Санкт-Петербургский государственный университет

Фундаментальная информатика и информационные технологии
Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Алимов Нурислам Ахтасович

Технология для моделирования и анализа систем классификации на основе машинного обучения

Магистерская диссертация

Научный руководитель:
д. ф.-м. н., профессор Граничин О. Н.

Рецензент:
м.н.с. ФГБУН ИПмаш РАН Ерофеева В. А.

Санкт-Петербург
2017

SAINT PETERSBURG UNIVERSITY

Fundamental Computer Science and Information Technologies
Mathematical and Software Support for Computers, Computer Systems and Networks

Alimov Nurislam

Modelling and Analysis Technology of Classification Systems Based on Machine Learning Approach

Master's Thesis

Scientific supervisor:
professor Oleg Granichin

Reviewer:
Victoria Erofeeva

Saint Petersburg
2017

Оглавление

1. Введение

Сложно переоценить важность обработки и анализа данных в современном мире. В различных областях жизнедеятельности, таких как медицина, промышленность, энергетика и других накапливается огромное количество данных, зачастую представленных в сложном для обработки и анализа человеком формате.

Мотивацией для исследования послужил тот факт, что в настоящее время растёт интерес научного сообщества и бизнеса к технологиям и методам машинного обучения и анализа данных. Также, наблюдается рост количества различных программных реализаций существующих алгоритмов и подходов.

Но сегодня, чтобы начать решать какую-либо реальную задачу машинного обучения (Machine Learning, ML), необходимо обладать достаточно высокой компетенцией и разбираться не только в математическом аппарате, но и в соответствующих программных продуктах. Сейчас набор технологий представлен в виде большого количества различных продуктов, чаще всего несовместимых между собой, научиться пользоваться которым в котором достаточно сложно.

На заре бурного развития технологий и методов машинного обучения каждый исследователь самостоятельно разрабатывал себе программное обеспечение, бравшее на себя рутинные операции по первоначальной обработке данных и поиска основной линии (baseline), являющаяся в общем смысле отправной точкой при решении задачи машинного обучения и представляющую собой метод обработки данных. Основная линия (baseline) даёт приемлемую величину метрики (мера оценки точности работы алгоритма), которую можно получить за минимальное время и минимальными усилиями. Позже начали появляться системы - агрегаторы, позволяющие автоматизировать повторяющиеся операции. Хотя подобные системы начали появляться сравнительно недавно, попытки их разработки предпринимались и ранее. Значительных успехов эти ранние реализации достичь так и не смогли, что можно объяснить недостаточным на тот момент развитием вычислительных

мощностей и ограниченной доступностью программных продуктов.

Системы-агрегаторы состоят из нескольких компонентов. Архитектура является основой компонентой подобных систем, так как она является каркасом, определяющий формальное поведение системы, от неё зависит какие структуры и паттерны используются для решения задач.

Программная реализация — компонента, которая определяет как будет работать система, какие нагрузки она должна выдерживать и какие данные будет обрабатывать. Программной реализацией является код, написанный на одном из языков программирования, зачастую высокоуровневом и объектно-ориентированном.

После того, как разработана архитектура и выполнена программная реализация, следующим следует апробация и сравнение с конкурентными аналогами. Каждая такая система уникальна, хотя все они имеют схожие черты, какие-то обладают неоспоримыми преимуществами перед другими, какие-то разработаны для решения узкого круга задач.

Тем, как быстро и легко можно с помощью системы получить решение задачи, определяется конкурентоспособность продукта на рынке. Примеры задач — классификация и кластеризация, которые имеют важное практическое применение. Например, в медицине актуальной является задача о предсказании успешности применения дефибрилляции. Дефибрилляция — вид электроимпульсной терапии, проводимой при нарушениях сердечного ритма, она восстанавливает ритм сердца путём проведения электрического тока через сердце. Задача о предсказании в этом контексте трактуется как задача классификации набора характеристик, полученных из кардиограммы пациента на два класса (ROEA — дефибрилляция оказалась успешной и NOROEA — дефибрилляция не помогла).

Подводя итог вышесказанному, можно заключить, что существует потребность рынка в такой системе, и исследование, предложенное в этой работе является актуальным, мотивация для его проведения является обоснованной.

2. Постановка задачи

Цель работы — разработать технологию, позволяющую эффективно решать задачи классификации и кластеризации. Для достижения этой цели бы сформулированы следующие подзадачи:

- Разработать архитектуру системы;
- Программно реализовать систему, согласно предложенной архитектуре;
- Провести апробацию реализованной системы;
- Продемонстрировать работу системы на основе трёх задач.

Ниже представлены основные требования, предъявляемые к системе, на основе которых будет выноситься суждение об успешности проводимого исследования:

1. *Возможность загружать данные.* В системе должна быть предусмотрена возможность добавлять свои данные, представленные в определенном формате. Примером таких данных может быть текстовый формат csv;

- *Добавлять свои алгоритмы.* Должна быть реализована возможность загружать алгоритмы, написанные на языке Python в виде скрипта в определённом виде;
- *Использовать существующие алгоритмы.* Помимо загрузки своих алгоритмов система должна обладать уже предзагруженными алгоритмами, которые пользователь может переиспользовать;
- *Находить наилучший алгоритм из загруженных.* Другими словами, необходимо наличие функции поиска baseline по имеющимся данными и алгоритмам;
- *Быстрый старт.* Настройка и подготовка системы, равно как и обучение пользователя, должны занимать немного времени;

- *Поддержка локализации.* Система должна быть мультиязычной, а именно должна быть поддержка как минимум трёх языков: русского, английского и китайского.

3. Обзор

3.1. Краткий обзор аналогов

На сегодняшний день, существует несколько подобных систем. Сделаем краткий обзор этих систем:

- *Amazon Machine Learning (компания Amazon)* [2]. Как известно компания Amazon один из лидеров по предоставлению AWS (Amazon Web Services - облачные услуги). Разработчикам компании удалось создать достаточно мощную систему для анализа данных и машинного обучения, поддерживающую многие из современных языков. Из недостатков можно выделить высокую цену и закрытый код, сложность настройки рабочего окружения, ориентированность на API (в текущем контексте - возможность обращаться к вычислительным ресурсам удалённо) и на использование серверов компании.

- *Azure Machine Learning (компания Microsoft)* [16]. Позволяет создавать свои проекты, использовать встроенные алгоритмы, а также загружать свои алгоритмы на различных языках программирования. Позволяет производить анализ результатов. Из недостатков можно выделить достаточно высокую цену, слабые возможности настройки и закрытую реализацию.

- *Google Cloud Machine Learning (компания Google)* [9]. Компания Google создала по настоящему “современный” продукт, способный интегрироваться во все современные технологии и использовать самые передовые разработки в области машинного обучения. Несмотря на свои преимущества, система обладает излишней сложностью, перегруженностью и высокой ценой, вкупе с закрытым кодом.

В таблицах 1-3 представлены основные отличия разработанной системы от аналогов.

№	Система	проприетарная лицензия
1	Amazon Machine Learning	+
2	Azure Machine Learning	+
3	Google Cloud Machine Learning	+
4	Разработанная	-

Таблица 1: Таблица сравнения систем (1)

№	программная расширяемость	быстрый старт
1	+	-
2	+	-
3	-	-
4	+	+

Таблица 2: Таблица сравнения систем (2)

№	baseline	простота установки и развёртывания
1	-	+
2	-	-
3	-	-
4	+	+

Таблица 3: Таблица сравнения систем (3)

В ходе диссертационного исследования были рассмотрены эти существующие решения. У всех подобных систем есть недостатки и ограничения. Вот основные из них:

- Сложная настройка (долгий старт);

- Проприетарная лицензия;

- Нельзя менять поведение системы по своему усмотрению;

- Некоторые из них заточены только под нейронные сети.

