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)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)

WEB-дизайн и мультимедиа
(направленность (профиль)/ специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка программного обеспечения для сравнения эффективности применения различных алгоритмов машинного обучения на данных из открытого репозитория»

Студент Е.И. Долгов
(И.О. Фамилия) (личная подпись)

Руководитель М.А. Тренина
(ученая степень, звание, И.О. Фамилия)

Консультант М.А. Дайнеко
(ученая степень, звание, И.О. Фамилия)

Тольятти 2021

Аннотация

Тема бакалаврской работы: «Разработка программного обеспечение для сравнения эффективности применения различных алгоритмов машинного обучения на данных из открытого репозитория».

Бакалаврская работа посвящена разработке программного обеспечения для автоматизированного сравнения эффективности работы алгоритмов машинного обучения на выбранном наборе данных.

На языке Python разработано программное обеспечение реализующее: загрузку исходного набора данных из файла формата csv, предобработку данных, тестирование на загруженных данных различных алгоритмов машинного обучения, вывод результатов тестирования алгоритмов в виде сравнительной таблицы.

Бакалаврской работа состоит из введения, трёх глав, заключения и списка литературы. Во введении описывается актуальность проводимого исследования, дается краткая характеристика проделанной работы.

В первой главе анализируется проблема подбора алгоритма машинного обучения для систем анализа данных.

Во второй главе описывают особенности использования алгоритмов машинного обучения и тестовый набор данных.

В третьей главе описывается разработанное программное обеспечение для сравнения эффективности алгоритмов машинного обучения. Приводятся результаты тестирования программного обеспечения на выбранном наборе данных.

В заключении представлены выводы по проделанной работе.

В работе присутствуют 1 таблица, 25 рисунков, 8 формул. Список литературы состоит из 22 литературных источников. Общий объем выпускной квалификационной работы составляет 46 страниц.

Abstract

The title of the graduation work is *Developing software to compare the effectiveness of using various machine learning algorithms related to data from an open repository*.

This graduation work is devoted to software development for automated comparison of the machine learning algorithms performance based on a selected dataset, which helps the user to choose a machine learning algorithm to be used for the dataset established.

The key issue of the present graduation work is to identify which algorithm shows the best results based on the analyzed dataset when training a binary classifier. To solve this problem, the following objectives have to be attained:

- to analyze the process of developing software for classifying the data based on the machine learning algorithms;
- to develop an approach to compare the effectiveness of the machine learning algorithms based on the chosen dataset;
- to develop software to conduct binary classification algorithms comparative analyses;
- to test the software on the basis of the medical data from the open repository.

The graduation work consists of an introduction, three chapters, a conclusion and the list of references.

The introduction reveals the relevance of the conducted research and provides a brief description of the work done.

The first chapter of the research analyzes the problem of selecting a machine learning algorithm referred to data analysis systems.

The second chapter of the investigation discusses the peculiarities of using the machine learning algorithms and the test dataset.

The third chapter of the graduation work describes the developed software to compare the performance of the machine learning algorithms and presents the results of testing software based on the selected dataset.

Содержание

Введение

Алгоритмы машинного обучения за последние время все чаще стали использоваться для решения практических задач из различных областей. В медицине, например, по данным химического анализа крови осуществляется классификация заболевания пациента (стоит отметить, что заключительный диагноз ставит врач после подтверждения анализов). В сфере бизнеса с помощью алгоритмов машинного обучения строятся классификаторы для оценки лояльности клиентов (любая компания стремиться сохранить лояльных клиентов).

Однако с развитием машинного обучения появляется все больше алгоритмов для классификации данных, основанных на разных математических принципах. Понятия «алгоритм классификации» и «метод классификации» являются идентичными. Перечислим наиболее известные алгоритмы для получения бинарных классификаторов: Logistic Regression (логистическая регрессия), Ada Boost Classifier (метод AdaBoost), Linear Discriminant Analysis (линейный дискриминантный анализ), Ridge Classifier (классификатор Ridge), Gradient Boosting Classifier (градиентный бустинг), Extra Trees Classifier, Random Forest Classifier (случайный лес), Light Gradient Boosting Machine, K Neighbors Classifier (K ближайших соседей), Decision Tree Classifier (деревья решений), Naïve Bayes (наивный байесовский классификатор), SVM - Linear Kernel (метод опорных векторов), Quadratic Discriminant Analysis (квадратичный дискриминантный анализ).

Проблема заключается в том, что заранее никогда не известно, какой из алгоритмов, покажет лучшие результаты на анализируемом наборе данных при обучении бинарного классификатора. Например, некоторые алгоритмы не могут продемонстрировать приемлемых результатов, если в исходной выборке данных присутствуют выбросы (аномальные значения числовых

признаков). Другие алгоритмы чувствительны к размеру обучающей выборки.

Для преодоления данной проблемы предложено разработать программное обеспечение для тестирования работы всех описанных выше алгоритмов на загруженных пользователем данных. По результатам тестирования программное обеспечение должно определять алгоритм, позволяющий получить наиболее точный классификатор на имеющихся данных.

Таким образом, цель работы – разработка программного обеспечения для сравнения эффективности применения различных алгоритмов машинного обучения на выбранных пользователем данных.

Для достижения поставленной цели в работе решаются следующие задачи:

1. Анализ процесса разработки программного обеспечения для классификации данных, основанного на алгоритмах машинного обучения.
2. Разработка подхода для сравнения эффективности алгоритмов машинного обучения на заданной выборке данных.
3. Разработка программного обеспечения для сравнительного анализа алгоритмов бинарной классификации.
4. Тестирование программного обеспечения на медицинских данных из открытого репозитория.

1 Моделирование процесса разработки систем анализа данных

1.1 Функциональное моделирование

С развитием машинного обучения появляется все больше различных алгоритмов, которые можно применять в составе систем анализа данных [9]. Проведем моделирование процесса разработки систем анализа данных и определим, как можно автоматизировать процесс выбора алгоритма классификации для описания анализируемых данных. Сразу отметим, что в данном исследовании рассматривается только задача бинарной классификации и алгоритмы для их построения (обучения).

Моделирование процесса разработки систем анализа данных необходимо начать с определения контекста, то есть наиболее абстрактного уровня описания системы разработки в целом. В контекст входит определение субъекта моделирования, цели и точки зрения на модель [19]. С точки зрения разработчиков системы анализа данных контекстная диаграмма «Разработка системы анализа данных» описывает разработку наиболее обобщённым образом (рисунок 1).

Информация о предметной области представлена в виде неформализованных описаний. Извлечение экспертных знаний о предметной области не входит в задачу разработки системы анализа данных и поэтому лежит за пределами модели.

Техническое задание предполагается уже утверждённым.

Предполагается, что разработчиками принята некоторая методология, которой они придерживаются при проектировании системы анализа данных. Эта методология включает, в частности, построение функциональной, статической и динамической моделей системы анализа данных. Поскольку методология разработки оказывает управляющее воздействие на все блоки модели, для удобства чтения диаграмм она «убрана в туннель».

