МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт Математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль)/специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка приложения для сравнения эффективности алгоритмов классификации данных»

Обучающийся И. Балтаев
(Инициалы Фамилия) (личная подпись)

Руководитель к.т.н., В.С. Климов
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант Е.В. Косс
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2022

Аннотация

Тема бакалаврской работы – «Разработка приложения для сравнения эффективности алгоритмов классификации данных».

Развитие технологий машинного обучения привело возникновению большого количества алгоритмов классификации данных. Эти алгоритмы анализируют исходных данных и определяют содержащиеся в них закономерности. На основе найденных закономерностей строится классификатор данных, который позволяет решать прогнозирования и распознавания объектов.

Так как заранее не известно какой алгоритм классификации лучше всего подходит под имеющиеся данные, актуальным вопросом является разработка программного обеспечения для сравнения эффективности алгоритмов классификации.

Цель работы - разработка приложения для сравнения эффективности алгоритмов классификации данных.

Объектом исследования является классификация данных.

Предметом работы является автоматизация сравнения алгоритмов классификации данных.

В данной работе используются такие методы исследования, как программное моделирования алгоритмов машинного обучения и статистический анализ данных.

Практическая значимость работы заключается в разработке исследовательского программного обеспечения.

Данная работа включает в себя введение, три главы, заключение и список используемой литературы.

Бакалаврская работа состоит из страниц 47 страниц текста, 23 рисунков, 1 таблицы и 20 источников.

Abstract

The title of the graduation work is "Development of an application for comparing the effectiveness of data classification algorithms".

The development of machine learning technologies has led to the emergence of a large number of data classification algorithms. These algorithms analyze the source data and determine the patterns contained in them. Based on the found patterns, a data classifier is built, which allows you to solve forecasting and object recognition.

It is unknown which classification algorithm best fits the available data. Therefore, an urgent issue is the development of software for comparing the effectiveness of classification algorithms.

The aim of the work is to develop an application for comparing the effectiveness of data classification algorithms.

The object of the study is the classification of data.

The subject of the work is automation of comparison of algorithms for data classification.

In this work, such research methods as software modeling of machine learning algorithms and statistical data analysis are used.

The practical significance of the work lies in the development of research software.

This work includes an introduction, three parts, a conclusion and a list of references.

The graduation work consists of 47 pages of text, 23 figures, 1 table and 20 sources.

Оглавление

Введение

Рост популярности алгоритмов машинного обучения привело к развитию многих научных отраслей, таких как машинное зрение, экспертные системы, интеллектуальный анализ данных, обработка естественного языка [1, 2]. Популярность машинного обучения обусловлена возможностью этих алгоритмов самостоятельно, без участия человека, анализировать данные и описывать закономерности в них в виде моделей. Так, например, алгоритмы машинного обучения позволяют строить модели классификации и регрессии на основе имеющихся данных [3, 4].

Модели классификации используется при решении таких задач, как прогнозирование, диагностика, распознавание образов и идентификация [8, 9, 10]. Примером задачи классификации является построение, на основе исторических данных банка, модели, определяющей класс платежеспособности клиента по его имеющимся данным. Класс платежеспособности клиента используется для определения рекламируемых услуг в приложении банка и при формировании персональных условий кредитования.

Развитие области машинного обучения приводит к появлению все новых и новых алгоритмов классификации [11]. Каждый алгоритм основан на своем подходе к решению задачи классификации и имеет свой математический аппарат. Разнообразие алгоритмов машинного обучения связано с тем, что пока не существует такого одного универсального алгоритма, который бы подходил для любых исходных данных [5, 6, 7].

Некоторые алгоритмы не умеют работать с пропущенными значениями в данных, другие алгоритмы требовательны к количеству анализируемых данных, также существуют алгоритмы, которые не допускают наличие в данных неинформативных признаков, потому что не умеют их отсеивать и т.д.

Таким образом, можно выделить следующие проблемы в области интеллектуального анализа данных:

– большое разнообразие технологий машинного обучения приводит к необходимости подбора наиболее подходящего алгоритма под имеющиеся данные;

– никогда заранее не известно, какой из алгоритмов машинного обучения позволит построить наиболее точную модель классификации на основе имеющихся данных.

Для решения этих проблем в работе предложено разработать приложения для сравнения эффективности алгоритмов классификации данных.

Объектом исследования является классификация данных.

Предметом работы является автоматизация сравнения алгоритмов классификации данных.

Цель выпускной квалификационной работы – разработка приложения для сравнения эффективности алгоритмов классификации данных.

Поставленная цель в бакалаврской работе решается путем выполнения следующих задач:

– проведения сравнительного анализ алгоритмов классификации данных;

– определение показателей для сравнения эффективности алгоритмов классификации данных;

– разработка и тестирование приложения для сравнения эффективности алгоритмов классификации.

В данной работе используются такие методы исследования, как программное моделирования алгоритмов машинного обучения и статистический анализ данных.

Практическая значимость работы заключается в разработке исследовательского программного обеспечения. Представленное

программное обеспечение использоваться разработчиками интеллектуальных система для предварительного анализа эффективности алгоритмов на имеющихся данных.

Данная работа состоит из введения, трех глав, заключения, списка используемой литературы.

В первой главе освящаются вопросы практического применения алгоритмов машинного обучения и дается сравнительная характеристика алгоритмов классификации данных.

Вторая глава посвящена разработке методике для сравнения эффективности различных алгоритмов классификации данных.

В третьей главе разработанное программное обеспечение для сравнения эффективности алгоритмов классификации данных.

В заключении представлены результаты, полученные при написании бакалаврской работы.

Бакалаврская работа состоит из страниц 47 страниц текста, 23 рисунков, 1 таблицы и 20 источников.

Глава 1 Анализ текущего состояния развития машинного обучения

1.1 Области применения алгоритмов машинного обучения

Алгоритмы машинного обучения, выполняющие классификацию данных, применяются в следующих областях:

- диагностика заболеваний в медицинской сфере [12, 15];
- распознавание объектов в компьютерном зрении;
- диагностика качества изделий, получаемых на производстве;
- прогнозирование результатов обучения в образовательной сфере [13];
- разметка данных при анализе текстовой и графической информации [14].

Наиболее известными алгоритмами классификации данных являются: K-nearest neighbors (k-ближайших соседей), Decision tree (деревья принятия решений), Support-vector machine (метод опорных векторов), Multilayer perceptron - neural network (многослойный персептрон) и Random forest (случайный лес).

Рассмотрим особенности работы этих для классификации данных.

Принцип работы алгоритм K-nearest neighbors основан на том, что объекты, относящиеся к одному классу, имеют близкие значения числовых признаков. Поэтому в пространстве признаков, объекты из одного класса будет находится рядом с друг другом. Исходя из этого, для классификации любого произвольного объекта требуется определить, объекты какого класса его окружают. У алгоритма есть управляющий параметр k, который отвечает за количество ближайших объектов, на основе которых будет обрядиться метка класса исследуемого объекта.

Пример работы алгоритма K-nearest neighbors для объектов с двумя числовыми признаками показан на рисунке 1. Черным цветов обозначен

исследуемый объект, а голубым и красным цветом – объекты из первого и второго класса. При k=3 объектов из второго класса больше, чем из первого, поэтому объект, отмеченный черным цветом, будет отнесен ко второму классу.

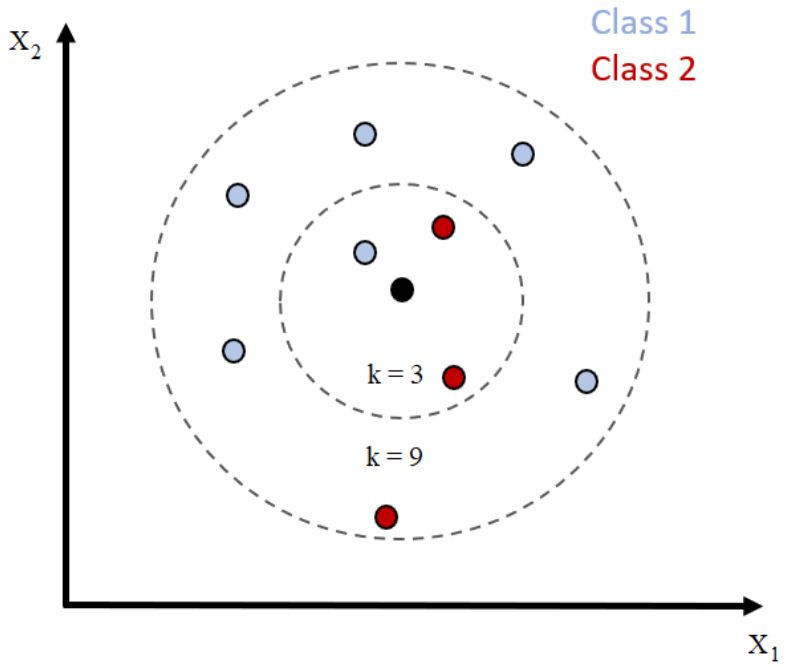


Рисунок 1 – Пример классификации объекта с использованием алгоритма K-nearest neighbors при разных значениях параметра k

Алгоритм Decision tree основан на графовом представлении процесса классификации данных [16, 17]. В этом случае, классификатором является дерево, в узлах которого содержатся условия перехода объектов на нижележащие уровни дерева. При классификации, объект, помещаемый в корень дерева будет спускать по дереву от узла к узлу. В какой узел будет осуществлён следующий переход зависит от параметров анализируемого объекта. В результате спуска объект попадет в один из конечных узлов дерева, содержащего номер класса. И номер этого класса будет считаться результатом классификации.

Пример дерева принятия решений, построенного на выборке данных «Ирисы Фишера» показан на рисунке 2.

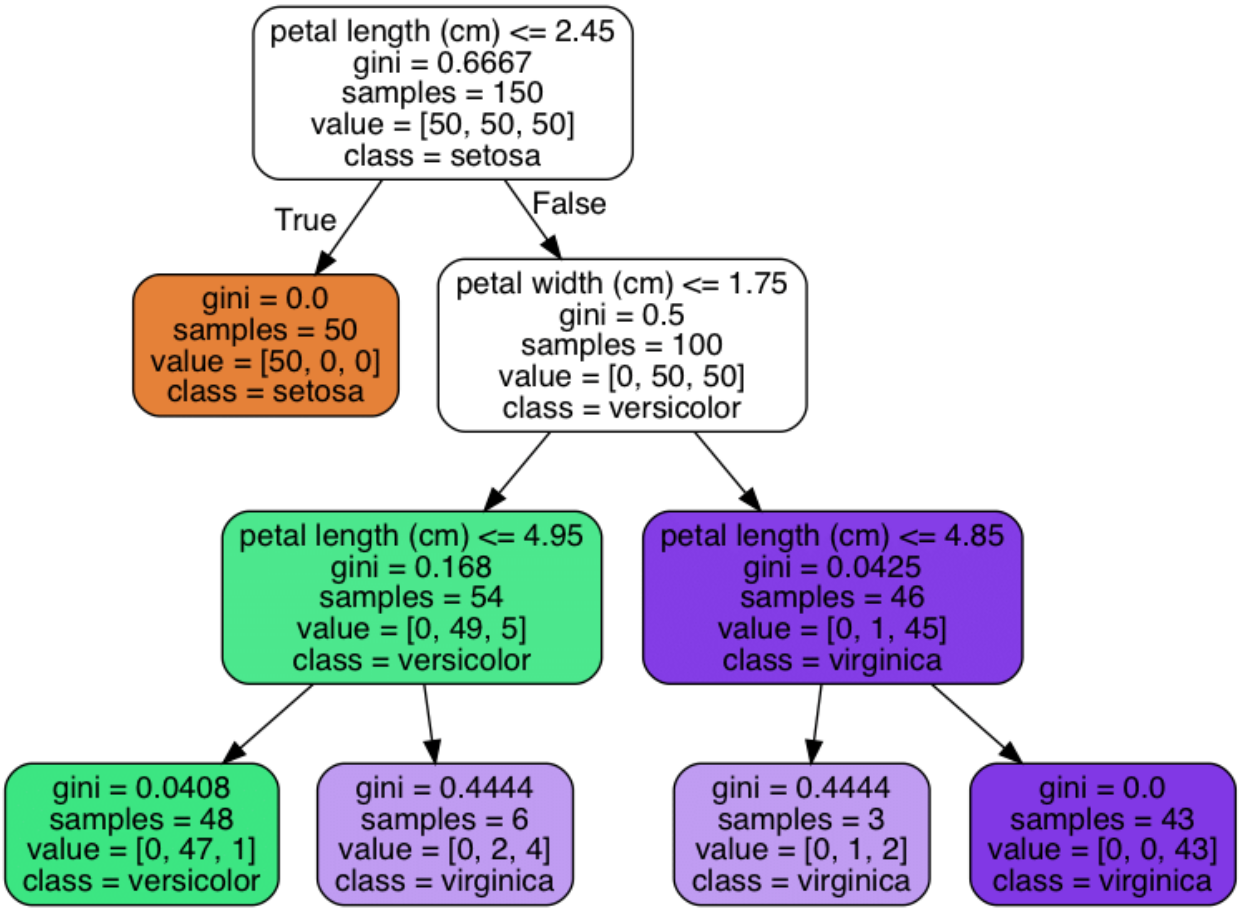


Рисунок 2 – Дерево принятия решений, построенное на наборе данных «Ирисы Фишера»

Принцип работы алгоритма классификации Support-vector machine основан на идее построения гиперплоскости, разделяющей объекты разных классов друг от друга. Параметры плоскости определяются в процессе обучения классификатора.