3.2. Машинное обучение

Машинное обучение и анализ данных находятся на стыке областей математики, программирования и математической статистики. В общем случае, машинное обучение представляет собой поиск зависимостей между данными и использование найденной зависимости для решения поставленной изначально задачи.

Формально, машинное обучение — математическая дисциплина, выделяющая знания из данных, использующая разделы математической статистики, численных методов оптимизации, теории вероятности и дискретного анализа.

Любая задача в машинном обучении формально записывается как: X — объекты (множество), Y — ответы (множество), $y : X \to Y$ — некоторая зависимость (target function).

Дано: $\{x_i, \ldots, x_l\} \subset X$ — обучающая выборка,
$y_i = y(x_i)$, $i = 1, \ldots, l$ — ответы (известные).

Требуется: найти: решающую функцию (decision function) s, приближающую y на множестве X.

В свою очередь, машинное обучение подразделяется на два подтипа, различающихся способом того, как модель будет обучена:

1. Обучение с учителем.

   Этот случай является наиболее распространённым. Необходимо найти функциональную зависимость между объектами и ответами.

2. Обучение без учителя.

   Требуется искать зависимости между объектами, так как ответы не задаются. Примером может служить задача кластеризации, ко-

гда у нас есть только множество объектов и из нужно разделить на несколько классов.

3.3. Аппроксимация функции с помощью линейной комбинации известных функций

Задача об аппроксимации функции по её значениям имеет большое количество областей применения, в частности в машинном обучении.

Предположим, что в точках $x_1, x_2, \ldots, x_N$ известны значения функции $g(x)$, определенной на некотором множестве $X, x_i \in X, i = 1, 2 \ldots N$. Пусть задан набор функций $a_1(\cdot), a_2(\cdot) \ldots a_r(\cdot)$, определенных на том же множестве X. Требуется аппроксимировать функцию $y(\cdot)$ с помощью линейной комбинации функций $a_i\cdot$, т.е. подобрать набор параметров (коэффициентов):

$$\theta = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \ldots \\ \theta_r \end{pmatrix}$$

,

при котором функция

$$s(x, \theta) = \sum_{i=1}^{r} a_i(x)\theta_i,$$

в соответствии с некоторым функционалом качества, наилучшим образом представляет $y(x)$. Часто рассматривают функционал качества

$$f(\theta) = \int ||y(x) - s(x, \theta)||^2 P(dx)$$

являющийся, по сути, мерой среднеквадратичной ошибки аппроксимации.

Предположим, что задан некоторый критерий качества аппроксимации и уже найден определяемый этим критерием набор θ.

Обозначим

$$\phi_n = \begin{pmatrix} a_1(x_n) \\ a_2(x_n) \\ \dots \\ a_r(x_n) \end{pmatrix}$$

Значения $y(\cdot)$ в заданных точках $x_1, x_2, \dots, x_N$ можно представить в виде

$$y(x_n) = \phi_n^T \theta + \upsilon_n, n = 1, 2, , N,$$

в котором последовательность величин υ_n трактуется как ошибка измерения полезного сигнала

$$s(x_n, \theta) = \phi_n^T \theta, n = 1, 2, , N$$

3.4. Нейронные сети

Пусть имеется некоторая система, организованная следующим образом: при предъявлении ей входного сигнала (стимула) w она вырабатывает, в зависимости от значения матрицы параметров θ, значения функций $a_i(x, \theta)$ и вычисляет их сумму $s(x, \theta)$ с соответствующими весовыми коэффициентами [22, 24]. Так определяются множества (образы)

$$\mathbb{X}_1(\theta) = x : s(x, \theta) > 0, \mathbb{X}_2(\theta) = x : s(x, \theta) \leq 0$$

и рассматриваемая система в состоянии классифицировать любой входной сигнал x, относя его либо к множеству $\mathbb{X}_1$, либо к $\mathbb{X}_2$.

Описанная схема обучаемой системы может быть реализована с помощью так называемых перцептронов — сложных сетей из пороговых элементов.

Задача, изложенная выше является задачей самообучения или также называемой задачей обучения с учителем

Для построения последовательности оценок параметров нейронной сети можно воспользоваться градиентным спуском.

$$\hat{\theta_n} = \hat{\theta}_{n-1} - \alpha_n(s(x_n, \hat{\theta}_{n-1})) \bigtriangledown_\lambda s(x_n, \hat{\theta}_{n-1}), n = 1, 2 \dots$$

с некоторой последовательностью положительных чисел $\{\alpha_n\}, \alpha \to 0$. Компонентны вектора-градиента $\bigtriangledown_\lambda s(x, k)$ вычисляются по следующим формулам [23]:

$$\frac{ds(x,\lambda)}{dx_k} = d_2'(\sum_r^{i=1} \lambda_i d_1(\sum_p^{j=1} \lambda_{i,j} x_j)) d_1(\sum_{j=1}^{p} \lambda_{j,k} x_j)$$

$$\frac{ds(x,\lambda)}{dx_{k,m}} = d_2'(\sum_r^{i=1} \lambda_i d_1(\sum_p^{j=1} \lambda_{i,j} x_j)) \lambda_k d_1'(\sum_{j=1}^{p} \lambda_{j,k} x_j) x_m$$

$k = 1, 2, \dots, r,\ m = 1, 2, \dots, p.$

Задача самообучения тесно связана с задачей автоматической классификации и является обобщением последней на случай неизвестного распределения, определяющего статистику показа классифицируемых сигналов.

3.5. Кластеризация и классификация

Проблема классификации входных сигналов (стимуляторов, объектов) — типичная проблема в теории обучения. Процесс классификации возвращает сигналы, которые характеризуют группу объектов как набор данных или классы (кластеры). Следовательно каждый сигнал может быть отнесен к определенному классу:

$$\chi^k(\mathbb{X}) = \Omega_1, \dots, \Omega_k$$

где множество $\mathbb{X}$ содержит k не пустых кластеров

$$\mathbb{X} = \cup_{i=1}^{k} \Omega_i$$

и

$$\Omega_i \cap \Omega_j = \emptyset, i \neq j.$$

Для частей $\chi^k(\mathbb{X})$, метки функции $\gamma_{\chi^k} : \mathbb{X} \to 1..k$ соответствуют точке кластера, определяемого следующим образом:

$$\gamma_{\chi^k}(x) = i, \text{ if and only if } x \in \Omega_i, i \in 1..k.$$

Итак

$$\Omega_i = \{x \in \mathbb{X} | \gamma_{\chi^k}(x) = i\}.$$

Различным k из $\mathbb{X}$ соответствуют различные части $\chi^k(\mathbb{X})$.

Следующее свойство должно быть сформулировано математически: объекты, находящиеся в одной группе должны быть более близки, соответственно, различные группы предполагают большие различия между объектами.

Стандартный подход заключается в том, чтобы связать некоторые функции искажения (штрафа, качества) q_i, определяемые как "мера близости" кластера с остальными кластерами (группами) $i \in 1..k$, рассмотрим задачу минимизации:

$$f(\Omega^k, x) = \sum_{i=1}^{k} \gamma_{\chi^k}(x) q_i(\Omega^k, x) \to \min_{\Omega^k}$$

3.6. Оценка точности кластеризации

Как уже упоминалось, выбор модели, которая будет использоваться в кластерном анализе, и в частности оценка «истинного» количества кластеров, является достаточно сложной проблемой, имеющей существенное значение для кластерного анализа [21, 20].