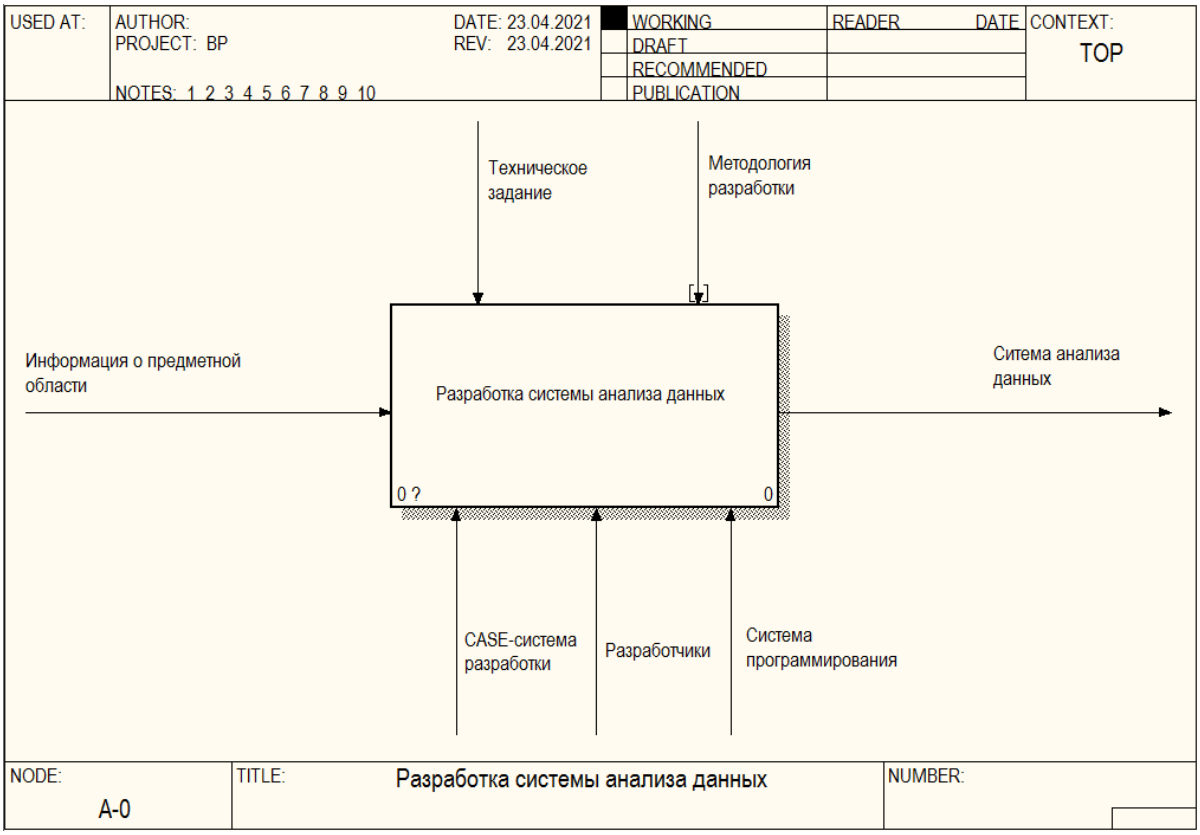

Рисунок 1 – Контекстная диаграмма «Разработка системы анализа данных» для модели «Как есть»

CASE-система – механизм, автоматизирующий разработку системы анализа данных [21]. Диаграмма декомпозиции «Разработка системы анализа данных» описывает разработку системы как итерационный процесс (рисунок 2). Под спецификациями понимаются детальные формальные описания интерфейсов компонентов системы анализа данных.

Модель системы анализа данных подробно описывает функционирование системы, её структуру и взаимодействие компонентов [16].

Понятие «Разработчики» включает в себя аналитиков, программистов и тестировщиков. Аналитики проектируют и тестируют систему анализа данных, программисты занимаются кодированием и отладкой системы. Тестировщики занимаются тестированием системы анализа данных. Среди

тестировщиков допускается наличие представителей заказчика, внедрённых в коллектив разработчиков.

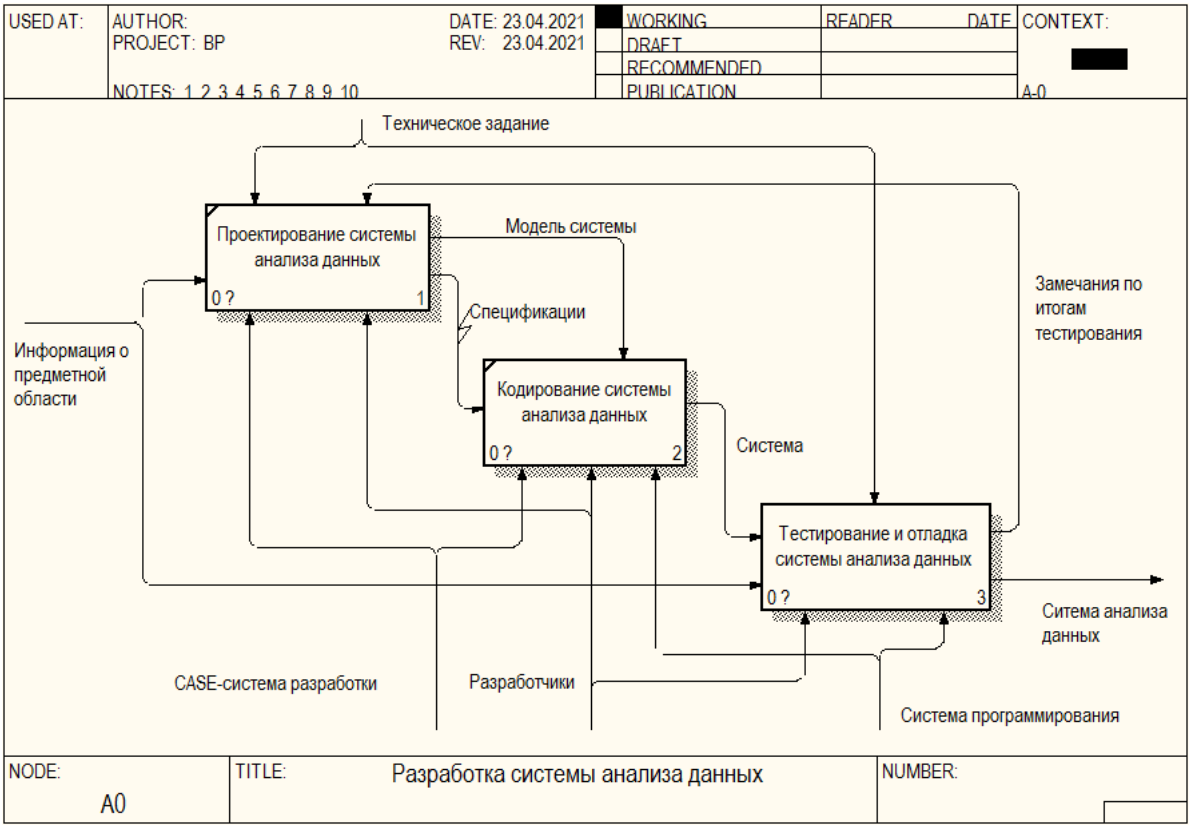

Рисунок 2 – Диаграмма «Разработка системы анализа данных» (A0) для модели «Как есть»

Диаграмма декомпозиции «Тестирование и отладка системы анализа данных» описывает последовательность проведения тестирования системы анализа данных (рисунок 3).