На рисунке 3 показан один из примеров использования Support-vector machine для разделения объектов двух классов. Синим цветом показаны объекты первого класса, а красным цветом – объекты второго класса. Каждый объект выборки описывается двумя параметрами (X_1, X_2). В этом случае, в качестве плоскости для разделения объектов разных классов

выступает черная линия. Чтобы определить класс исследуемого объекта надо понять с какой стороны от разделяющей линии находится объект.

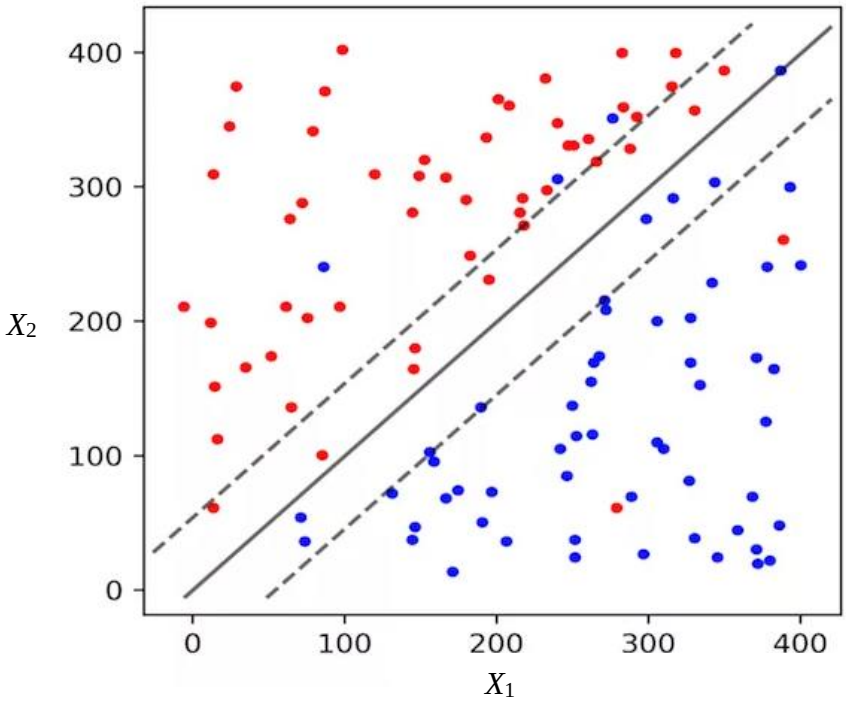


Рисунок 3 – Пример использования Support-vector machine для разделения объектов при бинарной классификации

Классификация объектов с помощью нейронной сети работает следующим образом [18]. Значения параметров x_1, x_2, …, x_n анализируемого объекта подаются на вход обученной нейронной сети. Затем эти сигналы двигаются по направлению к выходу нейронной сети. В процессе продвижению к выходному слою сигналы масштабируются с помощью весовых коэффициентов связей, суммируются на входе каждого нейрона и преобразуются с помощью функций активации. На выходе нейронной сети генерируется вектор выходных значений y_1, y_2, …, y_n. Каждое выходное значение сети отвечает за вероятность принадлежности анализируемого объекта к тому или иному классу [20]. Пример структуры многослойной неронной сети показан на рисунке 4.

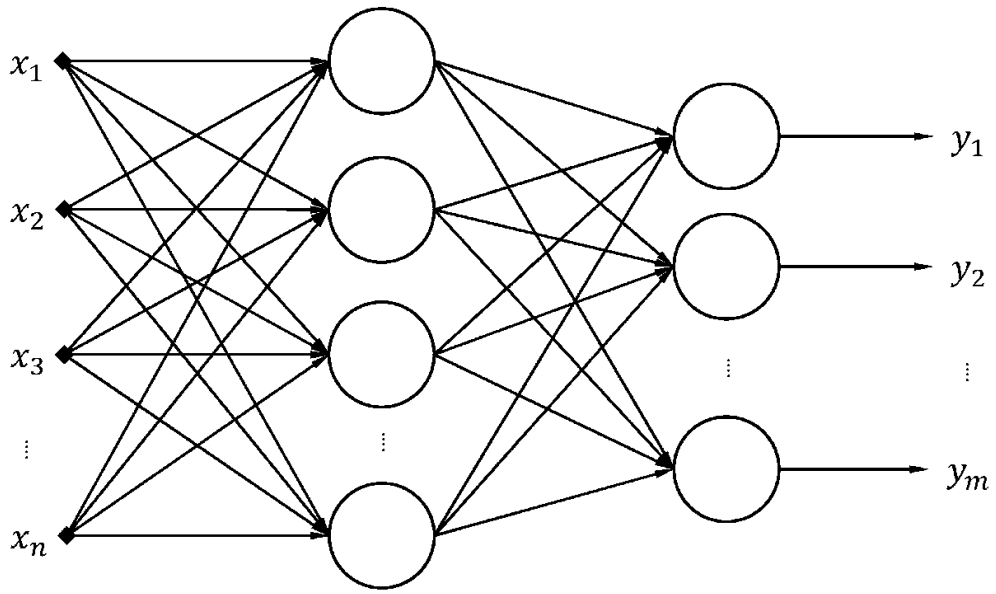


Рисунок 4 – Структура многослойный персептрона (нейронная сеть)

Если объединить несколько деревьев принятия ($tree_1$, $tree_2$, …, $tree_B$) решения в один классификатор то получится структура Random forest [19]. В этом случае, при классификации объекта каждое отдельное дерево голосует за свой. А конечный класс *k* определяется как класс набравший больше всех голосов. Схема показана на рисунке 5.

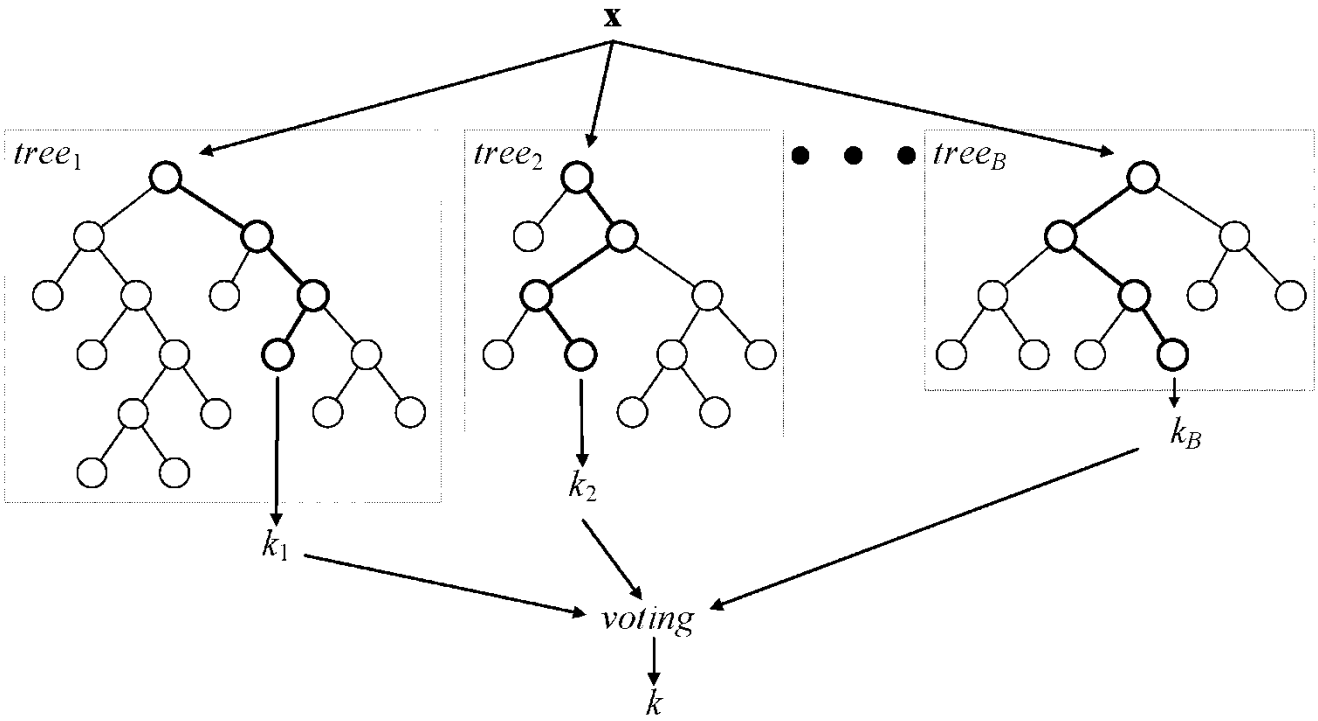


Рисунок 5 – Схема работы Random forest

Таблица 1 – Сравнение алгоритмов машинного обучения

Название	**Особенности алгоритма**	**Чувствительность к размеру обучающей выборки (низкая/средняя/ высокая)**	**Чувствительность к аномальным значениям в данных (низкая/высокая)**	**Возможность обработки пропусков в данных**	**Чувствительность к наличию неинформативных признаков в данных (высокая /низкая)**
K-nearest neighbors (k-ближайших соседей)	Метрический алгоритм	низкая	низкая	отсутствует	высокая
Decision tree (деревья принятия решений)	Иерархический алгоритм	средняя	низкая	присутствует	низкая
Support-vector machine (метод опорных векторов)	Алгоритм линейной классификации	высокая	высокая	отсутствует	высокая
Multilayer perceptron - neural network (многослойный персептрон)	Алгоритм с последовательным преобразованием данных	высокая	высокая	отсутствует	высокая
Random forest (случайный лес)	Ансамблевый алгоритм	средняя	низкая	присутствует	низкая

1.2 Сравнительный анализ алгоритмов классификации данных

На основе изучения различных источников литературы был проведен сравнительный анализ наиболее известных алгоритмов классификации данных. Результаты сравнительного анализа показаны в таблице 1.

Как видно из таблицы 1, у каждого алгоритма классификации есть своя область эффективного использования. Так, например, если обучающая выборка имеет малый размер, то классификаторы Support-vector machine и Multilayer perceptron не смогут обучиться с высокой точностью определять метку класса. И наоборот, классификатор K-nearest neighbors может успешно работать на основе малой выборке данных, так как он не ищет закономерности в данных, а определяет класс анализируемого объекта на основе сравнения с объектами обучающей выборки.