Многие авторы предлагают использовать процедуры повторной дискретизации, которые основаны на концепции «стабильности кластера», в качестве подходов к проверке правильности формирования кластера [5, 8, 14, 18].

Рассмотрим одну из метрик, а именно скорректированный случай-

ный индекс [11]. Вычисление внешних индексов основано на так называемой таблице перекрестной табуляции или таблицы сопряженности. Сравниваем членство элементов в двух разбиениях X^r и X^s из одного и того же набора данных. Пусть $T_{i,j}$ обозначает количество элементов, которые являются членами в кластере i из X^r, и в j из $X^s, i \in 1..r,\ j \in 1..s$. Сравнения можем производить с помощью коэффициента корреляции Крамера [19, 25].

$$T_i^r = \sum_{j=1}^{s} T_{i,j}, i \in 1..r, T_j^s = \sum_{i=1}^{r} T_{i,j}, i \in 1..s,$$

Очевидно, что:

$$T = \sum_{i=1}^{s} T_i^r = \sum_{j=1}^{r} T_j^s,$$

Критерий хи-квадрат:

$$\chi^2 = \sum_{i=1}^{r}\sum_{j=1}^{s} \frac{(T_{i,j} - e_{i,j})^2}{e_{i,j}}, e_{i,j} = \frac{T_j^s T_i^r}{T}.$$

Коэффициент Крамера:

$$V = \sqrt{\frac{\chi^2}{T \min(r-1, s-1)}}.$$

Несколько известных внешних индексов используют статистику [15, 12, 7]:

$$Z = \sum_{i=1}^{r}\sum_{j=1}^{s} T_{i,j}^2.$$

$$R = 1 + \frac{Z - 0.5(\sum_{j=1}^{s}(T_j^s)^2) + \sum_{i=1}^{r}(T_i^r)^2)}{\binom{T}{2}}.$$

$$JD = \frac{(Z - T)}{(\sum_{j=1}^{s}(T_j^s)^2 + \sum_{i=1}^{r}(T_i^r)^2 - Z - T)}.$$

$$FM = \frac{(Z - T)}{2\sqrt{\binom{\sum_{j=1}^{s} T_j^s}{2}\binom{\sum_{i=1}^{r} T_i^r}{2}}}.$$

Внешний индекс нормирован таким образом, что 0 - элементы случайны, 1 - идеально совпадают.

Общая формула, предлагаемая для нормирования индекса, выглядит следующим образом [10]:

$$Ind' = \frac{Ind - E(ind)}{Ind_{max} - E(Ind)},$$

где Ind_{max} - максимальное значение индекса Ind.

4. Архитектура системы

Основные элементами архитектуры системы являются:

- Структура базы данных;
- Взаимодействие клиента и сервера;
- Способ хранения моделей/данных/результатов;
- Способ осуществления анализа и обработки результатов.

Структура базы данных

Была разработана структура базы данных, представленная на рисунке 1.

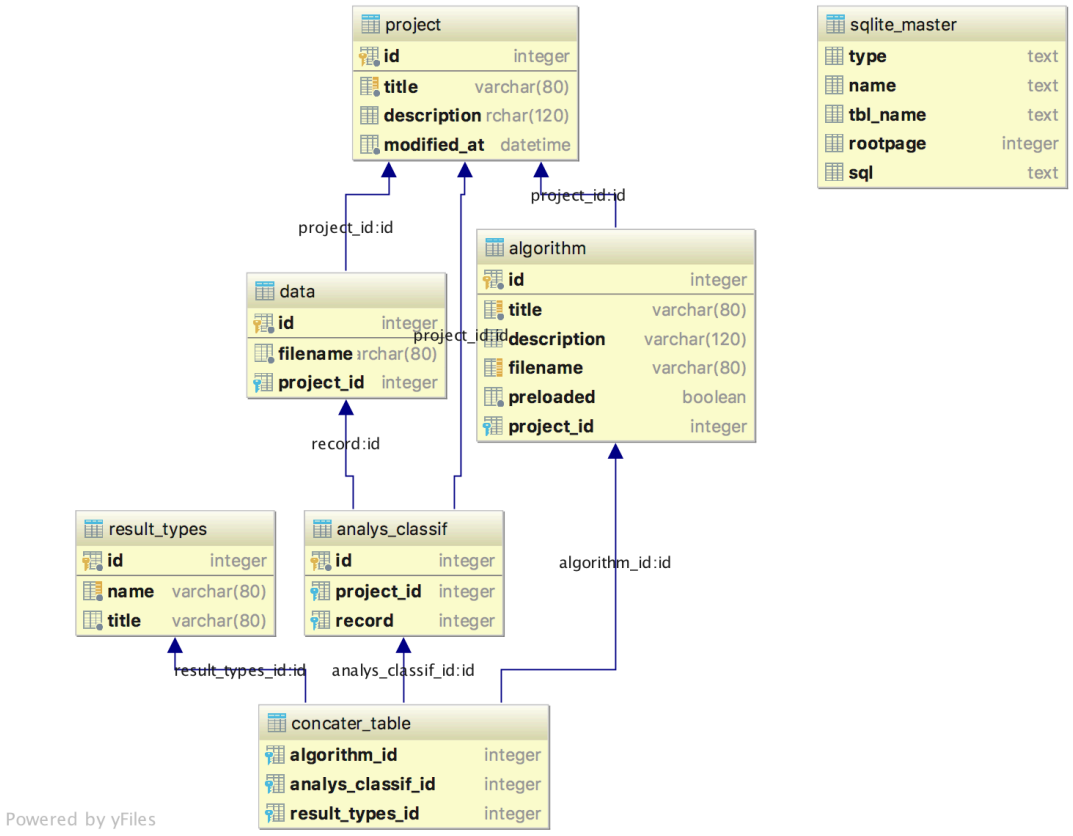

Рис. 1: Структура базы данных

На рисунке 1 представлены связи:

- Многие ко многим (у строки таблицы A может быть несколько совпадающих строк таблицы B).

- Один ко многим (строке таблицы A может сопоставляться несколько строк таблицы B, и наоборот).

Клиент - сервер

Взаимодействие клиента и сервера построена на использовании JSON (специализированный текстовый формат) для передачи данных. Необходимо было согласовать какие поля отвечают за каждый тип передаваемых значений. В итоге получилось простое и надёжное API (интерфейс приложения), позволяющее пользователю гарантированно получить требуемый результат.

Хранение моделей

Необходимо было спроектировать модель хранения обученных моделей, учитывая требование снижения нагрузки на вычислительные ресурсы сервера. Для этого был предложен подход использования отпечатка модели, представляющий собой хеш-сумму от данных и текстового кода алгоритма, благодаря которому не происходит повторного обучения уже ранее обученного алгоритма.

Анализ и результаты классификации

Пользователь может самостоятельно определить какие метрики использовать и какие графики ему необходимо построить. Также в системе уже программно заданы стандартные метрики (метрика) и простейшие графики (график).

5. Программная реализация

При написании кода использовался паттерн MVC (Model-View-Controller). Код хорошо продокументирован, в некоторых местах даны необходимые пояснения в виде комментариев.

Клиентская часть

Основной клиентский код был написан на JavaScript и Angular JS (первая версия).