В результате анализа замечаний по итогам тестирования могут быть внесены изменения в проект, касающихся применяемого метода классификации для описания имеющихся данных.

Выбор метода построения классификатора осуществляет разработчик путём перебора всех возможных вариантов до получения приемлемого результата.

В соответствии с моделью «Как должно быть» выбор подходящего метода построения классификатора будет осуществляться с помощью разработанного программного обеспечения. Контекстная диаграмма для модели «Как должно быть» представлена на рисунке 4.

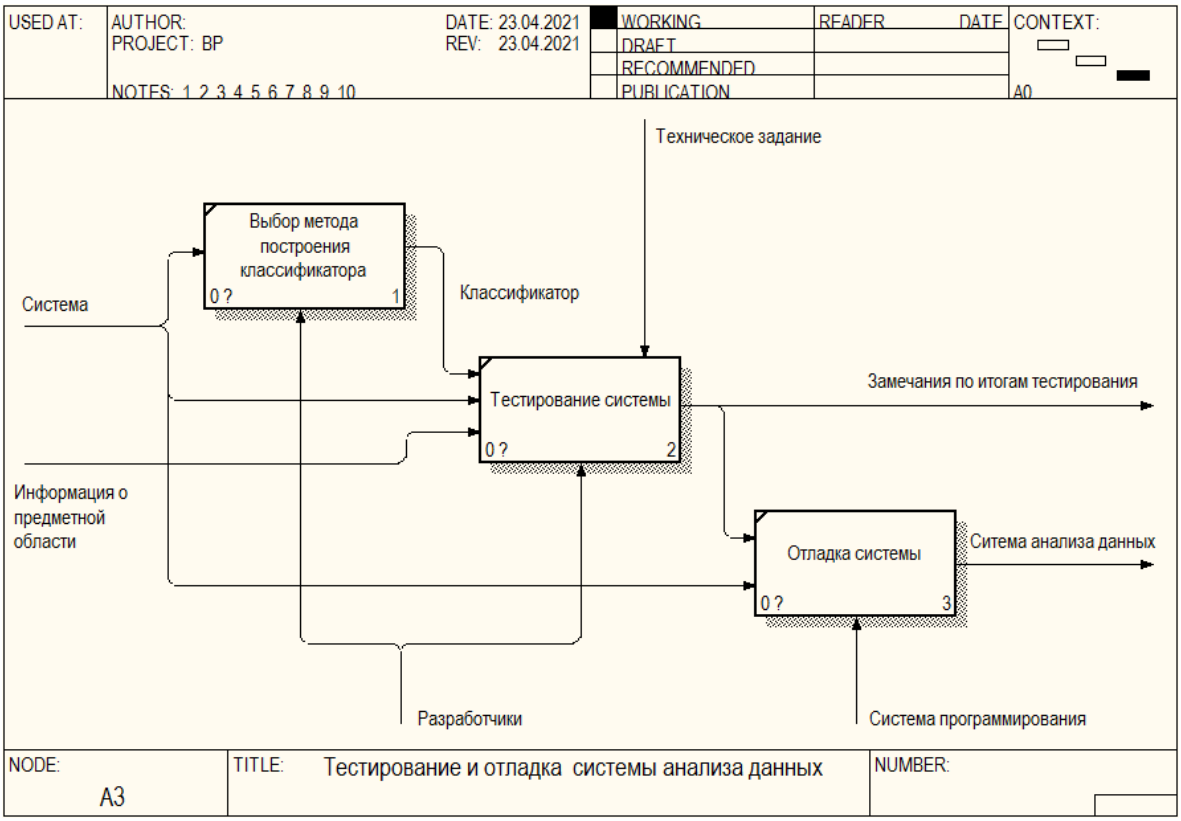

Рисунок 3 – Диаграмма «Тестирование и отладка системы анализа данных» для модели «Как есть»

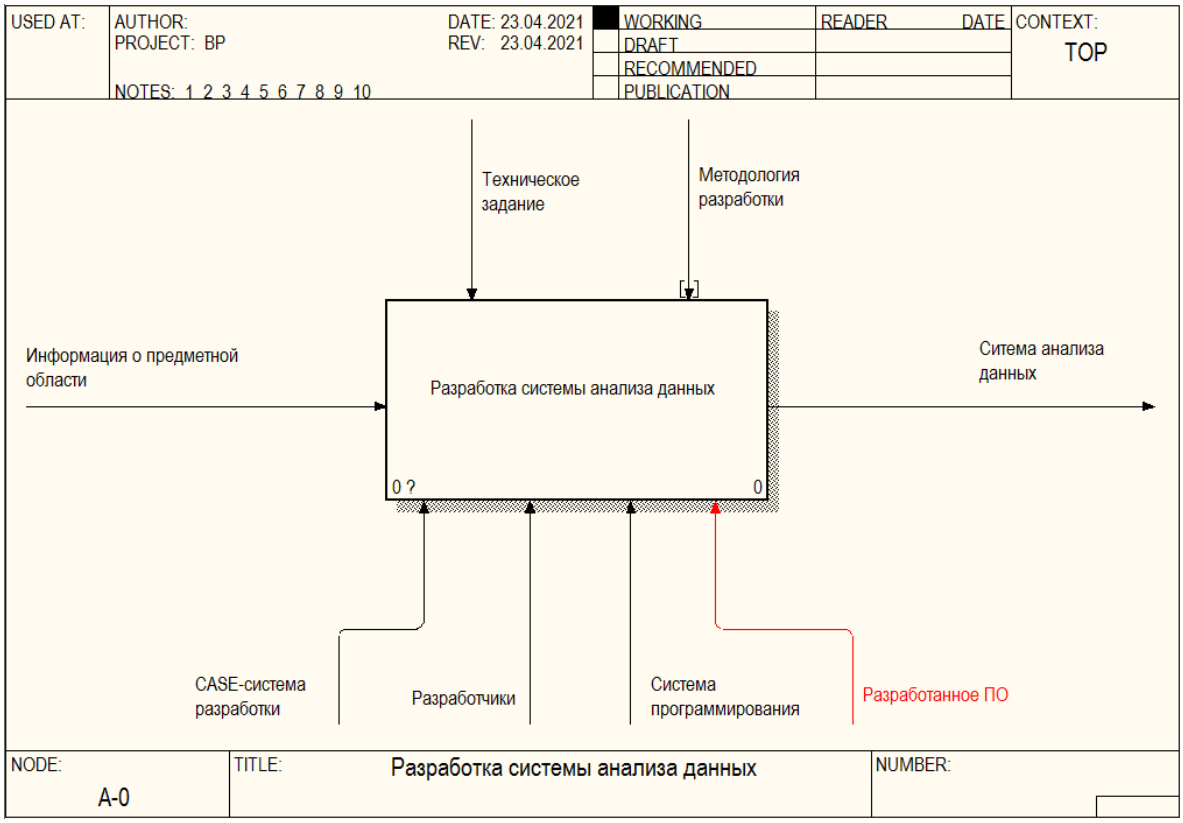

Рисунок 4 – Контекстная диаграмма «Разработка системы компьютерного зрения» для модели «Как должно быть»

Диаграмма декомпозиции «Разработка системы анализа данных» для модели «Как должно быть представлена на рисунке 5.