Отдельно стоит отметить, что не существует указаний на то, как определить достаточен ли размер обучающей выборки для применяемого алгоритма или нет. Требуемый размер выборки во многом завит от сложности закономерностей в анализируемых данных. Например, если нейронную сеть требуется обучить выполнять простейшую операцию конъюнкции, то достаточной будет обучающая выборка состоящей из 4 примеров (по одному примеру для всех возможных сочетаний входных данных).

В следствии ошибок в исходный набор данных могут попадают аномальные или неверные значения признаков. И алгоритмы классификации и имеют разную чувствительность к таким неверным значением признаков. Хуже всего с выбросами в данных справляются Multilayer perceptron и Support-vector, так как эти классификаторы не умеют сглаживать аномальные и ошибочные значения признаков. Однако, даже зная какие значения обучающей выборки являются не верными заранее невозможно сказать, сможет ли конкретный алгоритм классификации сгладить эти ошибки в

процессе свой работы или нет. Это связано с тем, что много факторов влияет на результат: концентрация ошибок в определенном классе, величина отклонения от правильных значений, отношение количества ошибочных значений к количеству верных значений и т.д.

Иногда в исходной выборке данных у объектов могут отсутствовать значения некоторых признаков. Алгоритмы классификации отличаются возможностью обработки пропущенных значений в данных. Такие классификаторы, как Decision tree и Random forest умеют обрабатывать пропуски в данных. А классификаторы K-nearest neighbors, Support-vector machine и Multilayer perceptron не могут обрабатывать данные с пропусками.

В обучающей выборке данных могут содержаться неинформативные признаки. Это такие признаки, от которых не завит целевое значение (метка класса). Алгоритмы классификации по-разному обрабатывают неинформативные признаки. Например, классификаторы Decision tree и Random forest умеют отсеивать неинформативные признаки на этапе обучения. Впоследствии, только значимые признаки принимаются во внимание при классификации объектов. Классификаторы K-nearest neighbors, Support-vector machine, Multilayer perceptron воспринимают все признаки как значимые, поэтому при большом количестве неинформативных признаков точность их работы снижается.

С учетом выше сказанного можно заключить, что заранее никогда не известно какой из алгоритмов классификации окажется наиболее эффективным при анализе имеющегося набора данных. Лучший способ найти оптимальный алгоритм – проверить эффективность всех алгоритмов на имеющемся наборе данных и выбрать, выбрать тот из них, который обеспечивает максимальную точность работы.

Вопросы выбора методологии сравнения алгоритмов, а также метрики точности и способы оценки эффективности представлены во второй главе бакалаврской работы.

Выводы к главе 1

В ходе написания 1 главы были сформулированы следующие выводы:

− анализ литературных источников показал, что развитие машинного обучения привело к возникновению большого числа алгоритмов, способных решать задачу классификации данных. Но так как заранее никогда не известно, какой из алгоритмов окажется наиболее эффективным при анализе имеющегося набора данных, актуальной задачей становится разработка программного обеспечения для сравнения эффективности алгоритмов;

− проведен анализ способов применения алгоритмов машинного обучения для решения практических задач из различных областей, выделены наиболее часто используемые алгоритмы классификации данных и проведен их сравнительный анализ с точки зрения взаимодействия с обучающей выборкой данных.

Глава 2 Разработка алгоритма для сравнения эффективности различных алгоритмов машинного обучения

2.1 Постановка задачи классификации данных

Классификация — один из разделов машинного обучения, посвященный решению следующей задачи. «Имеется множество объектов (ситуаций), разделённых некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется обучающей выборкой. Классовая принадлежность остальных объектов не известна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества. Классифицировать объект — значит, указать номер (или наименование класса), к которому относится данный объект. Классификация объекта — номер или наименование класса, выдаваемый алгоритмом классификации в результате его применения к данному конкретному объекту» [1].

Основные элементы задачи классификации показаны на рисунке 6.

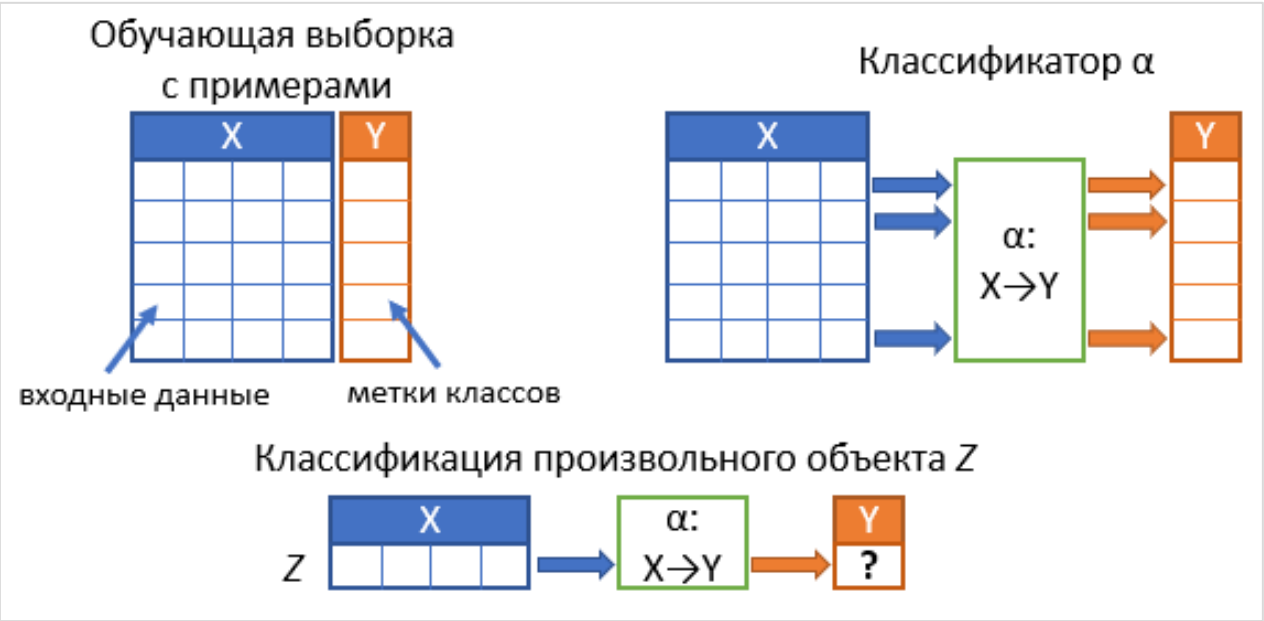


Рисунок 6 – Основные элементы задачи классификации

2.2 Показатели для сравнения эффективности алгоритмов

Одним из известных способов оценки эффективности классификации данных является анализ значений матрицы ошибок. Матрица ошибок составляется на основе наблюдения за работой классификатора на тестовом наборе данных, путем сравнивания выданных классификатором значений и фактическим значений классов. Основные элементы матрица ощибок с расшифровкой обозначений показаны на рисунке 7.

		Предсказанные классификатором значения	
	Суммарный размер выборки (P) + (N)	Positive (PP)	Negative (PN)
Фактические значения в выборке	Positive (P)	True positive (TP), Правильное распознавание класса	False negative (FN), **ошибка 2-ого типа**, пропуск класса Positive
	Negative (N)	False positive (FP), **ошибка 1-ого типа**, ложное срабатывание, распознавание класса Positive для класса отличного от Positive	True negative (TN), правильное распознавание класса отличного от Positive

Рисунок 7 – Схема матрицы ошибок

На основе значений матрицы показатель Accuracy, который оценивает точность работы классификатора по всем классам, рассчитывается по формуле (1):

$$Accuracy = \frac{tp + tn}{tp + fn + fp + tn} \quad (1)$$

Показатель Precision, который оценивает точность работа классификатора по результатам классификации объектов из целевого класса Positive (P), рассчитывает по формуле (2):

$$Precision = \frac{tp}{tp + fp} \quad (2)$$

Показатель Recall, который оценивает долю правильно распознанных объектов на множестве объектов класса Positive рассчитывается по формуле (3):

$$Recall = \frac{tp}{tp + fn} \quad (3)$$

Метрика точности F1, основанная на показателях Precision и Recall рассчитывается по формуле (4):

$$F1 = \frac{2Precision \cdot Recall}{Precision + Recall} \quad (4)$$

На первый взгляд может показаться, что такая схема подсчета показателей на основе матрицы ошибок (рисунок 7) подходит только для задач бинарной классификации, когда первый класс рассматривается как значение Positive (P), а второй класс как значения Negative (N). Однако эта схема может быть использована для задач с любым количеством классов. Для этого проводится анализ каждого класса по отдельности. Сначала берется первый класс и рассматривается как Positive (P), а все остальные классы считаются как Negative (N), при этом рассчитываются показатели по формулам (1-6). Аналогичным образом поступают со вторым классом, третьим классом и т.д. Для каждого показателя точности полученные значения суммируются, а сумма делится на количество l классов.

Таким образом, формулы (1-4) принимают следующий вид. Средняя точность Average Accuracy будет рассчитываться по формуле (5):

$$Average\ Accuracy = \frac{\sum_{i=1}^{l} \frac{tp_i + tn_i}{tp_i + fn_i + fp_i + tn_i}}{l} \quad (5)$$

Доля ошибок Error Rate рассчитывает по формуле (6):

$$Error\ Rate = \frac{\sum_{i=1}^{l} \frac{fp_i + fn_i}{tp_i + fn_i + fp_i + tn_i}}{l} \quad (6)$$

Микропоказатель точности $Precision_{\mu}$ рассчитывается по формуле (7):

$$Precision_{\mu} = \frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l} (tp_i + fp_i)} \quad (7)$$

Микропоказатель точности $Recall_{\mu}$ рассчитывается по формуле (8):

$$Recall_{\mu} = \frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l} (tp_i + fn_i)} \quad (8)$$

Микрометрика $F1_{\mu}$ рассчитывается по формуле (9):

$$F1_{\mu} = \frac{2Precision_{\mu} Recall_{\mu}}{Precision_{\mu} + Recall_{\mu}} \quad (9)$$

Макропоказатель точности $Precision_{M}$ рассчитывается по формуле (10):

$$Precision_{M} = \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fp_i}}{l} \quad (10)$$

Макропоказатель точности $Recall_{M}$ рассчитывается по формуле (11):

$$Recall_{M} = \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fn_i}}{l} \quad (11)$$

Макрометрика $F1_{M}$ рассчитывается по формуле (12):

$$F1_{M} = \frac{2Precision_{M} Recall_{M}}{Precision_{M} + Recall_{M}} \quad (12)$$