Серверная часть

Для реализации серверной части были использованы:

- Язык программирования Python

  в ходе разработки технологии, встал выбор основого языка программирования на котором была бы написана система, из всего многобразия языков был выбран этот язык, как достаточно простой для освоения и в то же время достаточно мощный для того, чтобы выдерживать высокие нагрузки (основные библиотеки к языку написаны на C).

- микрофреймворк Flask

  фреймворки нужны для того, чтобы разработчикам программного продукта не приходилось каждый раз заново проектировать и реализовывать низкоуровневую архитектуру и каркас программных модулей. Фреймворк позволяет быстро развернуть веб-приложение, используя уже готовые классы и модули. Микрофреймворк Flask был выбран за его просту и большое сообщество.

- библиотека машинного обучения и анализа данных scikit-learn

  эта библиотека по праву считается основной, когда речь заходит о разработки системы для машинного обучения, так как содержит большое количество готовых реализаций алгоритмов, метрик и многих других полезных компонентов.

- библиотеки scipy, numpy, matplotlib для языка Python

  эти библиотеки представляют собой математические пакеты для языка Python, их исходный код написан на языке C, за счёт чего достигается высокое быстродействие. Использование этих библиотек позволяет производить вычисления в несколько раз быстрее.

- база данных SQLite

  эта база данных компактная, простая в установке и использовании реляционная база данных, позволяющая без лишних временных затрат создать требуемую архитектуру.

- библиотека для языка Python Keras

  библиотека, основанная на TensorFlow и Theano, позволяющая работать с нейросетевыми моделями.

Хранение проектов

При создании проекта, система автоматически создаёт папку с проектом.

Например, создадим проект “дефибриллятор”. Допустим, что в системе не было до этого ни одного проекта. Система создаст следующую структуру папок:

- 1

  ├ algorithms

  ├ data

  ├ models

  ├ results

Структура данных

Технология предполагает возможность загрузки данных пользователем в полуавтоматическом режиме. Принимаются данные в формате

csv (текстовый формат для хранения данных в табличном виде). Разделитель: “,”. Для задач классификации последним признаком должен быть класс.

Пример данных в правильном формате:
0.06372113,0.18939289999999998,0.26748174,0
0.10083995,0.33313184,0.32653288,1
0.045220182000000005,0.1191672,0.16017328,1

Структура алгоритма

Существенным отличием разработанной технологии является то, что она позволяет использовать несколько алгоритмов, которые пользователи могут загружать самостоятельно, написав их на языке Python. Также пользователь может выбрать один из предзагруженных алгоритмов, которые в системе хранятся как common algorithms.

Фактически, алгоритм может быть любым и использовать любые подходы или технологии, если он подходит под требования. Требования к алгоритму:

- Соответствие структуре;
- Написан на языке Python;
- Имеет точку входа и точку выхода (не приводит к зацикливанию).
- Использует стандартные библиотеки или самостоятельно способен импортировать (настраивать сторонние);
- Нацелен на решение задачи классификации или кластеризации;
- Должен успешно запускаться в локальном режиме.

В листинге 1 приведён пример кода, который можно загрузить в систему. Текущий пример иллюстрирует реализацию алгоритма случайного леса (из scikit-learn). Требования к загружаемым алгоритмам: необходимо наличие 3 функций train, test, classify, выполняющих соответствующие задачи. Каких-либо более широких ограничений на загружаемые алгоритмы на накладывается, разработчик может выбрать

любую модель и архитектуру, включая нейронные сети и использовать в своём алгоритме любые библиотеки для языка Python.

Listing 1: Скрипт с алгоритмом

```
from sklearn.ensemble import RandomForestClassifier

def train(X, Y):
    rf = RandomForestClassifier()
    rf.fit(X, Y)

    return rf

def test(model, X, Y):
    metrics = {}
    metrics['mean_accuracy'] = model.score(X, Y)
    plots = None

    return metrics, plots


def classify(model, features_arr):
    res_arr_class = \
    [model.predict(i) for i in feat_arr]
    res_arr_proba = \
    [model.predict(i) for i in feat_arr]
    return res_arr_class, res_arr_proba
```

Система может решать и задачи классификации, и задачи кластеризации, но хотя задачи похожи, специфика самих моделей, метрик и получаемых результатов отличны. Соответственно, в системе предусмотрен выбор между этими двумя типами задач, при выборе которых система автоматически изменит своё поведение.

Основная функциональность системы

Внутреннее устройство системы построено по принципу MVC (Model

View Controller) и MVP (Model Template View). Используется микрофреймворк Flask [6].

Основные функции, обеспечивающие функциональность расположены в файлах views.py и models.py.

processing.py — здесь происходит работа с машинным обучением и анализом данных.

Протоколы передачи данных между компонентами системы

Передача данных между клиентом (javascript) и сервером (python) осуществляется по принципу REST через JSON. Все функции являются асинхронным и неблокирующими, что в свою очередь позволяет обучать несколько моделей одновременно.

Обработка результатов

Обучение модели занимает обычно продолжительное количество времен, что может Поэтому, в работе предложен подход, позволяющий отслеживать какие модели уже были обучены. Для этого применяется отпечаток, посчитанный на основе длины скрипта алгоритма и длины данных. Этот подход является одновременно простым, не накладывая излишних вычислительных нагрузок, и вместе с тем достаточно защищённым от коллизий.

Приведём пример:

algorithm.py - имеет длину 1080 символов.

data.csv - 74150 символов.

Соответственно $74150 * 1080 = 80082000$ — наш отпечаток. Заметим, что изменение алгоритма или данных повлечёт изменение отпечатка и соответственно переобучение модели, а такое поведение как раз и требуется.

Также, в технологии предусмотрен расчёт метрик, построение графиков с выводом на страницу клиента, а также поиск наилучшего алгоритма.

6. Апробация

Для тестирования разработанной технологии выполнена её программная реализация. Вид главной страницы, разработанной системы представлен на рисунке 2.

На рисунке 3 представлена страница, на которой пользователь может создать проект, задав его название, описание. Здесь же можно посмотреть уже созданные проекты, узнать информацию о дате и времени создания проекта.

На рисунке 4 представлена страница загрузки данных в систему. На этой же странице можно получить описание того, какие данные система может принимать и в каком формате. Также можно выбрать проект и узнать какие данные уже были загружены в систему.

На рисунке 5 представлена страница загрузки алгоритмов в систему. На этой же странице можно получить описание того, какие алгоритмы система может принимать и в каком формате.

На рисунке 6 представлен снимок экрана, представляющий предыдущую страницу, тут показано, что система позволяет выбрать проект и вывести информацию о уже предзагруженных алгоритмах, согласно выбранному типу задачи (классификация или кластеризация).

На рисунке 7 представлена страница с обработкой результатов классификации/кластеризации. Здесь выводиться результирующая информация зависимости от типа выбранного результата.

В результате тестирования были усовершенствованы некоторые компонентны системы, добавлено несколько дополнительных программных модулей. В итоге апробация системы показало хорошие результаты, соответственно её можно считать успешной, а технологию пригодной для дальнейшего использования.