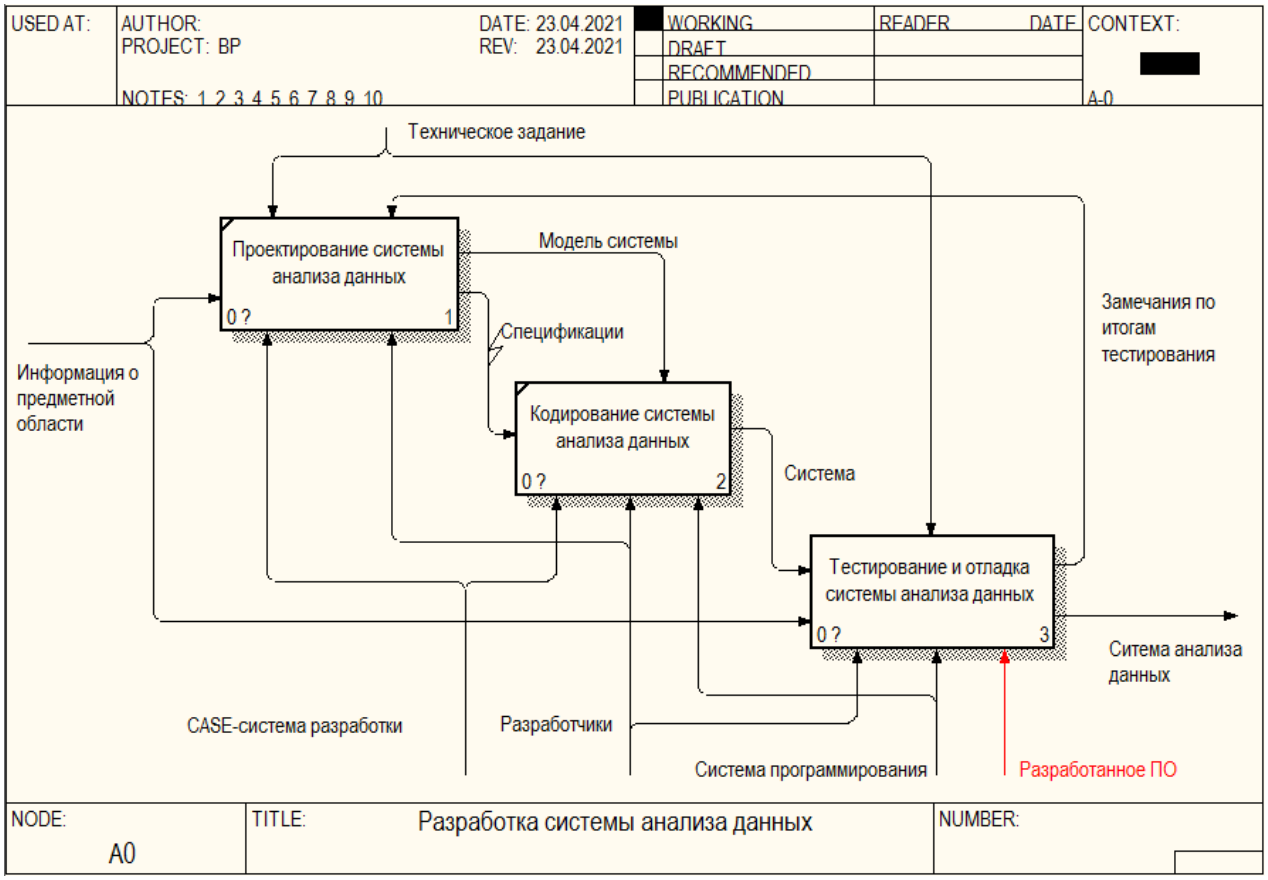

Рисунок 5 – Диаграмма «Разработка системы компьютерного зрения» (A0) для модели «Как должно быть»

На рисунке 6 представлена диаграмма декомпозиции «Тестирование и отладка системы анализа данных» для модели «Как должно быть».

Таким образом, проблему выбора классификатора предлагается решить путем разработки программного обеспечения, осуществляющего поиск лучшего алгоритма классификации под имеющиеся данные (предметную область).

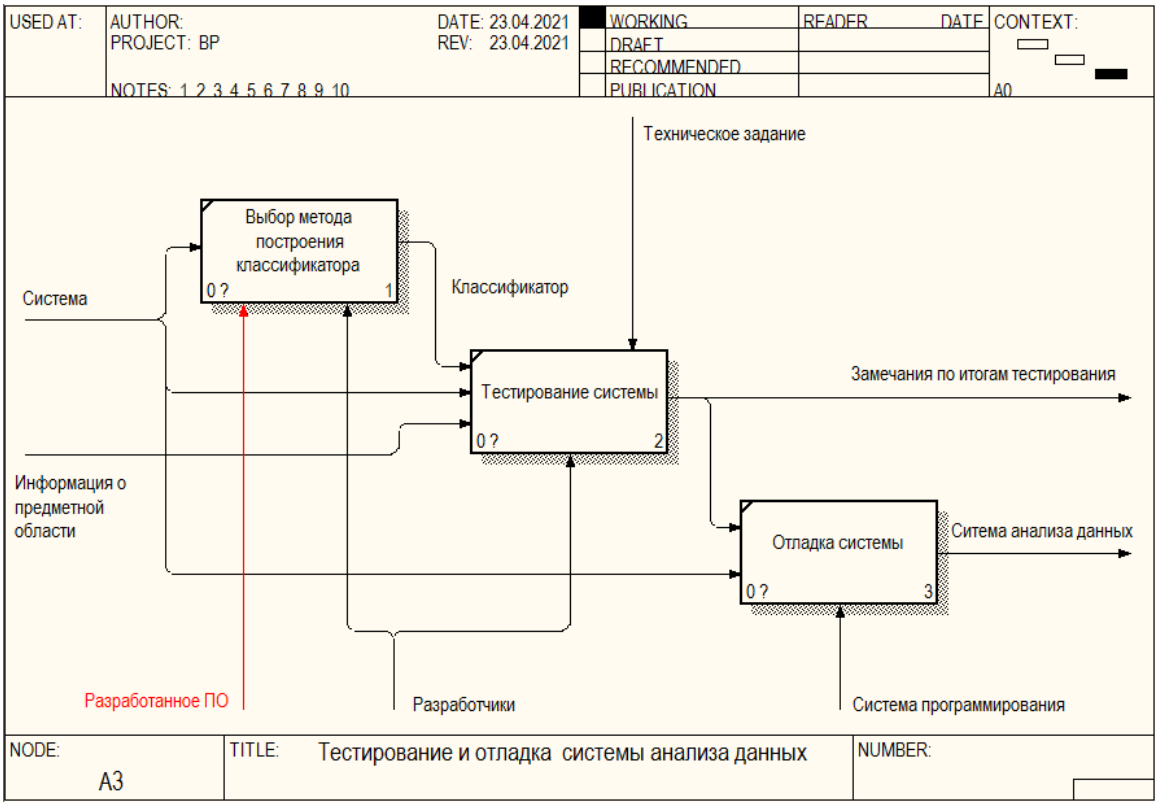

Рисунок 6 – Диаграмма «Тестирование и отладка системы анализа данных» для модели «Как должно быть»

1.2 Выводы по главе

В качестве выводов можно отметить следующее:

1. По результатам материалов, представленных в данной главе, была поставлена задача автоматизации процесса разработки систем анализа данных. Требуется разработать программу для определения лучшего метода классификации для имеющегося набора данных.