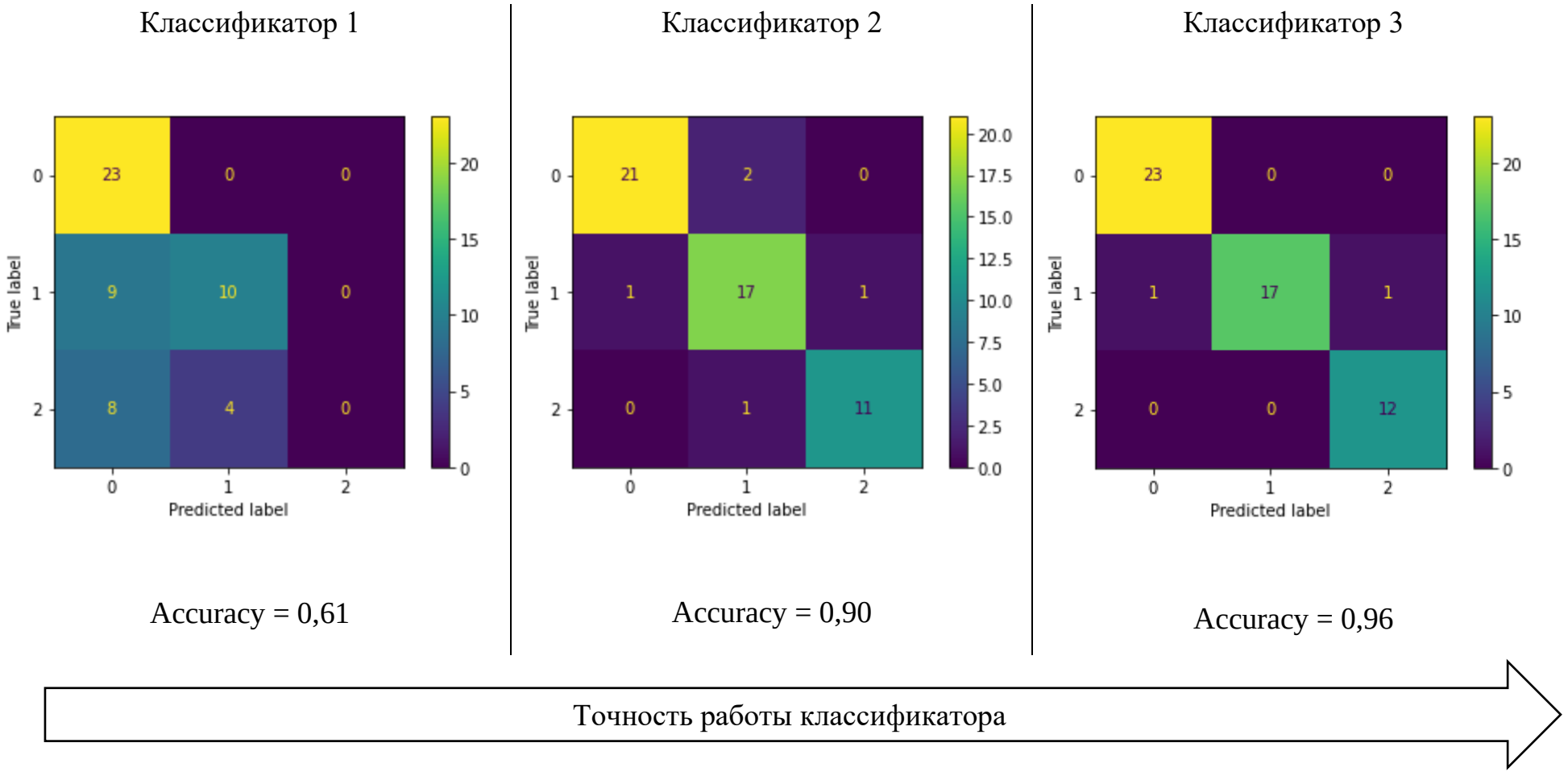


Рисунок 8 – Оценка эффективности классификатора по матрице ошибок

Графическое представление матрицы ошибок является мощным инструментом для оценки эффективности классификации данных. Рассмотрим основы использования матрицы ошибок для сравнения классификаторов в задаче с тремя классами на примере результатов, показанных на рисунке 8. На матрицы ошибок идеального классификатора значения всех компонентов за исключением главной диагонали равны нулю. Ближайший к идеальному – классификатор под номером 3. При работе этого классификатора на тестовой выборке данных, он совершил 2 ошибки, которые связаны с распознаванием объекта класса 1.

В данном примере, худшим является классификатор под номером 1. Он допустил больше всего ошибок при классификации объектов, а судя по правому нижнему значению матрицы равному 0, он не смог распознать правильно ни один объекта из класса 2.

Когда не требуется анализировать результаты работы алгоритмов классификации на тестовых данных, для оценки эффективности можно использовать суммарную точность классификации (показатель Accuracy). Как видно из рисунка 8, максимальное значение Accuracy у третьего классификатора.

На основе рассмотренных показателей точности классификации данных, а также с учетом возможности визуализации результатов работы алгоритмов на тестовой выборке данных с использованием матрицы ошибок была разработана схема оценки эффективности алгоритмов, показанная на рисунке 9.

Данная схема оценки эффективности алгоритмов классификации включает в себя выполнение следующих этапов:

- загрузка исходных данных из файла;
- разделение исходной выборки данных на обучающую и тестовую;

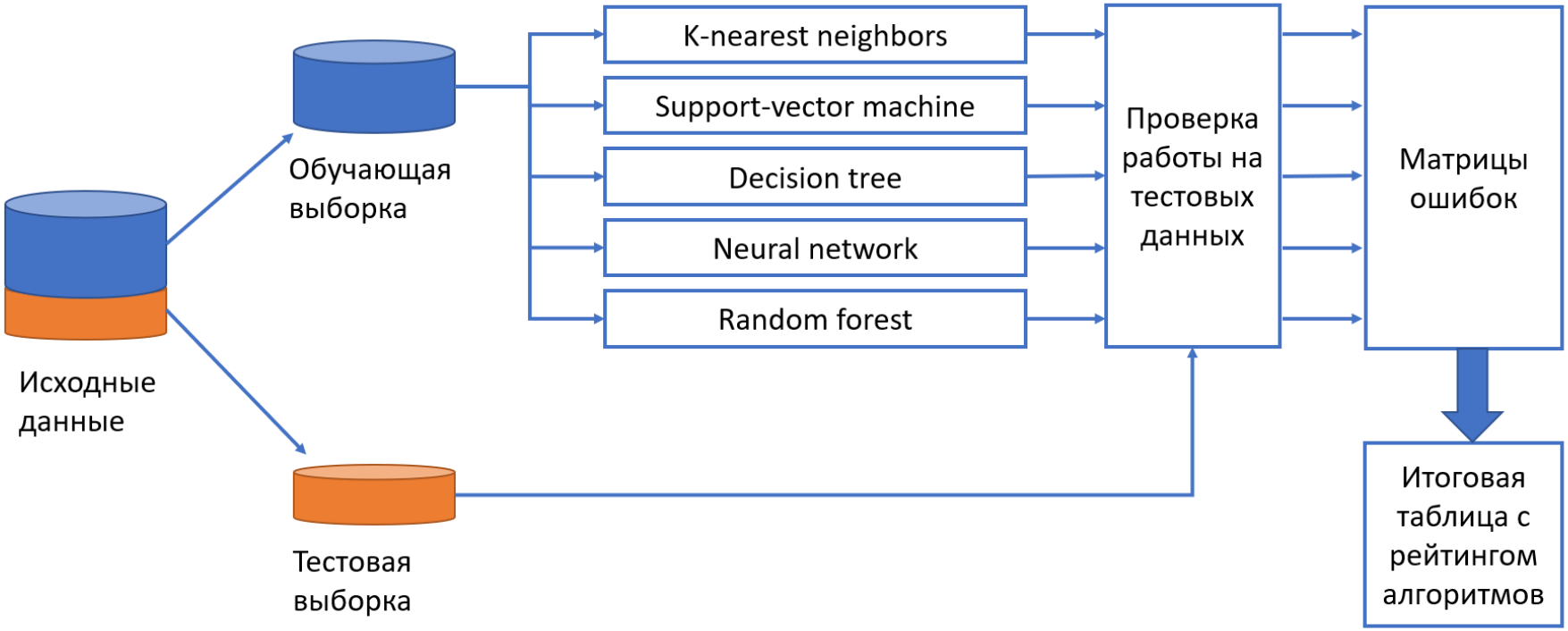


Рисунок 9 – Схема оценки эффективности алгоритмов классификации

− использования обучающей выборки данных для построения с использованием различных алгоритмов (k-nearest neighbors, random forest, support-vector machine, decision tree, neural network) соответствующих классификаторов;

− тестирование работы классификаторов на тестовой выборке данных;

− сбор данных и визуализация результатов классификации в виде набора матриц ошибок;

− формирование рейтинговой таблицы с обобщенными результатами работы алгоритмов классификации данных, отсортированной в порядке снижения эффективности.

Наиболее эффективным алгоритмом является тот, который показывает максимальную точность работы на тестовом наборе данных.

Выводы к главе 2

В ходе написания 2 главы были сформулированы следующие выводы:

− рассмотрена формальная постановка задачи классификации, метрики, используемые для оценки точности работы классификаторов, особенности использования матрицы ошибок для контроля корректности работы классификаторов.

− разработана методика сравнения эффективности работы алгоритмов для классификации данных, которая включает в себя разделение исходных данных на обучающую и тестовую выборки, использование обучающей выборки данных для построения множества классификаторов основанных различных алгоритмах (knn, random forest, svm, decision tree, neural network), проверку точности работы классификаторов на тестовой выборке и визуализацию работы классификаторов с использованием матрицы ошибок.

Глава 3 Приложение для сравнения эффективности алгоритмов классификации данных

3.1 Обзор разработанного приложения

При выполнении бакалаврской работы на языке python было разработано программное обеспечение, реализующее описанные в предыдущей главе подходы по сравнению эффективности алгоритмов классификации данных.

Разработанное программное обеспечение обладает следующим функционалом:

- импорт данных для тестирования алгоритмов классификации из файлов различных форматов (xslx и csv);
- возможность тестирования алгоритмов на встроенных в приложение наборах данных – «Ирисы Фишера», «Классификация вина».
- визуализация исходных данных в виде интерактивной таблицы;
- определение размерности исходных данных – количество объектов и признаков в данных, а также подсчет количества пропусков данных.
- разделение исходных данных на тренировочную и тестовую выборку в заданных пользователем пропорциях;
- поддержка тестирования различных алгоритмов классификации данных: k-nearest neighbors, random forest, Support-vector machine, decision tree, neural network;
- сравнение результатов работы алгоритмов классификации по точности и визуализация результатов сравнения в виде таблицы;
- детализации результатов тестирования алгоритмов классификации с расчетом метрик precision, recall, f1-score, support и матрицы ошибок.

3.2 Графический интерфейс пользователя

Для удобного взаимодействия с пользователем приложение имеет графический интерфейс, который представлен на рисунке 10. Логически интерфейс поделен на 2 части:

– область «Содержание», которая находится в левой части экрана и предназначена для перемещения между блоками, повещённым различным этапам анализа данных и построения классификаторов;

– область с основным содержанием, которая находится с правой части экрана и предназначена для отображения основных элементов управления приложением.

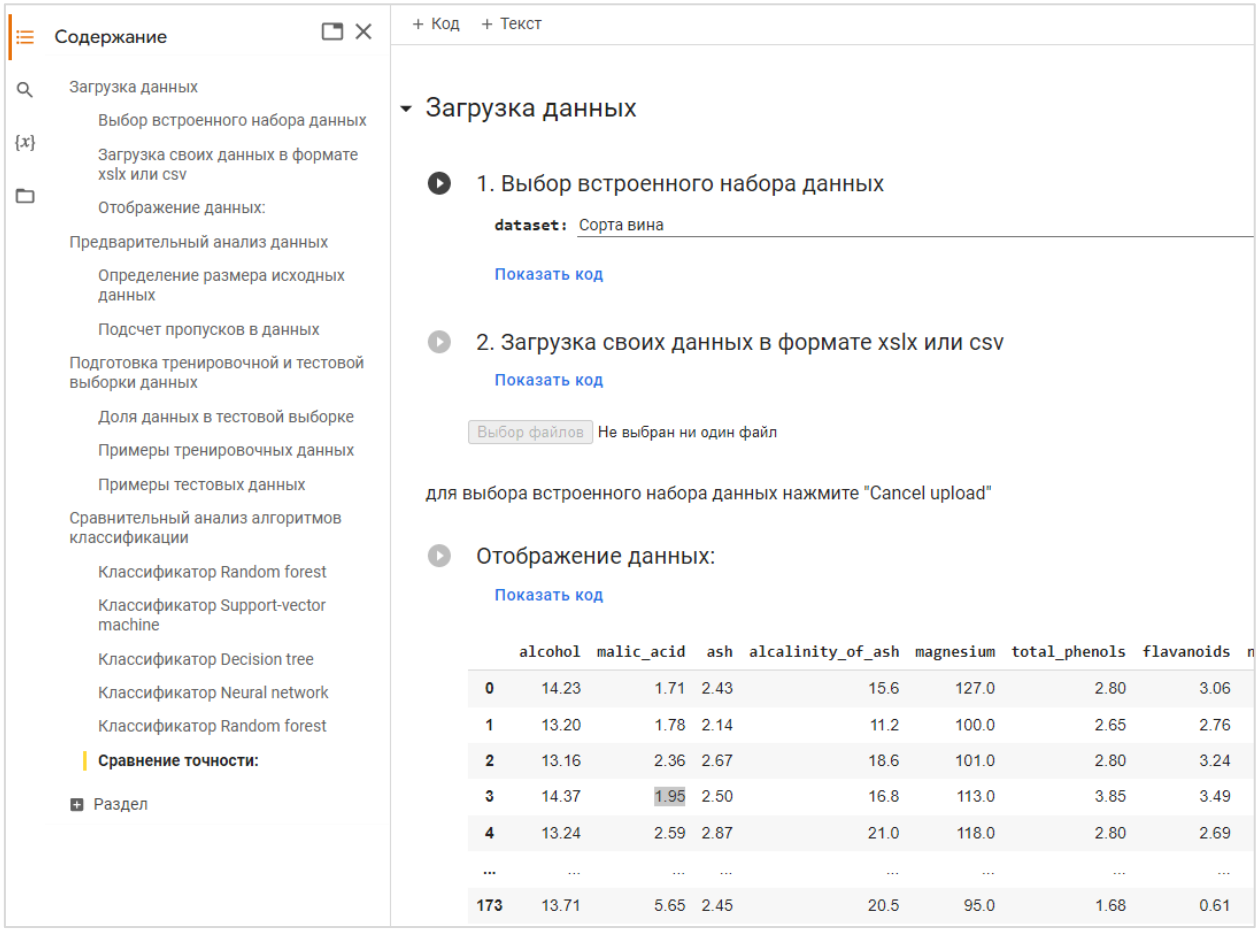


Рисунок 10 – Интерфейс приложения

Приложение для тестирования эффективности алгоритмов классификации включает в себя следующие разделы с элементами управления:

