнены. Общую результативность работы можно оценить на основании следующих компетенций.

В процессе проведения работы с классификаторами был изучен обширный спектр теории машинного обучения, был осуществлен поиск информации в большом количестве научных статей, после чего полученные сведения были отражены в программном коде, в частности, при выборе параметров классификаторов и были неоднократно оптимизированы для достижения лучших показателей, что соответствует компетенции ОПК-1. «Способен применять естественнонаучные и общетехнические знания, методы математического анализа и моделирования, теоретического и экспериментального исследования в профессиональной деятельности»

Для работы с методами классификации, их интегрирования, настройки, а также проведения их сравнительного анализа на ПК под управлением Windows 10 был установлен язык программирования Python 3.9.7, его компилятор, а также в редактор кода Visual Studio Code было установлено расширение Jupiter Notebook, что соответствует компетенции ОПК-5. «Способен устанавливать программное и аппаратное обеспечение для информационных

и автоматизированных систем» и компетенции ОПК-2. «Способен понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности».

В процессе работы был произведен поиск информации в литературных базах «Киберленинка», «IEEEExplore», а также в электронных библиотеках «Университетская библиотека ONLINE» и «Лань», что соответствует компетенции ОПК-3. «Способен решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности».

Работа выполнена в строгом соответствии с установленными кафедрой ИСКМ нормами, касающимися хода работы, оформления, а также необходимого перечня документов, что соответствует компетенции ОПК-4. «Способен участвовать в разработке стандартов, норм и правил, а также технической документации, связанной с профессиональной деятельностью».

Программа, полученная на выходе, способна предсказывать направление подготовки абитуриента на основании его ответов на фиксированный перечень вопросов, для чего необходимо просто загрузить объект в базу данных, что говорит о соответствии работы компетенции ОПК-6. «Способен разрабатывать алгоритмы и программы, пригодные для практического использования, применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов».

На основании фактических сведений о работоспособности алгоритмов классификации при различной их настройке были многократно изменены гиперпараметры, выбираемые при предсказании, из чего можно сделать вывод о соответствии компетенции ОПК-7. «Способен применять в практической деятельности основные концепции, принципы, теории и факты, связанные с информатикой».

Основным компонентом и хранилищем объектов, на которых обучались классификаторы, являлась база данных абитуриентов 2021 года поступления, информация из которой была загружена и обработана в программе, что свидетельствует о соответствии компетенции ОПК-8. «Способен осуществлять поиск, хранение, обработку и анализ информации из различных источников и баз данных, представлять ее в требуемом формате с использованием информационных, компьютерных и сетевых технологий».

Список литературы

1. Почему Python — лучший язык для машинного обучения и ИИ / «pythonist.ru». — URL: <https://pythonist.ru/pochemu-python-luchshij-yazyk-dlya-mashinnogo-obucheniya-i-ii/> (дата обр. 12 апр. 2022). — Режим доступа: свободный. — Текст: электронный.
2. *Geron A.* Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. — 2-е изд. — «O'Reilly Media», 2019. — С. 10—21.
3. *Клячкин В. Н., Жуков Д. А.* Прогнозирование состояния технического объекта с применением методов машинного обучения. // Прикладные продукты и системы. — 2019. — С. 1—2.
4. *Berkhin P.* Survey of Clustering Data Mining Techniques // Accrue Software. — 2006. — С. 1—56.
5. *Каниа К. А.* Нейронные сети. Эволюция. — «ЛитРес», 2020. — С. 1—7.
6. *Вьюгин В. В.* Математические основы теории машинного обучения и прогнозирования. — «МЦМНО», 2013. — С. 10—14.
7. *Rashid T.* Make Your Own Neural Network. — CreateSpace Independent Publishing Platform, 2016. — С. 101—103.
8. *Hackeling G.* Mastering Machine Learning with scikit-learn. — 2-е изд. — «РАСКТ», 2014. — С. 50—56.
9. Evaluating Categorical Models II: Sensitivity and Specificity. / «Towards Data Science.com». — URL: <https://towardsdatascience.com/evaluating-categorical-models-ii-sensitivity-and-specificity-e181e573cff8> (дата обр. 29 апр. 2022). — Режим доступа: свободный. — Текст: электронный.

10. *Шашкин А. И., Стрюков Р. К.* О модификации метода ближайших соседей // Вестник Воронежского государственного университета. — 2015. — № 1. — С. 1—7.
11. *Кувайскова Ю. Э.* Статистические методы прогнозирования: Учебное пособие (СЭБ). — УлГТУ, 2019. — С. 10—15.
12. *Сорокин А. Б., Железняк Л. М.* Технологии обучения кластеризация и классификация: Практикум (СЭБ). // ФГБОУ ВО «МИРЭА». — 2021. — С. 5—10.
13. *Гофман Е. А., Олейник А. А., Субботин С. А.* Кластер-анализ с использованием деревьев решений. // Компьютерные науки. — 2011. — Т. 1. — С. 1—7.
14. *Макшанов А. В., Журавлев А. Е., Тындыкаръ Л. Н.* Большие данные Big Data: учебник для вузов. — «Лань», 2022. — С. 120—188.
15. Объяснение и настройка параметров алгоритма случайного леса / «russianblogs.com». — URL: <https://russianblogs.com/article/15251109270/> (дата обр. 20 мая 2022). — Режим доступа: свободный. — Текст: электронный.
16. *Лесковец Ю., Раджараман А., Ульма Д. Д.* Анализ больших наборов данных. — «ДМК Пресс», 2016. — С. 225—240.