2. Были составлены диаграммы модели «Как есть» бизнес-процесса разработки систем анализа данных.

Усовершенствование исследуемого бизнес-процесса достигается путём разработки и внедрения программного обеспечения, которое позволит определять лучший метод классификации для имеющихся данных без участия разработчика.

2 Разработка подхода для сравнения эффективности алгоритмов машинного обучения

2.1 Задача классификации

Опишем задачу классификации. 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 [2].

Классифицировать объект – значит, указать номер (или наименование класса), к которому относится данный объект [1].

В данной работе рассматривается задача двухклассовой (бинарной) непересекающейся классификации.

Пусть X – множество описаний объектов, Y – конечное множество номеров классов. Существует неизвестная целевая зависимость – отображение $y^*: X \to Y$, значения которой известны только на объектах конечной обучающей выборки $X^m = \{(x_1, y_1), ..., (x_m, y_m)\}$ [12].

Требуется построить алгоритм $\alpha: X \to Y$, способный классифицировать произвольный объект $x \in X$.

В данной работе классификация рассматривается на примере задачи медицинской диагностики. В роли объектов выступают пациенты. Признаки характеризуют результаты обследований и симптомы заболевания. Требуется классифицировать вид заболевания (дифференциальная диагностика). В нашем случае класс “0” – отсутствие заболевания, а класс “1” – диабет. Более подробно набор исходных данных описан ниже.

Для решения данной задачи можно использовать один из следующих методов: Logistic Regression (логистическая регрессия), Ada Boost Classifier

(метод AdaBoost), Linear Discriminant Analysis (линейный дискриминантный анализ), Ridge Classifier (классификатор Ridge), Gradient Boosting Classifier (градиентный бустинг), Extra Trees Classifier, Random Forest Classifier (случайный лес), Light Gradient Boosting Machine, K Neighbors Classifier (K ближайших соседей), Decision Tree Classifier (деревья решений), Naïve Bayes (наивный байесовский классификатор), SVM - Linear Kernel (метод опорных векторов), Quadratic Discriminant Analysis (квадратичный дискриминантный анализ). Для того, чтобы понять какой из методов больше подходит для решения задачи классификации на имеющемся наборе данных требуется обучить все типы классификаторов и сравнить их по показателям точности.

2.2 Показатели точности классификаторов

После обучения классификационной модели одним из важнейших вопросов является оценка эффективности полученной модели. Это необходимо для того, чтобы определить, насколько точно модель описывает имеющиеся данные [6].

Рассмотрим, как производится оценка эффективности классификатора с помощью метрик Accuracy, Precision, Recall и F1 Score.

Для того чтобы приступить к определению точностных показателей модели (классификатора) необходимо построить матрицу ошибок (известную также, как Confusion matrix) [13]. Значения данной матрицы рассчитываются на тестовой выборке данных путем наблюдения за работой классификатора. Матрица ошибок для задачи бинарной классификация выглядит так, как это показано на рисунке 2.1. Если в задаче количество классов – 3, то размер матрицы будет 3×3, если 4, то размер матрицы 4×4 и т.д.

Истинные положительные (*TruePositive*) и истинные отрицательные значения (*TrueNegative*) – это наблюдения, которые правильно спрогнозированы и, поэтому, они показаны зеленым цветом. При обучении классификатора требуется минимизировать количество случаев с

неправильно определенными метками класса (значения *FalseNagative* и *FalsePositive*), эти значения показаны красным цветом [17].

В машинном обучении при бинарной классификации класс “0” принято называть Negative, а класс “1” – Positive. Это связано с логикой исходных данных. Поясню на примере. В данной работе мы анализируем медицинские данные пациентов для прогнозирования заболевания диабет. В этих данных под классом “1” находятся пациенты с положительным (Positive) диагнозом, а под классом “0” – с отрицательным (Negative) диагнозом.

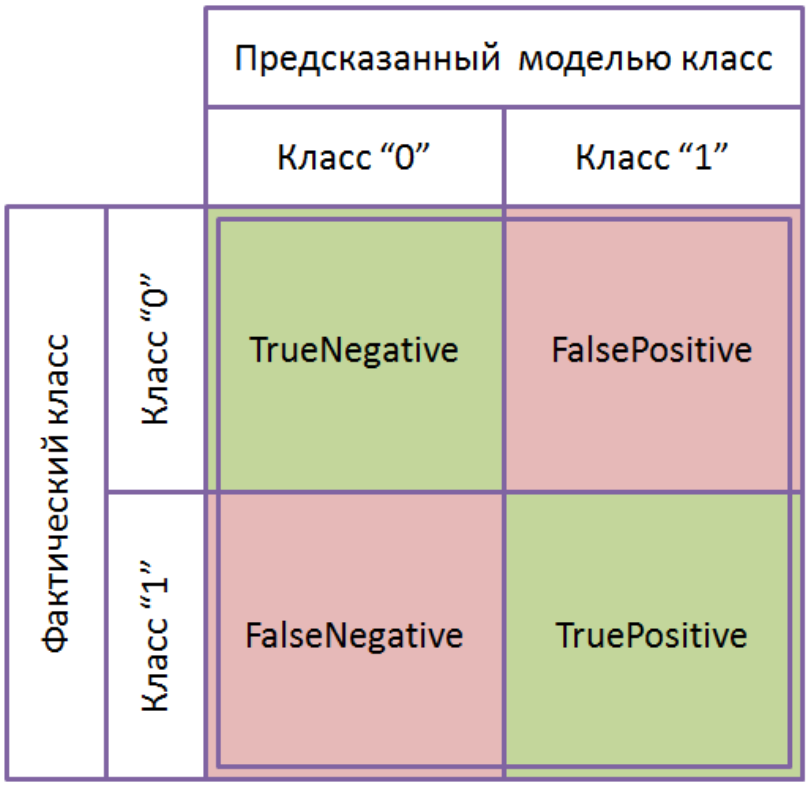

Рисунок 7 – Матрица ошибок для задачи бинарной (с двумя классами) классификации данных

TruePositive – это количество случаев правильно спрогнозированного значения класса “1”. Т.е. это те случаи, в которых фактическое значение класса и значение класса определенное моделью совпадают и равны “1”.

TrueNegative – это количество случаев правильно спрогнозированного значения класса “0”. Т.е. когда значение фактического класса и значение класса, определенного моделью одинаковы и равны “0”.