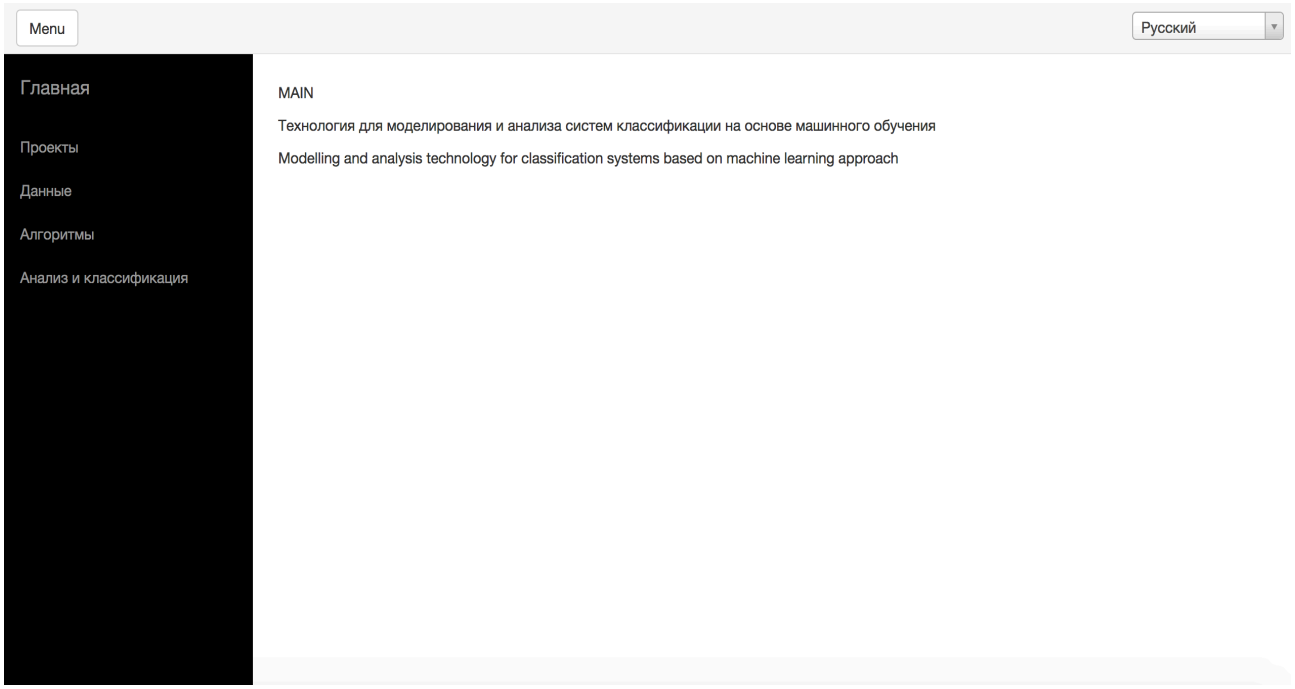

Рис. 2: Главная страница системы

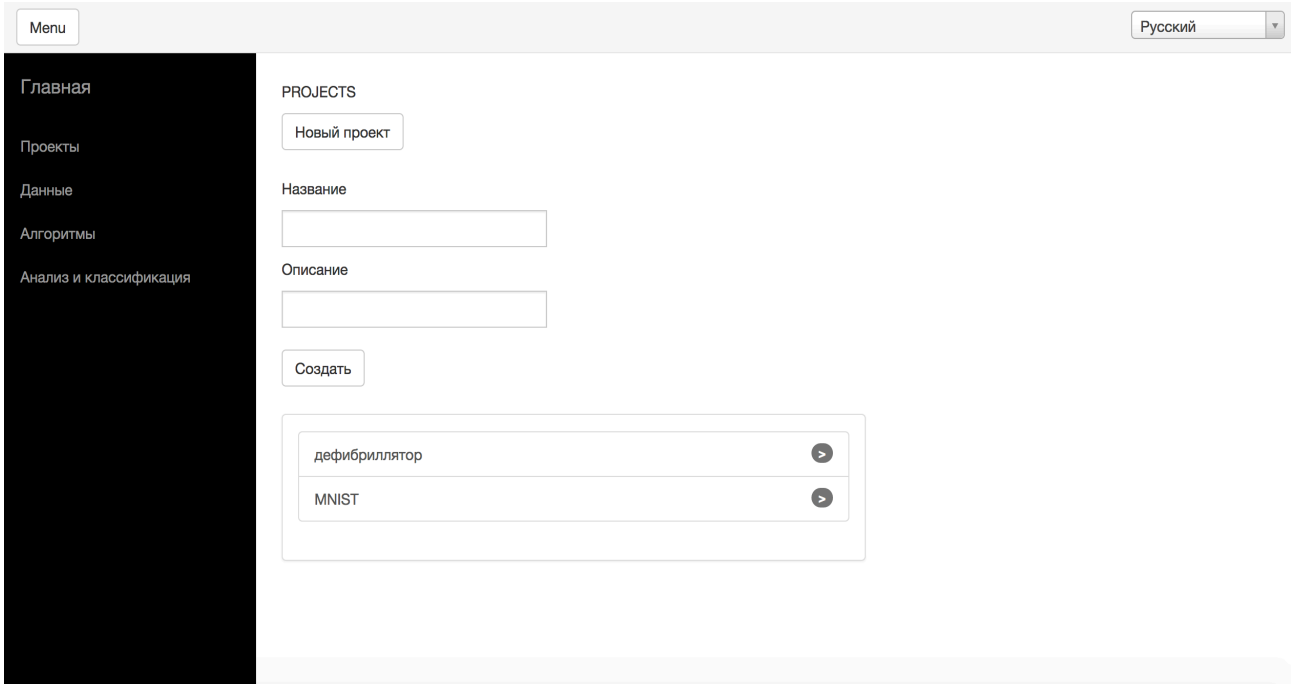

Рис. 3: Создание проекта

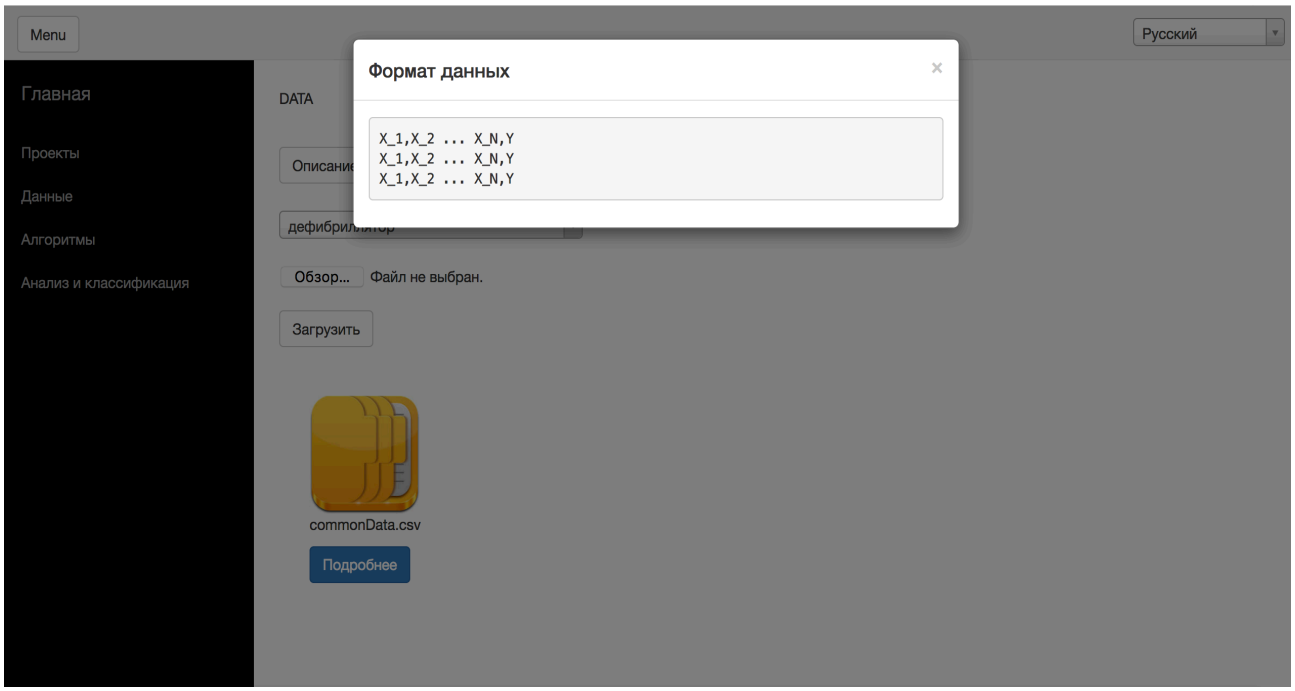

Рис. 4: Загрузка данных

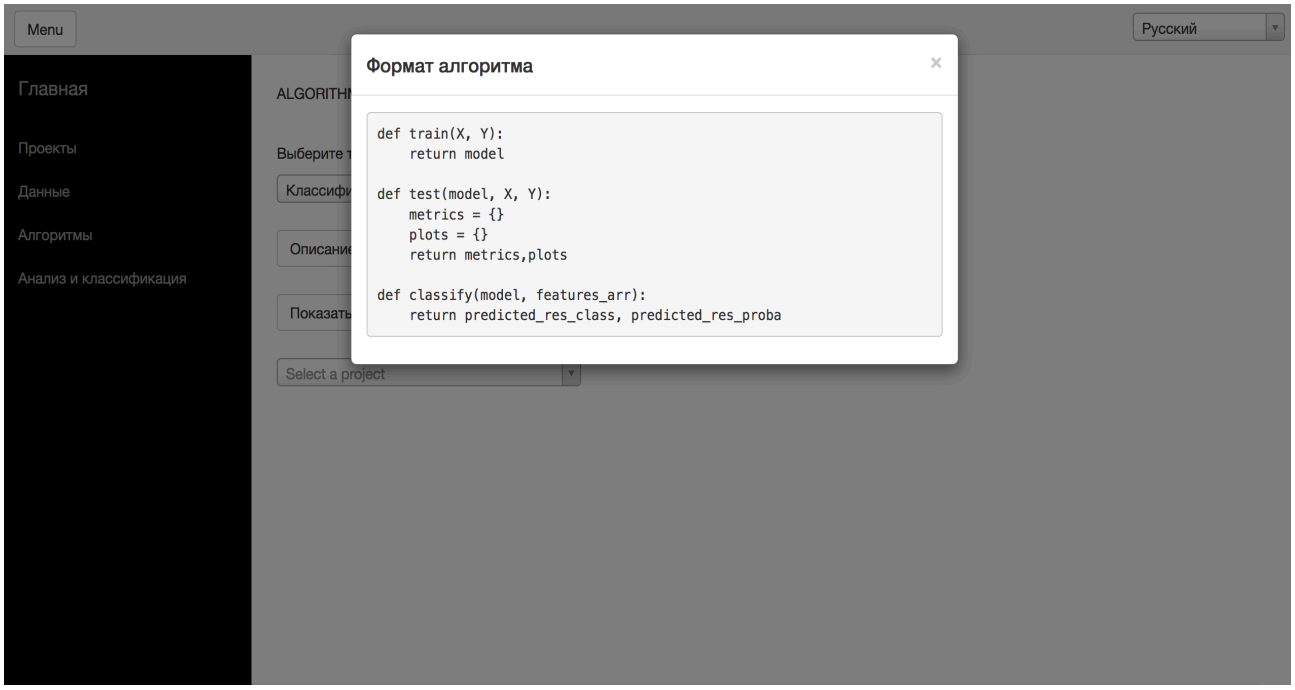

Рис. 5: Загрузка алгоритма

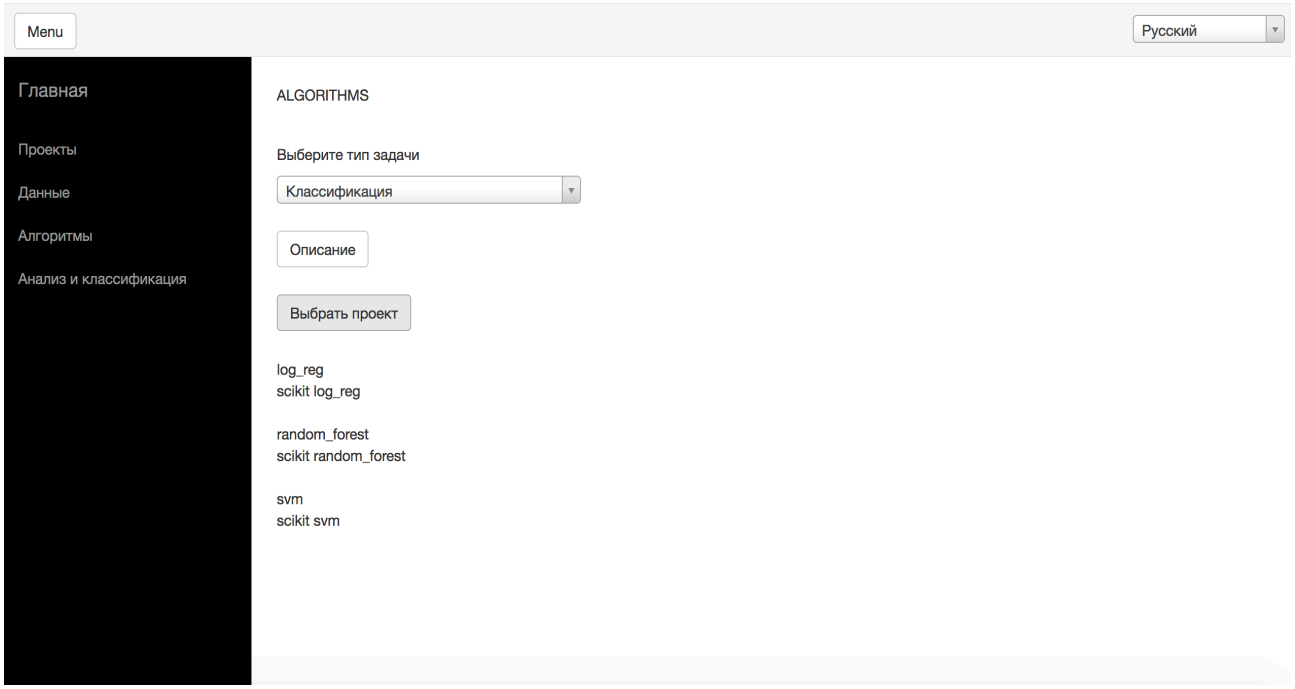

Рис. 6: Алгоритмы

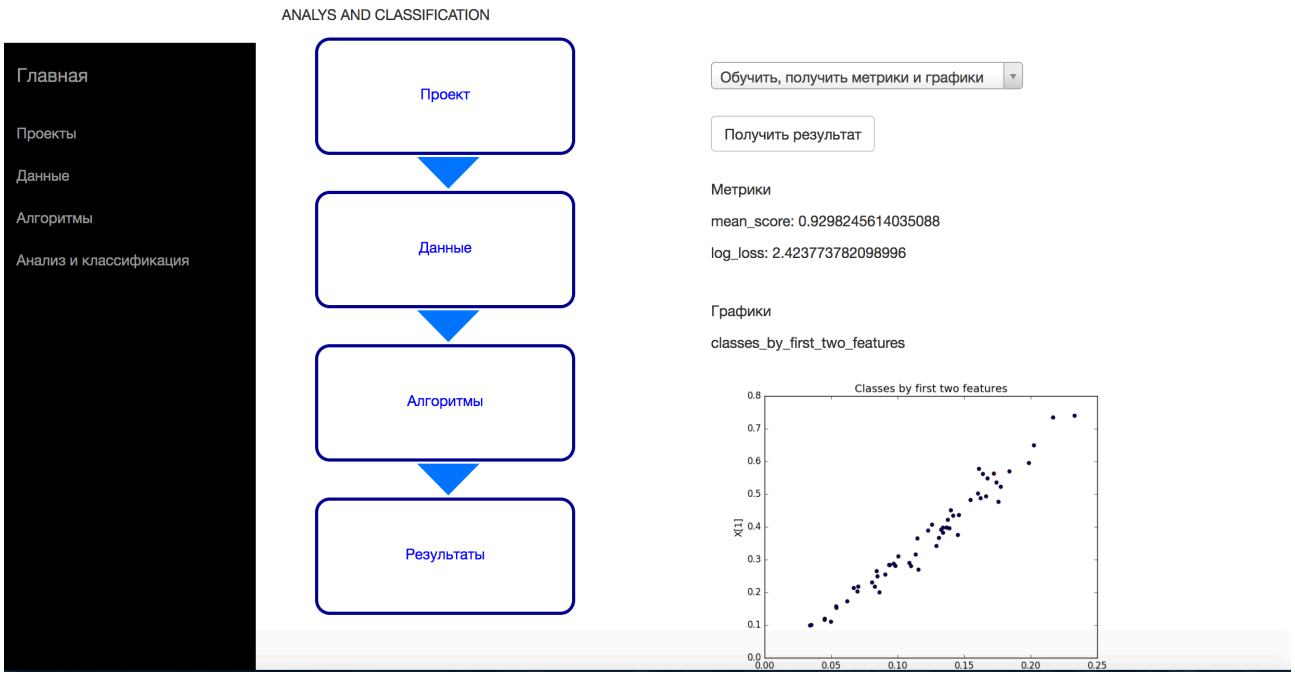

Рис. 7: Страница с результатами

7. Эксперименты

В качестве экспериментов были выбраны несколько задач, с помощью которых были продемонстрированы основные принципы работы с технологией, и был продемонстрирован принцип её использования на реальных задачах.

В диссертационном исследовании были рассмотрены три задачи:

1. Предсказание успешности дефибрилляции;
2. Классификация вин;
3. Кластеризация изображений цифр.

Для того, чтобы продемонстрировать работу технологии были, собраны данные для каждой задачи, подобраны и загружены алгоритмы в систему и осуществлён анализ, а также выполнен сравнительный анализ точности алгоритмов с помощью возможностей системы находить наилучший алгоритм из заданных. Для этого была использованы следующие метрики:

Для классификации:

$$mean\ accuracy = \frac{CORRECT}{ALL}$$

Для кластеризации (см. раздел 3):

$$adjusted\ rand\ score$$

Алгоритмы классификации

Были использованы известные алгоритмы:

1. Random Forest [17];
2. Logistic Regression [17];
3. SVM [17];
4. Random Algorithm (случайный выбор класса).

Нейросетевая модель (классификация)

Помимо указанных выше алгоритмов, в ходе исследования была предложена нейросетевая модель, а именно:

Нейронная сеть - перцептрон.

Со следующими параметрами:

Оптимизатор: adam [4]

Активация (сигмоид)

2 внутренних слоя $(15, 10)$

dropout после каждого скрытого слоя $(0.2, 0.1)$