Приложение А

Листинг разработанной программы

Листинг А.1. Пример листинга в приложении

```
1 import math
2 import pandas as pd
3 import numpy as np
4 import seaborn as sns
5 df = pd.read_excel('preliminary.xlsm')
6 print(df)
7 n = 5000
8 characteristic_table = [['KNN'], [0],[0],[0]],
9                          [['DT'], [0],[0],[0]],
10                         [['NB'], [0],[0],[0]],
11                         [['RF'], [0],[0],[0]],
12                         [['LR'], [0],[0],[0]],
13                         [['SVC'], [0],[0],[0]]
14 method_id = 0
15 every_question_base = []
16 j = 0
17 for i in range (4, 34):
18     every_question_base.append(np.array(df.iloc[:, i]))
19     print(every_question_base[j])
20     print('\n')
21     j+1
22
23 from sklearn.metrics import confusion_matrix
24 import matplotlib.pyplot as plt
25
26 def confusion_matrix_visualization(cm):
27     matrix = sns.heatmap(cm, annot=True, cmap='Blues')
28
29     matrix.set_title('General number of each criteria\n\n')
```

```

30     matrix.set_xlabel('\nPredicted Values')
31     matrix.set_ylabel('Actual Values ')
32
33     matrix.xaxis.set_ticklabels(['physics','maths'])
34     matrix.yaxis.set_ticklabels(['physics','maths'])
35
36     plt.show(matrix)
37
38 def acc(cm):
39     total=sum(sum(cm))
40     accuracy=(cm[0,0]+cm[1,1])/total
41     return accuracy
42
43 def sens(cm):
44     sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
45     return sensitivity
46
47 def spec(cm):
48     specificity = cm[1,1]/(cm[1,0]+cm[1,1])
49     return specificity
50
51 def effect(cm):
52     effectivity = math.sqrt(sens(cm)*spec(cm))
53     return effectivity
54
55
56 def count_values(true, predicted):
57     cm = confusion_matrix(true, predicted)
58     print("ACCURACY: ", acc(cm))
59     values = [0, 0, 0]
60     values[0] = sens(cm)
61     values[1] = spec(cm)
62     values[2] = effect(cm)
63     return (values)
64

```

```

65 x = df.values[:, 4:32]
66 x=x.astype('int')
67 y = df.values[:, 34]
68 y=y.astype('int')
69 print(x)
70 print(y)
71 from sklearn.model_selection import train_test_split
72
73 from sklearn.neighbors import KNeighborsClassifier
74 method_id = 0
75 values = [0, 0, 0]
76 sensitivity = []
77 specifity = []
78 effectivity = []
79 knn = KNeighborsClassifier(n_neighbors=9, weights="uniform",
                             metric="euclidean")
80 for i in range(0, n):
81     x_train, x_test, y_train, y_test = train_test_split(x, y,
                                                            test_size=0.20, stratify=y)
82     knn.fit(x_train, y_train)
83     predicted = knn.predict(x_test)
84     values = count_values(y_test, predicted)
85     print(values)
86     sensitivity.append(values[0]), specifity.append(values[1]),
        effectivity.append(values[2])
87 print(np.mean(sensitivity), np.mean(specifity), np.mean(
        effectivity))
88 characteristic_table[method_id][1] = np.mean(sensitivity)
89 characteristic_table[method_id][2] = np.mean(specifity)
90 characteristic_table[method_id][3] = np.mean(effectivity)
91
92 from sklearn.tree import DecisionTreeClassifier
93 dtc = DecisionTreeClassifier(criterion = "entropy", max_depth =
        5)
94 method_id = 1

```

```

95 values = [0, 0, 0]
96 sensitivity = []
97 specificity = []
98 effectivity = []
99 for i in range(0, n):
100     x_train, x_test, y_train, y_test = train_test_split(x, y,
101                                                         test_size=0.20, stratify=y)
101     dtc.fit(x_train, y_train)
102     predicted = knn.predict(x_test)
103     values = count_values(y_test, predicted)
104     print(values)
105     sensitivity.append(values[0]), specificity.append(values[1]),
106         effectivity.append(values[2])
106 print(np.mean(sensitivity), np.mean(specificity), np.mean(
107     effectivity))
107 characteristic_table[method_id][1] = np.mean(sensitivity)
108 characteristic_table[method_id][2] = np.mean(specificity)
109 characteristic_table[method_id][3] = np.mean(effectivity)
110 sensitivity, specificity, effectivity = 0, 0, 0
111
112 from sklearn.tree import export_graphviz
113 from six import StringIO
114 from IPython.display import Image
115 import pydotplus
116
117 answer_cols = df.iloc[-1, 4:32]
118
119 dot_data = StringIO()
120 export_graphviz(dtc, out_file=dot_data, filled=True, rounded =
121     True, special_characters=True, feature_names=answer_cols,
122     class_names = ['0', '1'])
121 graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
122 graph.write_png('Students.png')
123 Image(graph.create_png())
124