Значения *FalseNegative* и *FalsePositive* объединяют случаи неправильного срабатывания классификатора данных.

FalseNegative – это количество неправильных срабатываний классификатора, когда при фактическом классе “1” он выдает значение класса “0”. И по аналогии, *FalsePositive* – это количество записей из тестовой выборки данных на которых при фактическом классе “0” классификатор выдал класс “1”.

Как только на тестовой выборке данных определены эти 4 значения (*TruePositives*, *TrueNegatives*, *FalseNegative*, *FalsePositive*) можно переходить к расчёту метрик *Accuracy*, *Precision*, *Recall* и *F1_Score*.

Accuracy является наиболее интуитивно понятной мерой измерения эффективности работы классификатора. Данная метрика представляет собой простое отношение правильно спрогнозированных наблюдений к общему количеству наблюдений (формула 1) [4].

$$Accuracy = \frac{TruePositive + TrueNegative}{TruePositive + FalsePositive + FalseNegative + TrueNegative} \quad (1)$$

Может показаться, что наиболее эффективным является тот классификатор, у которого значение *Accuracy* самое высокой. На самом деле данное утверждение верно только для случаев, когда значения *FalseNegative* и *FalsePositive* практически одинаковы. Поэтому, при сравнении эффективности работы нескольких классификаторов необходимо рассчитывать и другие метрики.

Precision – это отношение правильно спрогнозированных положительных наблюдений (класс “1”) к общему количеству спрогнозированных положительных наблюдений (формула 2) [11].

Применительно к нашему медицинскому набору данных, данная метрика определяет какая доля пациентов, для которых классификатор спрогнозировал диабет, действительно имеют это заболевание.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \quad (2)$$

Recall – это отношение правильно спрогнозированных положительных (класс “1”) наблюдений ко всем наблюдениям в тестовой выборке с фактическим классом “1”. Применительно к нашему медицинскому набору данных, данная метрика отвечает на вопрос – из всех пациентов с заболеванием диабет скольких классификатор смог правильно выявить. Удовлетворительным считаются значения выше 0,5 для данной метрики (формула 3) [15].

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \quad (3)$$

F1_score – это средневзвешенное значение метрик *Recall* и *Precision*. Таким образом, эта оценка учитывает как ложноположительные, так и ложноотрицательные результаты (формула 4) [3].

$$F1\_score = \left(\frac{Recall^{-1} + Precision^{-1}}{2} \right)^{-1} = 2\frac{Recall \cdot Precision}{Recall + Precision} \quad (4)$$

Отметим, что *F1_score* используется в качестве показателя эффективности модели при неравномерном распределении наблюдений по классам. Метрика *Accuracy* используется при приблизительно одинаковых значениях *FalsePositive* и *FalseNegative*, в противном случае необходимо ориентироваться на значения метрик *Precision* и *Recall*.

Коэффициент *Kappa* похожа на метрику Accuracy, за исключением того, что она нормализуется на основе случайных значений на тестовом наборе данных (формула 5) [7]:

$$Kappa = \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_o}{1 - p_e} \quad (5)$$

Этот коэффициент говорит о том, насколько лучше обученный классификатор (p_o) работает по сравнению с классификатором, который просто угадывает случайным образом метку класса на основе частоты каждого класса в тестовой выборке данных (p_e).

Значение коэффициента *Kappa* всегда меньше или равно 1. Значение 0 и меньше указывает на то, что полученный классификатор работает плохо. Чем выше данное значение, тем лучше. Данный коэффициент используется при любом количестве классов и в условиях несбалансированного распределения наблюдений по классам в тестовой выборке данных.

Так же при несбалансированной выборке тестовых данных для оценки моделей классификации применяется Matthews correlation coefficient (*MCC*) (формула 6) [20].

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FP)}} \quad (6)$$

где *FP* – значение *FalsePositive, TP* – значение *FalsePositive, FN* – значение *FalseNegative , FP* – значение *FalsePositive.*

Значение этого коэффициент лежит в диапазоне от -1 до +1. Чем больше значение, тем точнее работает классификатор данных.

.

2.3 Визуализация результатов обучения и работы классификаторов

Разберем способы визуализации результатов работы классификаторов на тестовых данных.

График ROC часто используется для оценки эффективности работы бинарных классификаторов. ROC кривая строится путем фиксирования пары показателей true positive rate (*TPR*) и false positive rate (*FPR*) для классификатора при различных значениях порога (*threshold*) [3]. Например, в классификаторе, основанном на логистической регрессии, если предсказано, что наблюдение принадлежит к первому классу с вероятностью > 0.5, то прогнозируемый класс обозначается как “1”, в противном случае прогнозируется класс “0”. Тем не менее, мы можем выбрать любой порог (*threshold*) вероятности в диапазоне от 0 до 1, а не только значение равное 0.5. График ROC помогает визуально оценить, как выбор порога влияет на работу классификатора.

Значения *TPR* и *FPR* рассчитываются по следующим формулам [22]:

$$TPR = Sensitivity = \frac{TP}{TP + FN}, \tag{7}$$

$$FPR = 1 - Specificity = \frac{FP}{FP + TN} \tag{8}$$

На рисунке 8 показано, как будет выглядеть графики ROC для различных классификаторов. Серая пунктирная линия представляет собой классификатор, который на выходе выдает случайную метку класса. Фиолетовая линия представляет собой идеальный классификатор, который никогда не ошибается при любом пороговом значении. Графики почти всех реальных классификаторов будут находиться между линиями идеального классификатора и линией классификатора со случайным угадыванием класса.

Чем сильнее линия ROC классификатора приближается к левому верхнему углу графика, тем он лучше.