- загрузка данных;
- предварительный анализ данных;
- подготовка тренировочной и тестовой выборок данных;
- сравнительный анализ алгоритмов классификации данных.

В разделе «Загрузка данных» расположены элементы управления для выбора данных, на которых будут тестироваться алгоритмы классификации. Пользователю предоставляется выбор: он может воспользоваться одним из встроенных в приложение наборов данных, выбрав его в списке «dataset» или загрузить свои данные, нажав на кнопку «выбор файлов». Если пользователь предпочел тестировать алгоритмы классификации на встроенном наборе данных, то он подтверждает свой выбор нажатием кнопки отказа загрузки своего файла - «Cancel upload». В любом случае выбранный или загруженный пользователь набор данных отобразиться в виде интерактивной таблицы в подразделе «Отображение данных».

В разделе «Предварительный анализ данных» осуществляется расчет размерности данных и проверка на отсутствие пропусков в данных. При этом сначала рассчитывается количество объектов в используемой выборке и количество признаков данных (рисунок 11).

Расчет пропуска данных осуществляется по каждому признаку отдельно. В итоге на экране будет отображена таблица с результатом подсчета пропусков и если пропусков не будет обнаружено, то приложение продолжит свою работу. В противном случае на экране будет отображено предупреждение и пользователю будет предложено удалить все объекты из выборки, в которых содержатся пропуски.

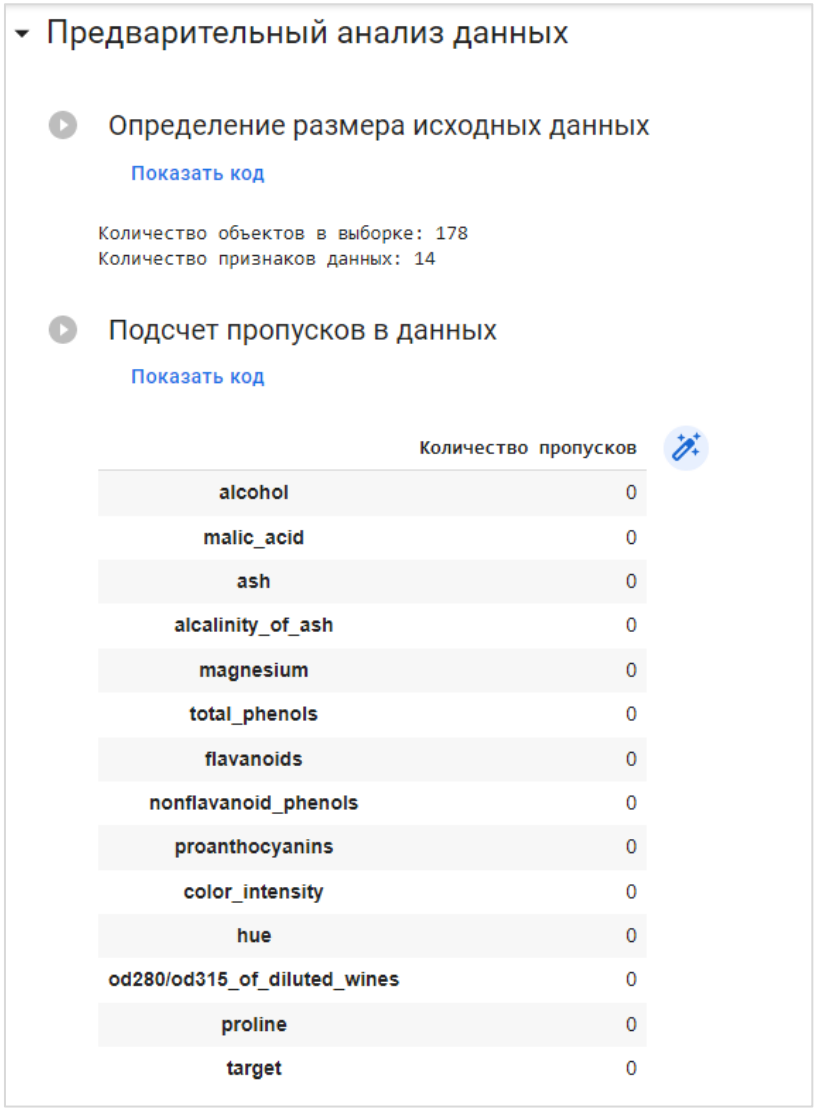


Рисунок 11 – Интерфейс раздела «Предварительный анализ данных»

В разделе «Подготовка тренировочной и тестовой выборок данных» пользователю предлагается задать параметр, определяющий в каких пропорциях исходные данные будут разделены на тестовую и тренировочную выборку. Наличие общих объектов в текстовой и тренировочной выборках данных не допускается. Пропорция распределения объектов по выборкам данных задается через слайдер «test_size» (рисунок 12). Значение равное 0,3 означает, что исходные данные между тестовой и обучающей выборкой будут распределены в пропорции 30/70. Для обеспечения наглядности в этом разделе интерфейса также предусмотрено

отображение в виде таблиц нескольких объектов обучающей (тренировочной) выборки данных и тестовой выборки данных.

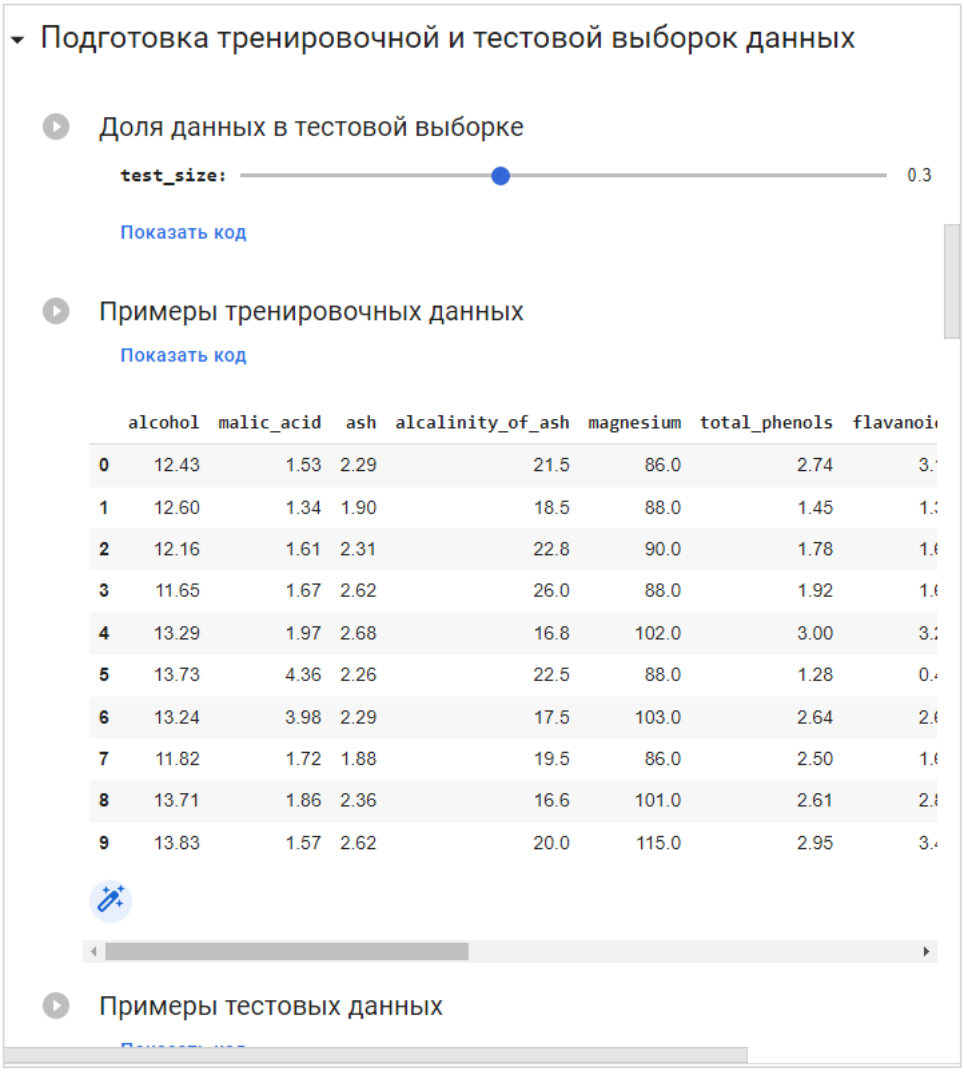


Рисунок 12 – Интерфейс раздела «Подготовка тренировочной и тестовой выборок данных»

В разделе «Сравнительный анализ алгоритмов классификации данных» пользователю отображаются результаты тестирования различных алгоритмов классификации. При этом для каждого классификатора сторонится матрица ошибок, полученная на основе результатов его работы на тестовых данных. Также для каждого классификатора определяются такие precision, recall, f1-score и support (рисунок 13).