loss: categorical_crossentropy (binary)

200 epoch, 5 per batch

Нейросетевая одель была реализована при помощи библиотеки Keras.

Алгоритмы кластеризации

1. k-means [17];
2. Mini Batch KMeans [17];
3. Birch [17].

7.1. Задача предсказания успешности применения дефибриллятора

Протестируем работу системы на основе решения задачи прогнозирования успешности дефибрилляции.

Введение в задачу

Основная идея в том, чтобы научиться в режиме реального времени использовать стрим (потоковую передачу данных) с отведений дефибриллятора для того, чтобы в автоматическом режиме предсказывать момент, когда необходимо дать разряд, чтобы остановить угрожающее жизни опасное патологическое состояние — фибрилляцию желудочков.

Это достаточно глобальная задача, рассмотрим в диссертации небольшую подзадачу — предсказать поможет ли дефибриллятор в определенном случае (да - 1, нет - 0). Условно обозначив их — NOROEA, ROEA.

Пояснения используемых терминов даны в разделе 3.

Сбор данных

Были получены данные о характеристиках ритма сердца. После обработки исходных "сырых" данных были получены следующие характеристики сигнала, представленные в формате, который может быть использован для тренировки и валидации моделей.

Всего 19 признаков.

В итоге:

$$FEATURES\_ARRAY - class$$

где class — ROEA(1) или NOROEA(0)

Количество по каждому из классов:
NOROEA = 155 (не помогло)
ROEA = 15 (помогло)
Всего 170 примеров.

Все примеры для обучения были объединены по времени ($0-4+5-9$ секунд до разряда).

Экспериментальная часть работы заключалась в том, чтобы оценить насколько хорошо можно предсказывать успешность дефибрилляции на имеющихся данных.

Результаты

Были проведены эксперименты с этими данными и оценим. Результаты представлены в таблице 4.