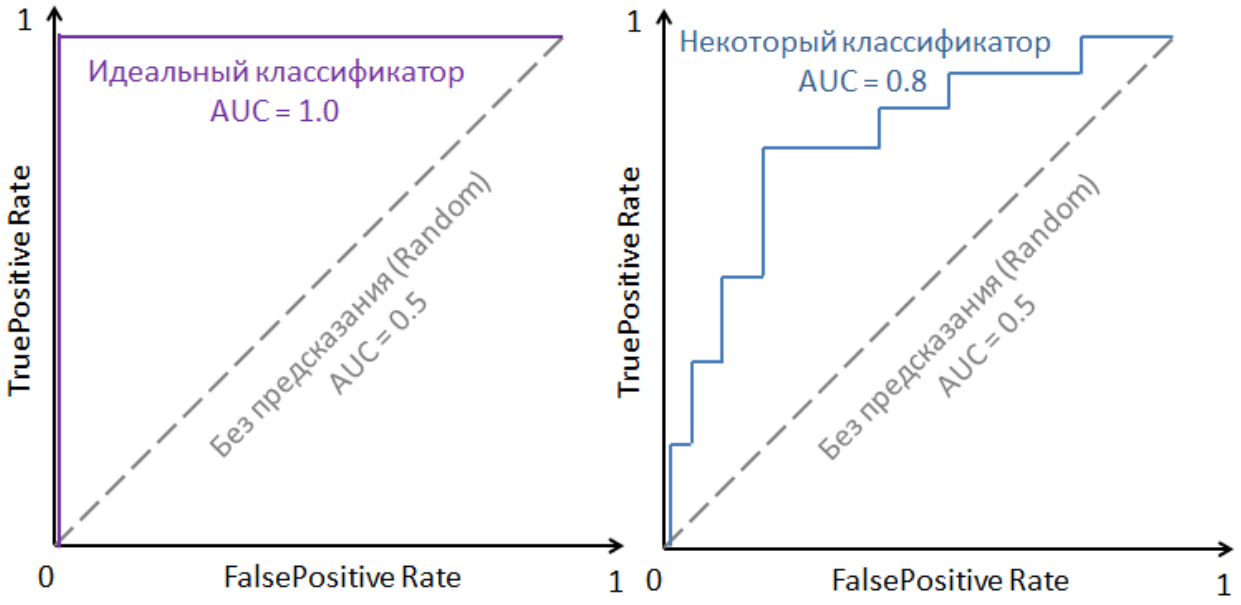

Рисунок 8 – Графики ROC (receiver operating characteristic): слева – для идеального классификатора, справа – для некоторого существующего классификатора

Так же, как и график ROC, график Precision-Recall используется для оценки работы бинарного классификатора [13]. Он часто используется в ситуациях, когда классы сильно несбалансированы. Также как и график ROC, график Precision-Recall обеспечивает графическое представление эффективности классификатора по нескольким значения (в отличие от одиночных параметров *Accuracy, Precision, F1_score, Kappa, MCC*).

График Precision-Recall строится путем расчета значений Precision и Recall для классификатора при различных пороговых значениях [5]. Например, если мы используем в качестве классификатора логистическую регрессию, то порогом будет являться прогнозируемая вероятность наблюдения, принадлежащего к первому (*positive*) классу. График Precision-

Recall помогает визуализировать, как выбор порога вероятности (для отнесения объекта к классу “1”) влияет на эффективность работы классификатора, и даже может помочь нам выбрать лучший порог для конкретного набора данных.

На рисунке 9 показано, как будут выглядеть графики Precision-Recall для теоретических классификаторов. Серая пунктирная линия “Baseline” представляет собой график Precision-Recall который всегда предсказывает принадлежность объекта к первому классу. Положение этой пунктирной линии зависит от доли классов в обучающей выборке. Фиолетовая линия представляет собой идеальный классификатор - классификатор с идеальной точностью при любых значениях порога. Графики реально существующих классификаторов будут всегда находиться где-то между этими двумя линиями (фиолетовой и серой). Хороший классификатор будет поддерживать и высокое значение Precision и высокое значение Recall по всему графику, и будет огибать верхний правый угол системы координат Precision-Recall.

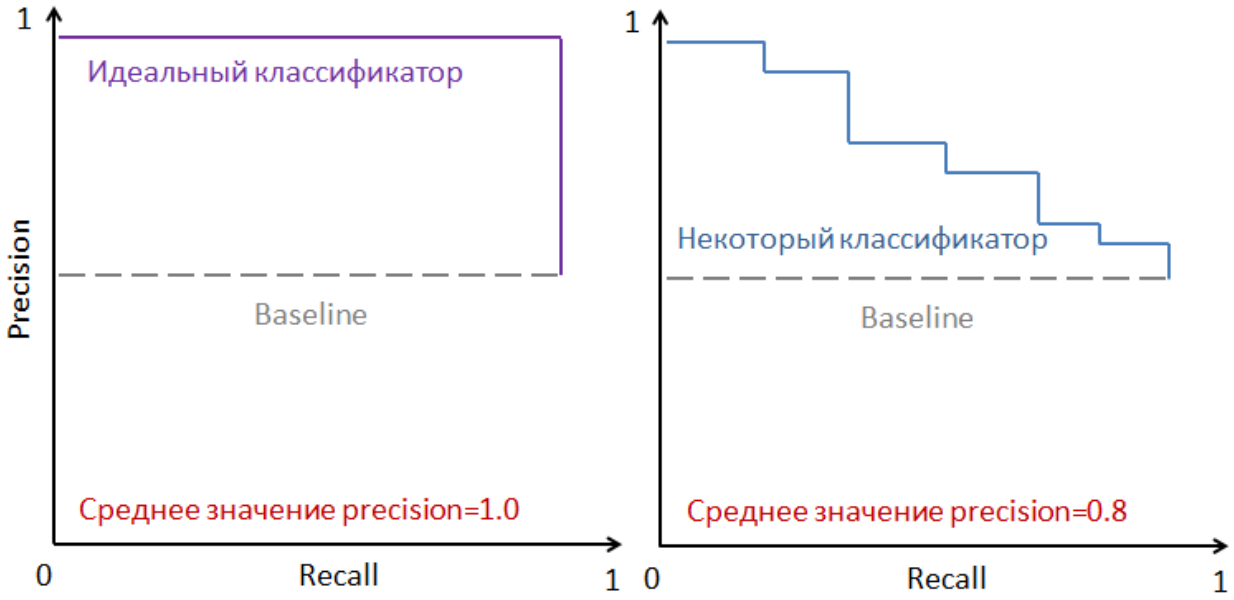

Рисунок 9 – Графики Precision-Recall: слева – для идеального классификатора, справа – для некоторого существующего классификатора

Так же, как и для графика ROC, мы можем просуммировать площадь под графиком для получения дополнительного значения – *AUC-PR* (area under curve precision recall). Эта сводная метрика, которая характеризует работу классификатора на имеющихся тестовых данных - чем выше значение *AUC-PR*, тем лучше классификатор подходит под тестовые данные [18].

Одним из способов использование *AUC-PR* является нахождение параметра *AP* – средневзвешенного значение precision по всем порогам [8].

В идеальном классификаторе *AUC-PR* = 1. В "базовом" классификаторе, *AUC-PR* будет зависеть от доли наблюдений, принадлежащих к классу “1”. Например, в сбалансированном наборе данных (с равными долями наблюдений относящихся к классам “0” и “1”) “базовый” классификатор будет иметь *AUC-PR* = 0,5.

Так как расчет точностных параметров классификатора осуществляется на основе значений матрицы ошибок, то необходимо также обеспечить визуализацию данной матрицы. При визуализации, для удобства восприятия, ячейки таблицы окрашиваются в зависимости от значений компонентов матрицы ошибок [14].

На рисунке 10 представлены 2 примера визуализации матрицы ошибок. На рисунке слева показан пример матрицы, описывающей работу идеального классификатора данных, работающего без ошибок (тестовая выборка в данном случае состоит из 231 записи). У идеального классификатора правая верхняя и нижняя левая ячейки окрашены белым цветом из-за отсутствия случаев неправильно определенных классов. С правой стороны на рисунке 10 показана визуализация матрицы ошибок для некоторого реального классификатора, совершившего 59 ошибок при работе на тестовой выборке данных.