```



```

125 from sklearn.naive_bayes import GaussianNB
126 method_id = 2
127 nb = GaussianNB()
128 sensitivity = []
129 specifity = []
130 effectivity = []
131 for i in range(0, n):
132     x_train, x_test, y_train, y_test = train_test_split(x, y,
133                                                         test_size=0.20, stratify=y)
134     nb.fit(x_train, y_train)
135     predicted = knn.predict(x_test)
136     values = count_values(y_test, predicted)
137     print(values)
138     sensitivity.append(values[0]), specifity.append(values[1]),
139         effectivity.append(values[2])
140 print(np.mean(sensitivity), np.mean(specifity), np.mean(
141     effectivity))
142 characteristic_table[method_id][1] = np.mean(sensitivity)
143 characteristic_table[method_id][2] = np.mean(specifity)
144 characteristic_table[method_id][3] = np.mean(effectivity)
145 sensitivity, specifity, effectivity = 0, 0, 0
146
147 from sklearn.ensemble import RandomForestClassifier
148 method_id = 3
149 rf = RandomForestClassifier(n_estimators=100,
150                             bootstrap = True,
151                             max_features = 'sqrt')
152 sensitivity = []
153 specifity = []
154 effectivity = []
155 for i in range(0, n):
156     x_train, x_test, y_train, y_test = train_test_split(x, y,
157                                                         test_size=0.20, stratify=y)
158     rf.fit(x_train, y_train)
159     predicted = rf.predict(x_test)

```

```

156     values = count_values(y_test, predicted)
157     print(values)
158     sensitivity.append(values[0]), specificity.append(values[1]),
        effectivity.append(values[2])
159 print(np.mean(sensitivity), np.mean(specificity), np.mean(
    effectivity))
160 characteristic_table[method_id][1] = np.mean(sensitivity)
161 characteristic_table[method_id][2] = np.mean(specificity)
162 characteristic_table[method_id][3] = np.mean(effectivity)
163 sensitivity, specificity, effectivity = 0, 0, 0
164
165 from sklearn.linear_model import LogisticRegression
166 method_id = 4
167 logreg = LogisticRegression()
168 sensitivity = []
169 specificity = []
170 effectivity = []
171 for i in range(0, 50):
172     x_train, x_test, y_train, y_test = train_test_split(x, y,
        test_size=0.20, stratify=y)
173     logreg.fit(x_train, y_train)
174     predicted = logreg.predict(x_test)
175     values = count_values(y_test, predicted)
176     print(values)
177     sensitivity.append(values[0]), specificity.append(values[1]),
        effectivity.append(values[2])
178 print(np.mean(sensitivity), np.mean(specificity), np.mean(
    effectivity))
179 characteristic_table[method_id][1] = np.mean(sensitivity)
180 characteristic_table[method_id][2] = np.mean(specificity)
181 characteristic_table[method_id][3] = np.mean(effectivity)
182 sensitivity, specificity, effectivity = 0, 0, 0
183
184 from sklearn import svm
185 method_id = 5

```

```

186 svc = svm.SVC(kernel='linear')
187 sensitivity = []
188 specifity = []
189 effectivity = []
190 for i in range(0, n):
191     x_train, x_test, y_train, y_test = train_test_split(x, y,
192                                                         test_size=0.20, stratify=y)
193     svc.fit(x_train, y_train)
194     predicted = svc.predict(x_test)
195     values = count_values(y_test, predicted)
196     print(values)
197     sensitivity.append(values[0]), specifity.append(values[1]),
198     effectivity.append(values[2])
199 print(np.mean(sensitivity), np.mean(specifity), np.mean(
200     effectivity))
201 characteristic_table[method_id][1] = np.mean(sensitivity)
202 characteristic_table[method_id][2] = np.mean(specifity)
203 characteristic_table[method_id][3] = np.mean(effectivity)
204 sensitivity, specifity, effectivity = 0, 0, 0
205
206 print(characteristic_table)
207 for j in range(1, 5):
208     for i in range(1, 4):
209         if(j == 1):
210             print('KNN: ', characteristic_table[j][i])
211         if(j == 2):
212             print('DT: ', characteristic_table[j][i])
213         if(j == 3):
214             print('NB: ', characteristic_table[j][i])
215         if(j == 4):
216             print('RF: ', characteristic_table[j][i])
217         if(j == 5):
218             print('RF: ', characteristic_table[j][i])
219     print(' ')
220

```

```

218 fig, ax = plt.subplots()
219 ax.xaxis.set_visible(False)
220 ax.yaxis.set_visible(False)
221 clust_data = characteristic_table
222 collabel=("method", "sensitivity", "specitivity", "effectivity")
223 ax.table(cellText=clust_data,colLabels=collabel,loc='center')
224
225 for i in range (0 , 30):
226     ax = sns.boxplot(x=df.iloc[:, 34], y=every_question_base[i])
227     plt.show()

```