7.2. Задача классификации типов вин

Введение в задачу

Задача заключается в том, чтобы на основе химического анализа вина предсказать его сорт (один из трёх). Вина были выращены в Италии.

Сбор данных

Данные были получены из [3]. Они представляют собой набор векторов, размерностью 13. Всего 178 примеров. Три класса вин.

Алгоритм	mean accuracy
Random Forest	0.86
Logistic Regression	0.93
SVM	0.91
Random Algorithm	0.45
Perceptron	0.91

Таблица 4: Метрики оценки точности классификации (задача: дефибриллятор)

Количество по каждому из сортов:

class 1: 59

class 2: 71

class 3: 48

Ниже приведён список того, что каждый из признаков (компонентов вектора) означает.

1. алкоголь;
2. малеиновая кислота;
3. зола;
4. щелочность золы;
5. магний;
6. общее содержание фенолов;
7. флаваноиды;
8. нефлаваноидные фенолы;
9. проантоцианины;
10. интенсивность цвета;
11. оттенок;
12. OD280/OD315 разбавленных вин;
13. пролин.

Характерной чертой этих данных является то, что они не слишком сложны и иллюстрируют проблему “well behaved” [1].

Результаты

Результаты представлены в таблице 5.

Алгоритм	mean accuracy
Random Forest	0.95
Logistic Regression	0.98
SVM	0.42
Random Algorithm	0.34
Perceptron	0.74

Таблица 5: Метрики оценки точности классификации (задача: вина)

7.3. Задача кластеризации на изображениях картинок MNIST

Введение в задачу MNIST ("Mixed National Institute of Standards and Technology") — набор данных "hello world" компьютерного зрения. После выпуска в 1999 году этот классический набор рукописных изображений послужил основой для сравнительного анализа алгоритмов классификации и кластеризации. По мере появления новых технологий обучения, MNIST остается надежным ресурсом для исследователей.