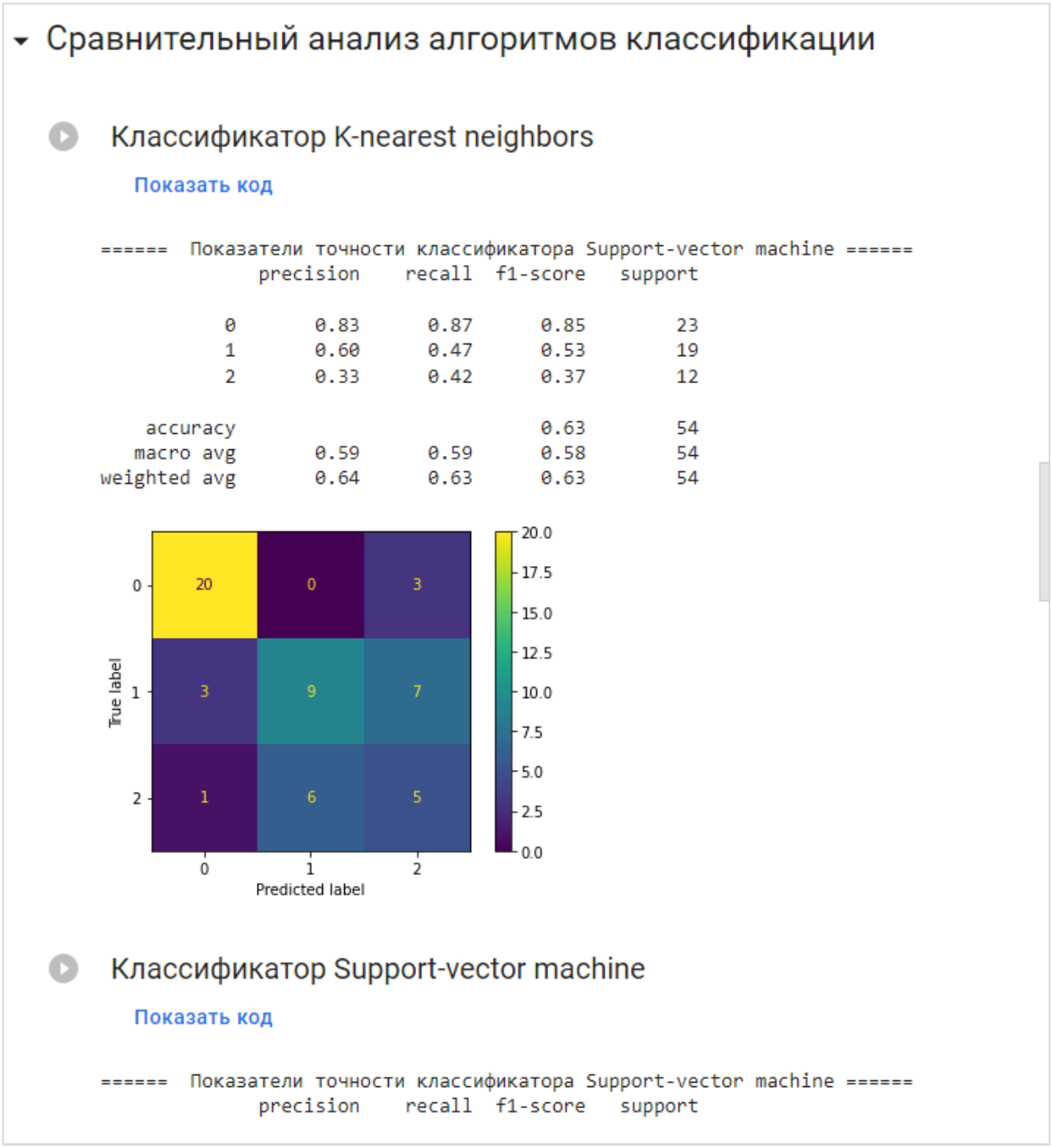


Рисунок 13 – Интерфейс раздела «Подготовка тренировочной и тестовой выборок данных»

Для случая, если пользователю требуется определить лучший алгоритм классификации для имеющихся данных без детализации особенностей их работы в приложении предусмотрено краткое представление результатов сравнительного анализа алгоритмов в виде таблицы (рисунок 14).

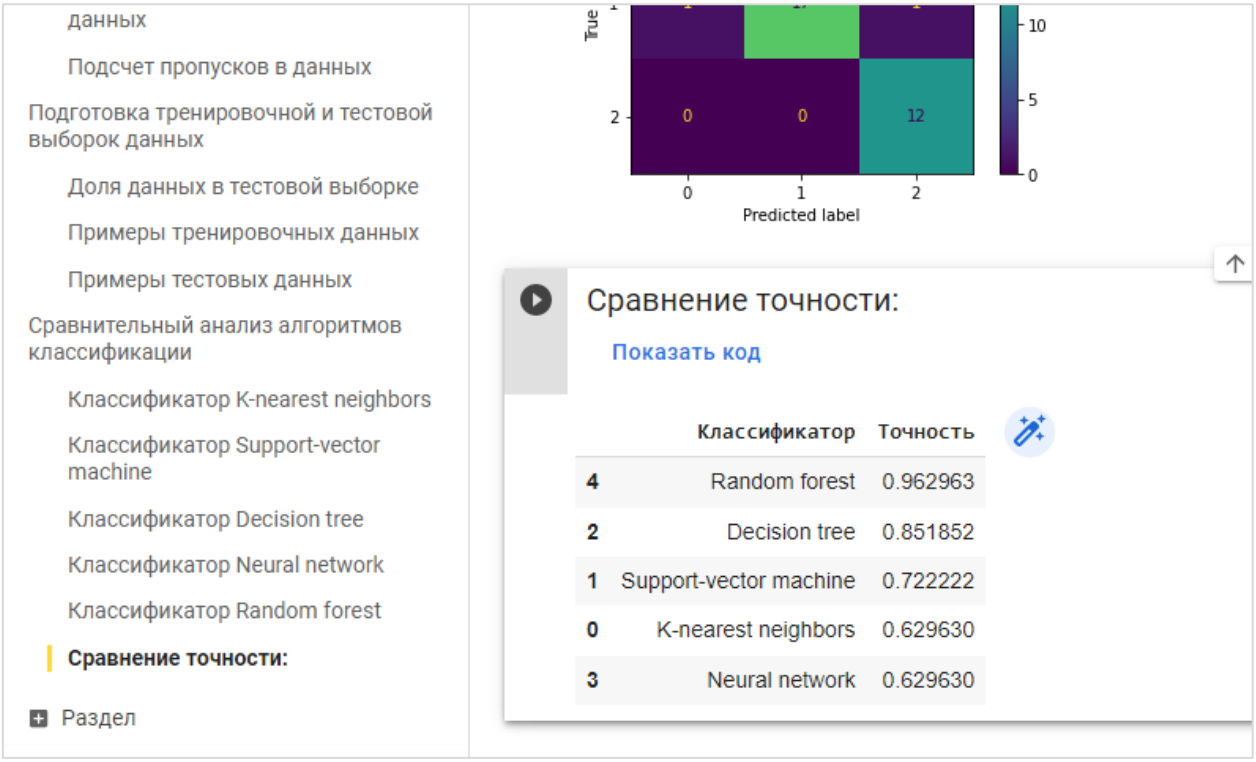


Рисунок 14 – Сравнение точности работы алгоритмы

В данной таблице приведены все протестированные алгоритмы. Значения в таблице отсортированы от лучшего по точности алгоритма к худшему.

3.3 Проектные решения используемые в приложении

Разработка программного обеспечения осуществлялась на языке программирования Python, в облачной среде Google colab. В приложении используются различные дополнительные библиотеки, благодаря которым удалось значительно сократить количество программного кода:

– методы библиотеки sklearn используются для разделения данных на обучающую и тестовую выборки, моделирования работы алгоритмов машинного обучения и загрузки наборов данных «Ирисы Фишера» и «Классификация вина»;

– методы библиотеки pandas применяются для импорта данных из

файлов формата xslx и csv с последующей визуализацией их в виде таблиц, а также для определения размерности данных и подсчета количества пропусков;

– методы библиотеки matplotlib применяются для визуализации матрицы ошибок в виде тепловой карты;

– методы библиотеки colab применяются для создания графического интерфейса приложения за счет использования виджетов и интерактивных форм.

Одним из первых этапов работы приложения является выбор пользователем данных, на которых будут тестироваться алгоритмы классификации. Для создания стандартного окна выбора пользователем файла использовался метод files.upload(), который выводит на экран пользователя стандартные элементы управления. Программный кода показан на рисунке 15.

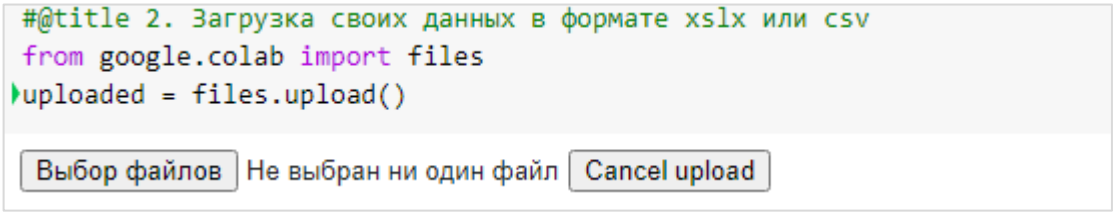


Рисунок 15 – Программный код для создания диалогового окна с выбором файла

В случае если пользователь отказался от загрузки своего файла, с помощью оператора условного перехода происходит проверка, какой элемент выпадающего списка со встроенными в приложение наборами данных выбран. Выбранное элемент списка храниться в переменной dataset. Если выбранный элемент – ‘Ирисы Фишера’, то вызывается метод load_iris(), который загружает в переменную df_data требуемый набор данных. В противном случае вызывается метод load_wine(). Так как методы load_iris() и

load_wine() загружают данные в своем специализированном формате, то перед тем как отобразить загруженные данные на экране их нужно объединить в одну таблицу. При объединении данных в одну таблицу, хранящейся в переменной data, применяется метод np.c_, программный код показан на рисунке 16.

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.datasets import load_wine

#@title Отображение данных:
if dataset=="Ирисы Фишера":
  df_data = load_iris()
else:
  df_data = load_wine()
data = pd.DataFrame(data= np.c_[df_data['data'], df_data['target']],
                    columns= df_data['feature_names'] + ['target'])
data
```

Рисунок 16 – Программный код для работы со встроенными наборами данных

Так как типом переменной data является dataframe, то появляется возможность получения доступа к свойствам данных через поле shape. Это используется на этапе предварительного анализа данных для получения значений количества объектов в выборке и количества признаков в данных. Программный код показан на рисунке 17.

```
#@title Определение размера исходных данных
print("Количество объектов в выборке:", data.shape[0])
print("Количество признаков данных:", data.shape[1])
```

Рисунок 17 – Программный код для получения свойств данных

На этапе проверки отсутствия пропусков в данных к переменной data

последовательно применяется сначала функция isna(), а затем sum(). Таким образом обеспечивается подсчет пропущенных значений по каждому признаку данных отдельно. Для отображения результатов на экране в удобном для восприятия виде посредством библиотеки pandas создается интерактивная таблица. Программный код показан на рисунке 18.

```
#@title Подсчет пропусков в данных
pd.DataFrame(data.isna().sum(), columns=['Количество пропусков'])
```

Рисунок 18 – Программный код для подсчёта количества пропусков в данных

Перед началом тестирования алгоритмов классификации необходимо сформировать две выборки данных: тренировочную выборку данных для обучения классификаторов и тестовую для проверки их эффективности. В приложении данный этап реализован с использованием метода train_test_split, который получает данный исходную выборку данных посредством переменной data, перемешивает ее и делит на два подмножества в соответствии с требуемыми долями, заданными через значение параметра test_size. Результат разделения данных сохраняется в 4 переменных: признаки объектов помещаются в переменные X_train, X_test, а значения меток класса – в переменные y_train, y_test (рисунок 19).

Следующем шагом является отображение на экране посредством метода head() нескольких примеров объектов из получившихся выборок данных.

```
#@title Доля данных в тестовой выборке
test_size = 0.3 #@param {type:"slider", min:0.1, max:0.6, step:0.1}

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df_data.data, df_data.target,
                                                    test_size=test_size,
                                                    random_state=3)

#@title Примеры тренировочных данных
pd.DataFrame(X_train, columns=data.columns[:-1]).head(10)

#@title Примеры тестовых данных
pd.DataFrame(X_test, columns=data.columns[:-1]).head(10)
```

Рисунок 19 – Программный код для разделения исходных данных на тренировочную и тестовую выборки

Для создания экземпляра классификатора вызывается метод-конструктор, название которого соответствует названию алгоритма. Затем с помощью метода fit() производится его обучение на обучающем наборе данных.

Для тестирования эффективности работы алгоритмов классификации данных применяются 4 основных программный метода, реализованных библиотекой sklearn:

– classification_report(), который подсчитывает отдельно для каждого класса количество ошибок совершенных классификатором на тестовой выборке данных;

– confusion_matrix(), который по результатам тестирования формирует матрицу ошибок;

– ConfusionMatrixDisplay(), который визуализирует матрицу ошибок посредством библиотеки matplotlib;

– accuracy_score(), который рассчитывает итоговую точность работы классификатора на тестовой выборке данных (рисунок 20).

```
#@title Классификатор K-nearest neighbors
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import ConfusionMatrixDisplay, accuracy_score

from sklearn.neighbors import KNeighborsClassifier
knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train, y_train)

y_pred_knn = knn_clf.predict(X_test)
print("======  Показатели точности классификатора K-nearest neighbors ======")
print(classification_report(y_test, y_pred_knn))
confmat_knn = confusion_matrix(y_true=y_test, y_pred=y_pred_knn)
disp = ConfusionMatrixDisplay(confusion_matrix=confmat_knn)
disp.plot()

accuracy_knn = accuracy_score(y_test, y_pred_knn)
```

Рисунок 20 – Программный код тестирования алгоритма классификации k-ближайших соседей

рисун

Аналогичный программный код применяется и для тестирования других алгоритмов классификации.

```
#@title Сравнение точности:

df = pd.DataFrame([['K-nearest neighbors', accuracy_knn],
                   ['Support-vector machine', accuracy_svm],
                   ['Decision tree', accuracy_dtc],
                    ['Neural network', accuracy_nn],
                    ['Random forest', accuracy_rf]],
columns=['Классификатор', 'Точность'])
df.sort_values('Точность', ascending=False)
```

Рисунок 21 – Программный код для вывода таблицы с результатами работы алгоритмов классификации данных

Визуализация результатов тестирования всех алгоритмов классификации обеспечивается с помощью библиотеки pandas. Для этого единый массив собираются результаты тестирования точности алгоритмов

(переменные accuracy_knn, accuracy_svm, accuracy_dtc, accuracy_nn, accuracy_rf) и их названия в виде строковых переменных. Затем через параметр columns задаются название столбцов. Таким образом в переменной df храниться табличное представление полученных результатов тестирования алгоритмов классификации.

Для сортировки значений в таблице в порядке снижения эффективности алгоритмов применяется метод sort_values(). Для применения данного метода к данным хранящимся в df достаточно написать его название символ точки. Для того чтобы значения шли в порядке убывания используется параметр ascending=False, а также задается название столбца, по которому будет выполняться сортировка. Программный код представлен на рисунке 21.

Для генерации графического представления таблицы и вывода ее на экран требуется в коде названия переменной df.

Теперь рассмотрим примеры работы приложения.

3.4 Пример работы приложения

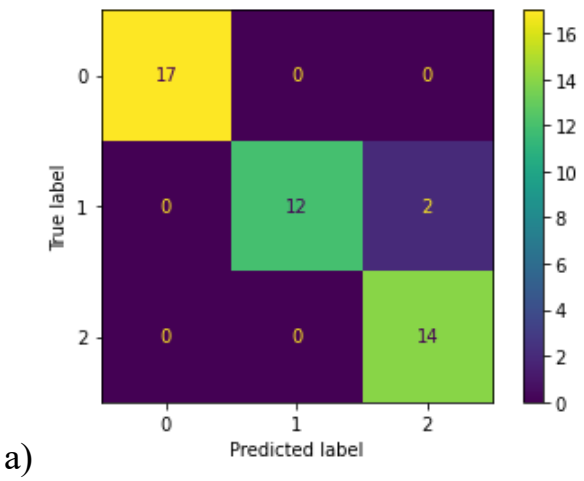
Тестирование работы приложения осуществлялось на двух наборах данных:

- ирисы Фишера;
- классификация вина.

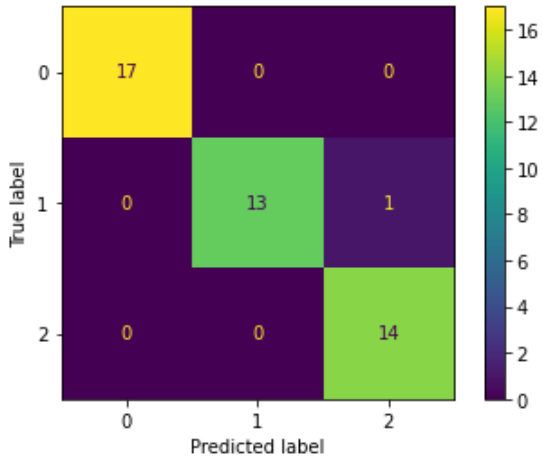
Первый набор данных содержит в себе 150 записей, каждая из которых описана четырьмя числовыми признаками. Числовые признаки – это геометрические параметры цветов (ирисов). Требуется обучить классификатор, который по этим параметрам может максимально точно определять сорт цветов. В исходной выборке данных представлены 3 класса цветов. В обучающую выборку данных было помещено 105 записей, а в тестовую выборку 45 записей (для этого в приложении была задано значения

параметра test_size равно 0,3). Были протестированы разные алгоритмы классификации данных. По результатам тестирования каждого классификаторов были построены матрицы ошибок, показанные на рисунке 22: а – матрица ошибок K-nearest neighbors, б – матрица ошибок Support-vector machine, в – матрица ошибок Decision tree, г – матрица ошибок Neural network, д – матрица ошибок Random forest, е – итоговая точность алгоритмов классификации.

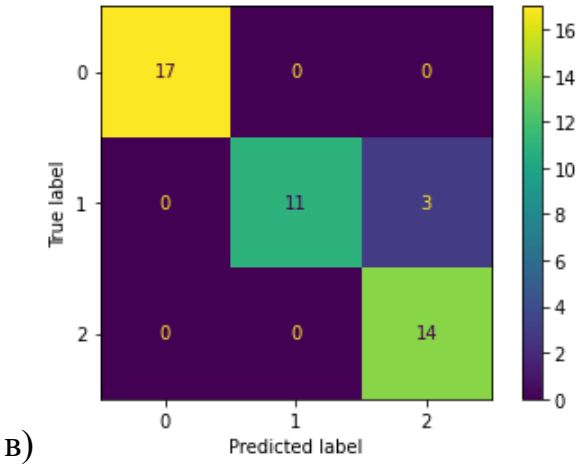
Второй набор данных состоит из 178 записей, каждая из которых описана 13 числовыми признаками. Числовые признаки – это химические и физические параметры вина (цветовой оттенок, кислотность, доля алкоголя и т.д.). Требуется обучить классификатор, который по этим параметрам может определять тип вина. В исходной выборке данных представлены 3 типа вина. В результате распределения исходных данных в обучающую выборку было помещено 124 элемента, а в тестовую выборку - 54 записи (пропорция распределения 70 на 30). На этих данных также были протестированы разные алгоритмы классификации данных. Полученные матрицы ошибок и таблица с точностью работы алгоритмов показаны на рисунке 22: а – матрица ошибок K-nearest neighbors; б – матрица ошибок Support-vector machine; в – матрица ошибок Decision tree; г – матрица ошибок Neural network; д – матрица ошибок Random forest; е – итоговая точность алгоритмов классификации.



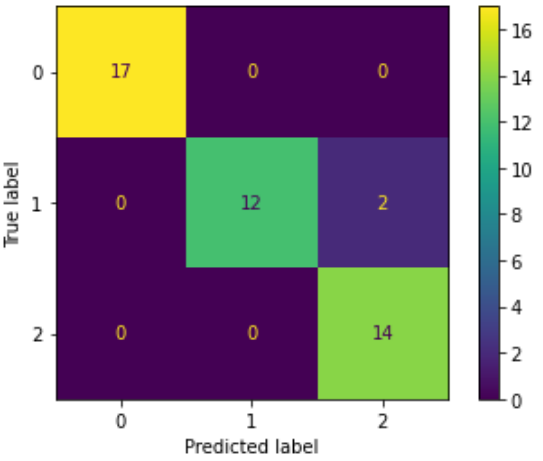
а)



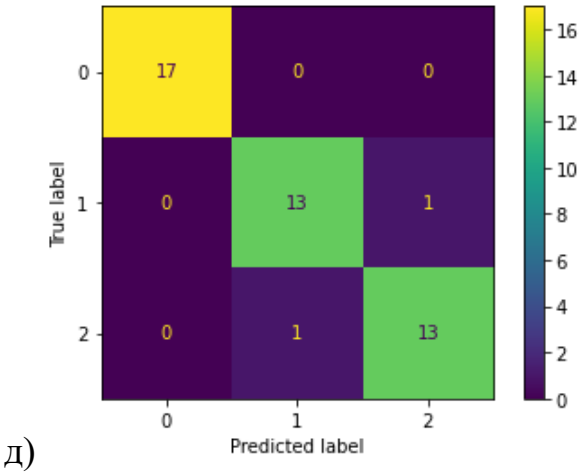
б)



в)



г)

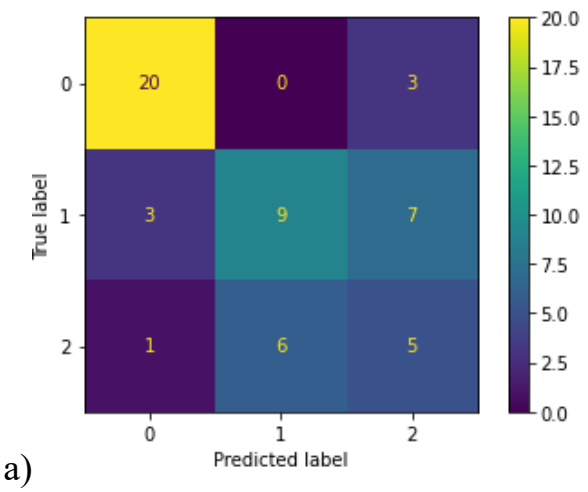


д)

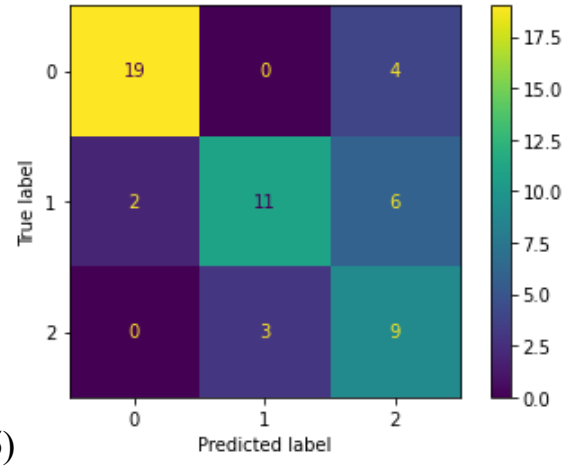
е)

Классификатор	Точность
Support-vector machine	0.977778
K-nearest neighbors	0.955556
Neural network	0.955556
Random forest	0.955556
Decision tree	0.933333

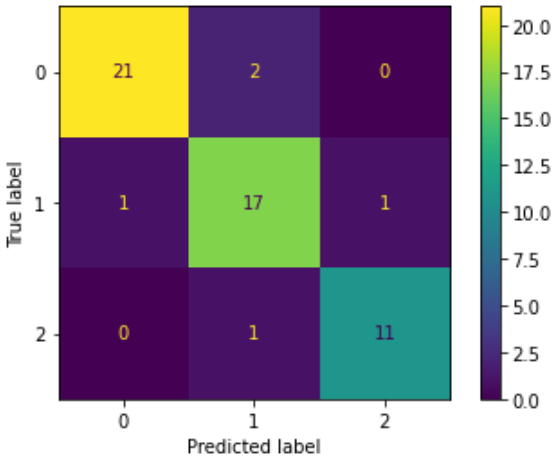
Рисунок 22 – Результаты тестирования эффективности алгоритмов на наборе данных «Ирисы Фишера»



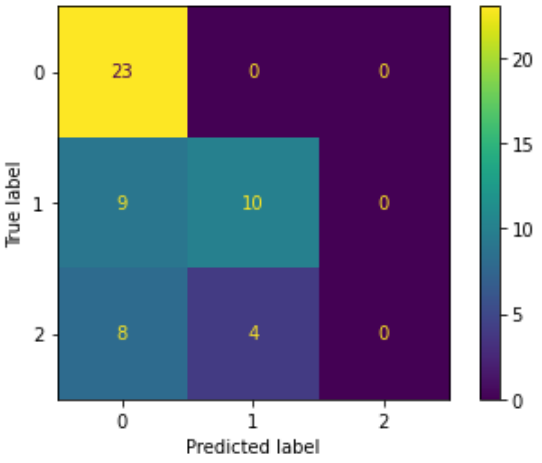
а)



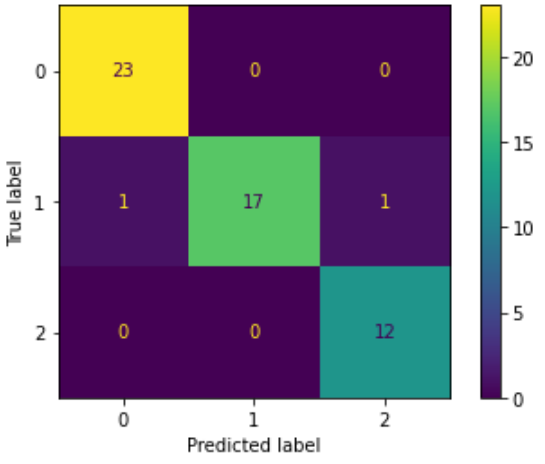
б)



в)



г)



д)

е)

Классификатор	Точность
Random forest	0.962963
Decision tree	0.907407
Support-vector machine	0.722222
K-nearest neighbors	0.629630
Neural network	0.611111

Рисунок 23 – Результаты тестирования эффективности алгоритмов на наборе данных «Классификация вина»

На основе полученных результатов тестирования алгоритмов можно сделать следующие выводы. Для построения классификатора на наборе данных «Ирисы Фишера» лучше использовать Support-vector machine, т.к. данный алгоритм обеспечивает максимальную точность классификации, равную 0,97. Как показывает матрица ошибок, данный классификатор на тестовых данных совершил единственную ошибку на объекте данных из класса с меткой «1».

Для набора данных «Классификация вина» лучшим классификатором оказался Random forest, с точность классификации равной 0,96.

Выводы к главе 3

В ходе написания 3 главы были сформулированы следующие выводы:

- на языке Python разработано приложения для тестирования эффективности алгоритмов классификации, которое позволяет: импортировать данные для последующего анализа из файлов формата xslx и csv, строить классификаторы с использованием различных алгоритмов машинного обучения (knn, random forest, svm, decision tree, neural network), тестировать построенные классификаторы для оценки точности их работы, просматривать результаты работы классификаторов в виде матрицы ошибок и сравнивать эффективность работы алгоритмов классификации;
- показана работа приложения выборках данных «Ирисы Фишера», «Классификация вина», а результаты тестирования показали работоспособность программного обеспечения для сравнения эффективности работы классификаторов данных.

Заключение

Приведем в качестве заключения основные выводы и результаты, полученные при выполнении бакалаврской работы:

- анализ литературных источников показал, что развитие машинного обучения привело к возникновению большого числа алгоритмов, способных решать задачу классификации данных. Но так как заранее никогда не известно, какой из алгоритмов окажется наиболее эффективным при анализе имеющегося набора данных, актуальной задачей становится разработка программного обеспечения для сравнения эффективности алгоритмов;
- проведен анализ способов применения алгоритмов машинного обучения для решения практических задач из различных областей, выделены наиболее часто используемые алгоритмы классификации данных и проведен их сравнительный анализ с точки зрения взаимодействия с обучающей выборкой данных.
- рассмотрена формальная постановка задачи классификации, метрики, используемые для оценки точности работы классификаторов, особенности использования матрицы ошибок для контроля корректности работы классификаторов;
- разработана методика сравнения эффективности работы алгоритмов для классификации данных, которая включает в себя разделение исходных данных на обучающую и тестовую выборки, использование обучающей выборки данных для построения множества классификаторов основанных различных алгоритмах (knn, random forest, svm, decision tree, neural network), проверку точности работы классификаторов на тестовой выборке и визуализацию работы классификаторов с использованием матрицы ошибок;