Задача состоит в том, чтобы правильно идентифицировать цифры из набора данных, в котором собраны изображения десятков тысяч рукописных изображений.

Сбор данных

Данные были получены из [13]. Всего выборка была составлена из 4200 примеров для обучения.

Распределение по классам:

0: 425

1: 460

2: 454

3: 386

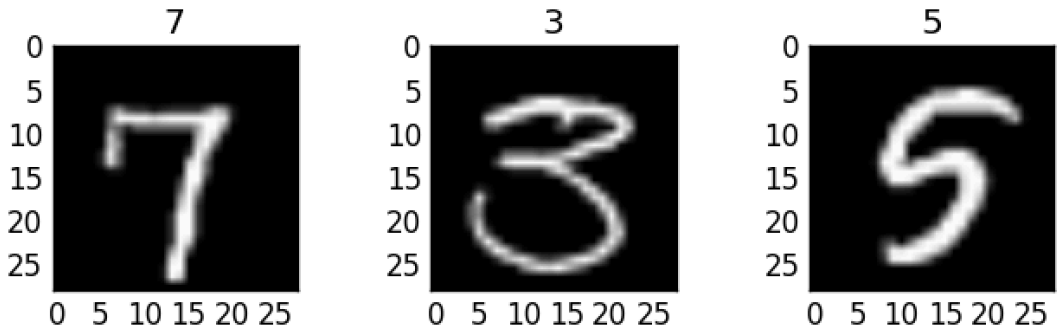

Рис. 8: Пример данных MNIST

4: 409

5: 398

6: 436

7: 427

8: 404

9: 401

Каждый объект, характеризуется вектором размерности 784 (28 на 28 пикселей в изображении).

Результаты

Результаты представлены в таблице 6.

Алгоритм	adjusted rand score
KMeans	0.38
Mini Batch KMeans	0.37
Birch	0.30

Таблица 6: Метрики оценки точности кластеризации (задача: MNIST)

8. Заключение

Основной вывод, который можно сделать на основе исследования, описанного в диссертации, заключается в том, что на сегодняшний момент существует свободная ниша в области программных продуктов и систем, агрегирующих методы и подходы анализа данных и машинного обучения.

В условиях постоянного роста и развития области чувствуется нехватка продуктов, способных решать задачи, задействуя минимальные усилия со стороны дата-саентистов и исследователей.

Также в ходе работы, на основе решения нескольких задач, предложенная технология была протестирована.

В ходе этой работы были получены следующие результаты.

- Разработана архитектура платформы, агрегирующая подходы машинного обучения и анализа данных, спроектирована структура базы данных, спроектировано клиент-серверное взаимодействие. Разработан принцип хранения обученных моделей и полученных результатов классификации;

- Выполнена программная реализация на языке программирования Python, также были использованы языки JavaScript. Программная реализация построена на основе фреймворков Flask и Anguar;

- Система протестирована. Рассмотрены все возможности системы, даны необходимые пояснения, показан образец алгоритма, который может быть загружен в систему;

- Работа системы продемонстрирована на решении трёх задач. В ходе решения этих задач был проведён сравнительный анализ алгоритмов классификации и кластеризации средствами системы, в частности была использована также собственная нейросетевая модель, представляющая собой перцептрон.

Open source

Исходный код, документация и инструкция по развёртыванию системы были выложены в открытый доступ и находятся по адресу `https://github.com/nsmalimov/ml_analysis_ws`.

Также, в качестве примера система была развёрнута на удалённом сервере, доступ к веб-интерфейсу можно получить, перейдя по ссылке:

`http://mlanalysisws.com` — тестовая установка.

Отметим простоту установки, доработки и развёртывания системы, а также поддержку мультиязычности.

Список литературы

[1] Alekh A. Sahand N. Martin J. Fast global convergence of gradient methods for high-dimensional statistical recovery. — 2012. — P. 31.

[2] Amazone Machne Learning. — URL: `https://aws.amazon.com/ru/machine-learning` (online; accessed: 21.05.2017).

[3] Archive ICS. — URL: `https://archive.ics.uci.edu/ml/datasets/wine` (online; accessed: 21.05.2017).

[4] Diederik P. Jimmy L. Adam: a Method For Stochastic Optimization. — 2015. — P. 15.

[5] Dudoit S. Fridlyand J. Leader–follower consensus problems of multi-agent systems with noise perturbation and time delays // Genome Biology. — 2002. — Vol. 7, no. 3. — P. 112–129.

[6] Flask. — URL: `http://flask.pocoo.org` (online; accessed: 21.05.2017).

[7] Fowlkes E. Mallows C. A Method for Comparing Two Hierarchical Clusterings // Journal of the American Statistical Association. — 1983. — no. 78. — P. 553—584.

[8] Fridlyand J. Dudoit S. Application of Resampling Methods to Estimate the Number of Clusters and to Improve the Accuracy of a Clustering Methods. — Stat. Berkely Tech. Report, 2001. — P. 600.

[9] Google Machine Learning. — URL: `https://cloud.google.com/products/machine-learning` (online; accessed: 21.05.2017).

[10] Granichin O. Volkovich Z. Toledano-Kitai D. Randomized Algorithms in Automatic Control and Data Mining. — USA, New York, 2015. — P. 275.

[11] Hubert L. Arabie P. Comparing Partitions // Journal of Classification. — 1985. — P. 193—218.

[12] Jain A. Dubes R. Algorithms for Clustering Data. — Prentice-Hall, Englewood Cliffs, 1988. — P. 320.

[13] Kaggle. — URL: https://www.kaggle.com/c/digit-recognizer (online; accessed: 21.05.2017).

[14] Levine E. Domany E. Resampling method for unsupervised estimation of cluster validity // Neural Computation. — 2001. — no. 13. — P. 2573–2593.

[15] M. William. Objective Criteria for the Evaluation of Clustering Methods // Journal of the American Statistical Association. — 1971. — no. 66. — P. 846–850.

[16] Microsoft Azure. — URL: https://azure.microsoft.com/ru-ru/services/machine-learning (online; accessed: 21.05.2017).

[17] Mohammed J. Wagner M. Randomized Algorithms in Automatic Control and Data Mining. — USA, New York, 2014. — P. 640.

[18] Roth V. Lange T. Braun M. Buhmann J. A Resampling Approach to Cluster Validation // Chinese Physics Letters. — 2012. — P. 123–128.

[19] V. Fomin. Matematical Theory of the Learning Systems of Identifying. — Leningrad University Publisher, Leningrad, 1976. — P. 1312.

[20] V. Vapnik. Statistical Learning Theory. — Wiley, New York, 1968. — P. 768.

[21] Vapnik V. Chervonenkis A. Leader–follower consensus problems of multi-agent systems with noise perturbation and time delays // Soviet Math. Doklady. — 1968. — no. 9. — P. 915–918.

[22] Айзерман М. А. Браверман Э. М. Розоноэр Л. И. Метод потенциальных функций в теории обучения машин. — М.: Наука, 1970. — P. 384.

[23] Граничин О. Н. Поляк Б. Т. Рандомизированные алгоритмы оценивания и оптимизации при почти произвольных помехах. — М.: Наука, 2003. — Р. 191.

[24] Н. Фомин В. Математическая теории обучаемых опознающих систем. — Л.: Изд-во Ленингр. ун-та, 1976. — Р. 236.

[25] Н. Фомин В. Рекуррентное оценивание и адаптивная фильтрация. — М.: Наука, 1984. — Р. 288.