- на языке Python разработано приложения для тестирования

эффективности алгоритмов классификации, которое позволяет: импортировать данные для последующего анализа из файлов формата xslx и csv, строить классификаторы с использованием различных алгоритмов машинного обучения (knn, random forest, svm, decision tree, neural network), тестировать построенные классификаторы для оценки точности их работы, просматривать результаты работы классификаторов в виде матрицы ошибок и сравнивать эффективность работы алгоритмов классификации;

− показана работа приложения выборках данных «Ирисы Фишера», «Классификация вина», а результаты тестирования показали работоспособность программного обеспечения для сравнения эффективности работы классификаторов данных.

Таким образом можно заключить, что поставленная цель была достигнута, а все задачи исследования решены в полном объеме.

Практическая значимость работы заключается в разработке исследовательского программного обеспечения. Представленное программное обеспечение использоваться разработчиками интеллектуальных система для предварительного анализа эффективности алгоритмов на имеющихся данных.

Список используемой литературы и используемых источников

1. Бутакова, М.А. Классификация потоков данных и метод статистического моделирования в системах и сетях телекоммуникаций на транспорте / М.А. Бутакова // Вестник ростовского государственного университета путей сообщения. №2, 2005. – Ростовский государственный университет путей сообщения (Ростов-на-Дону), 2005. – с. 38-43.

2. Власов, А.В. Машинное обучение применительно к задаче классификации семян зерновых культур в видеопотоке / А.В. Власов, А.С. Федеев // Молодежь и современные информационные технологии – сборник трудов XIV Международной научно-практической конференции студентов, аспирантов и молодых учёных, 07–11 ноября 2016. – Национальный исследовательский Томский политехнический университет (Томск), 2016. – с. 133-135.

3. Жуков, Д.А. Формирование контрольных выборок при технической диагностике объекта с применением машинного обучения / Д.А. Жуков, А.С. Хорева, Ю.Е. Кувайскова, В.Н. Клячкин // Математические методы и модели: теория, приложения и роль в образовании – международная научно-техническая конференция : сборник научных трудов, 28–30 апреля 2016 года. – Ульяновский государственный технический университет (Ульяновск), 2016. – с. 44-48.

4. Иванников Ю.Ю. Применение методов машинного обучения для выявления бот-трафика среди запросов к веб-приложению / Ю.Ю. Иванников, Е.Ю. Митрофанова // Сборник студенческих научных работ факультета компьютерных наук ВГУ, Факультет компьютерных наук, 2017. – ФГБОУ ВО «Воронежский государственный университет», 2017. – с. 119-123.

5. Клячин В.Н. Использование агрегированных классификаторов при технической диагностике на базе машинного обучения / В.Н. Клячин,

Ю.Е. Кувайскова, Д.А. Жуков // Информационные технологии и нанотехнологии (ИТНТ-2017) – сборник трудов III международной конференции и молодежной школы. Самарский национальный исследовательский университет имени академика С.П. Королева. 2017. – Предприятие "Новая техника" (Самара), 2017. – с. 1770-1773.

6. Кононова, Н.В. Исследование подсистемы контентной фильтрации с использованием методов машинного обучения / Н.В. Кононова, Ю.А. Андрусенко, Т.А. Самокаева // Студенческая наука для развития информационного общества – сборник материалов VI Всероссийской научно-технической конференции. 22–26 мая 2017. – Северо-Кавказский федеральный университет (Ставрополь), 2017. – с. 268-270.

7. Малинин, П.В. Иерархический подход в задаче идентификации личности по голосу с помощью проекционных методов классификации многомерных данных / П.В. Малинин, В.В. Поляков // Доклады томского государственного университета систем управления и радиоэлектроники. №21, 2010. – Томский государственный университет систем управления и радиоэлектроники (Томск), 2010. – с. 128-130.

8. Мелдебай, М.А. Анализ мнений покупателей на основе машинного обучения / М.А. Мелдебай, А.К. Сарбасова // Прикладная математика и информатика: современные исследования в области естественных и технических наук – материалы III научно-практической всероссийской конференции (школы-семинара) молодых ученых. 24–25 апреля 2017 года. – Издатель Качалин Александр Васильевич, 2017. – с. 360-363.

9. Наумов, Д.П. Регулятор САР на основе машинного обучения / Д.П. Наумов, Д.П. Стариков // Информационные технологии в управлении, автоматизации и мехатронике – сборник научных трудов Международной научно-технической конференции. 06–07 апреля 2017 года. – ЗАО "Университетская книга" (Курск), 2017. – с. 106-114.

10. Осколков, В.М. Использование метода машинного обучения для повышения продуктивности на предприятии / В.М. Осколков, Н.И. Шаханов, И.А. Варфоломеев, О.В. Юдина, Е.В. Ершов // Автоматизация и энергосбережение машиностроительного и металлургического производств, технология и надежность машин, приборов и оборудования – материалы XII Международной научно-технической конференции, 21 марта 2017. – Вологодский государственный университет (Вологда), 2017.

11. Рагимов Э.Р. Классификация угроз по сервисам безопасности на основе статистических данных / Э.Р. Рагимов // Информационное противодействие угрозам терроризма. №5, 2005. – Технологический институт Федерального государственного образовательного учреждения высшего профессионального образования "Южный федеральный университет", 2005. – с. 87-92.

12. Соловьев, А.Ю. Применение машинного обучения для прогнозирования неблагоприятных исходов в ургентной хирургии / Соловьев А.Ю., Берегов М.М., Вахеева Ю.М., Баутин А.Н., Гусев А.В. // Медико-биологические, клинические и социальные вопросы здоровья и патологии человека – материалы III Всероссийской образовательно-научной конференции студентов и молодых ученых с международным участием в рамках XIII областного фестиваля "Молодые ученые - развитию Ивановской области". 2017. – Ивановская государственная медицинская академия (Иваново), 2017. – с. 129-130.

13. Ткач, Т.Ч. Машинное обучение и обработка больших данных - обучение в основной и средней школе / Т.Ч. Ткач // Актуальные проблемы методики обучения информатике и математике в современной школе – материалы международной научно-практической интернет-конференции. Московский педагогический государственный университет, Москва, 24 апреля 2020 года. – Московский педагогический государственный университет (Москва), 2020. – с. 217-223.

14. Федотов, И.А. Применение технологий машинного обучения для прогнозирования ситуации на финансовых рынках / И.А. Федотов // Студенческая наука для развития информационного общества – сборник материалов VI Всероссийской научно-технической конференции. – Северо-Кавказский федеральный университет (Ставрополь), 2017. – с. 361-363.

15. Френкель, Ф.Е. Классификация триплетной периодичности последовательностей ДНК генов, собранных в банке данных KEGG / Ф.Е. Френкель, Е.В. Коротков // Молекулярная биология. №4, 2008. – Российская академия наук (Москва), 2008. – с. 707-720.

16. Jiang, S. A Combined Classification Algorithm Based on C4.5 and NB / ShengYi Jiang, Wen Yu // International Symposium on Intelligence Computation and Applications – Third International Symposium, ISICA 2008 Wuhan, China, December 19-21, 2008. Proceedings: Advances in Computation and Intelligence. – Springer-Verlag Berlin Heidelberg 2008. – pp. 350-359.

17. Kong, X. Principal Component Analysis Networks and Algorithms / Xiangyu Kong, Changhua Hu, Zhansheng Duan. – Springer Singapore, 2017 – 323 p.

18. Liu, J. Radial Basis Function (RBF) Neural Network Control for Mechanical Systems: Design, Analysis and Matlab Simulation. – Springer Berlin Heidelberg, 2014 – 365 p.

19. Min, F. A Competition Strategy to Cost-Sensitive Decision Trees / Fan Min, William Zhu // International Conference on Rough Sets and Knowledge Technology – 7th International Conference, RSKT 2012, Chengdu, China, August 17-20, 2012. Proceedings: Rough Sets and Knowledge Technology. – Springer-Verlag Berlin Heidelberg 2012. – pp. 359-368.

20. Silva, I.N. Artificial Neural Networks: A Practicle course / Ivan Nunes da Silva, Danilo Hernane Spatti, Rogerio Andrade Flauzino, Luisa Helena Bartocci Liboni, Silas Franco dos Reis Alves. – Springer International Publishing, 2017 – 307 p.