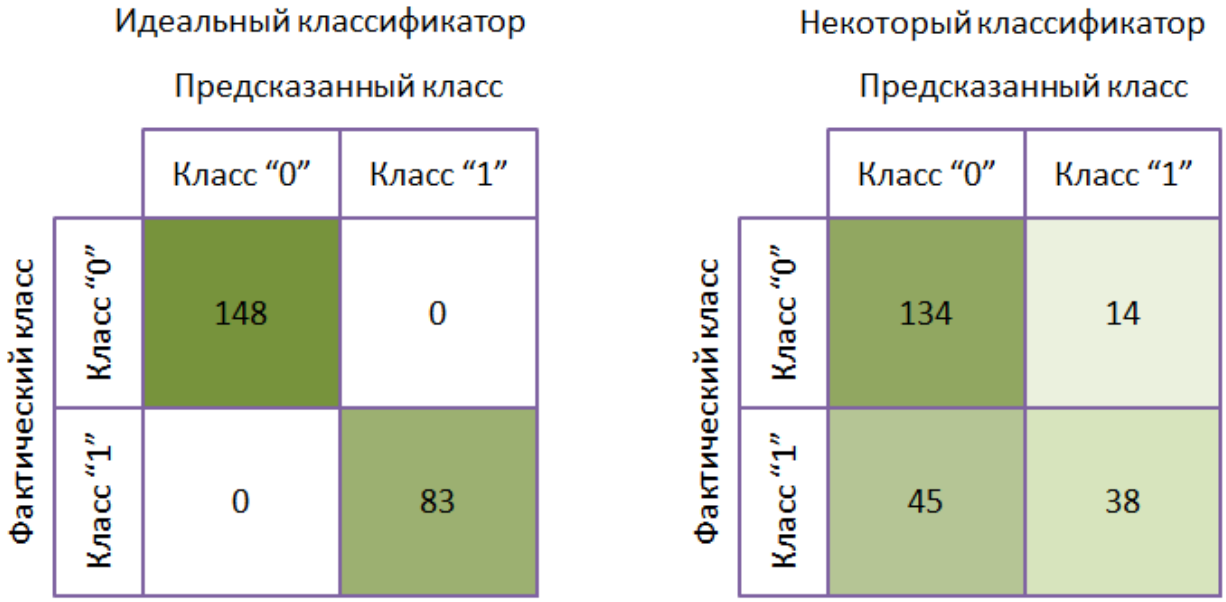

Рисунок 10 – Визуализация матрицы ошибок на тестовой выборке данных: слева – для идеального классификатора, справа – для некоторого существующего классификатора

С точки зрения количественной оценки ошибок, совершенных классификатором на тестовой выборке данных более наглядной, является столбчатая диаграмма результатов определения классов (рисунок 11). По оси X расположены классы, определенные классификатором, а цветом показаны фактические значения классов (указанные в тестовой выборке данных). На рисунке 11 для сравнения приведены два случая столбчатых диаграмм: слева – вид диаграммы при работе идеального классификатора, не совершающего ошибок, справа – вид диаграммы для некоторого реального классификатора. Благодаря данной столбчатой диаграмме можно по цвету визуально оценить эффективность работы классификатора.

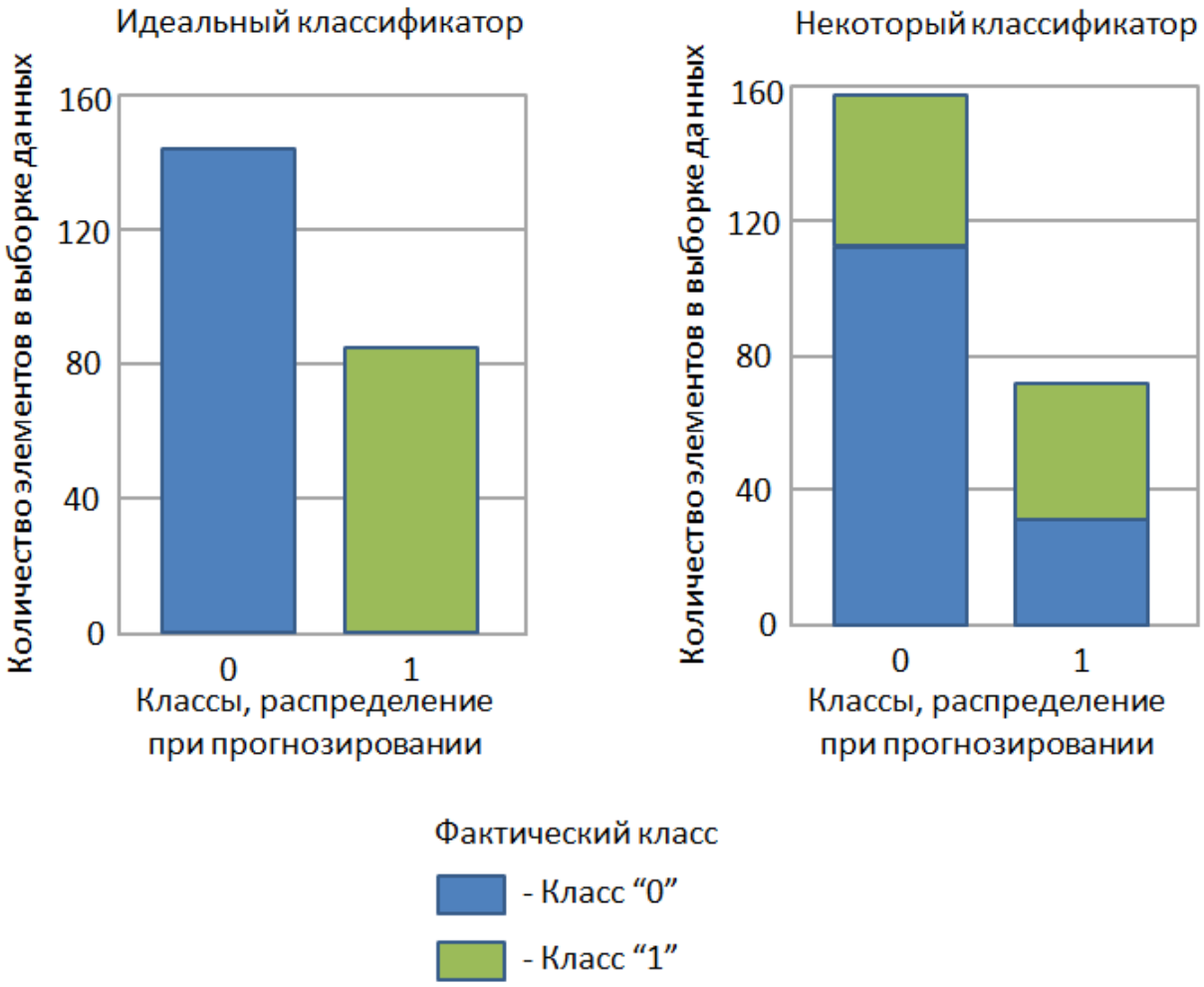

Рисунок 11 – Столбчатая диаграмма для визуализации ошибок классификации на тестовом наборе данных: слева – для идеального классификатора, справа – для некоторого существующего классификатора

Объединив точностные показатели работы классификатора на тестовой выборке данных можно построить и визуализировать матрицу точности. Примеры визуализации точностных показателей классификаторов показаны на рисунке 12.

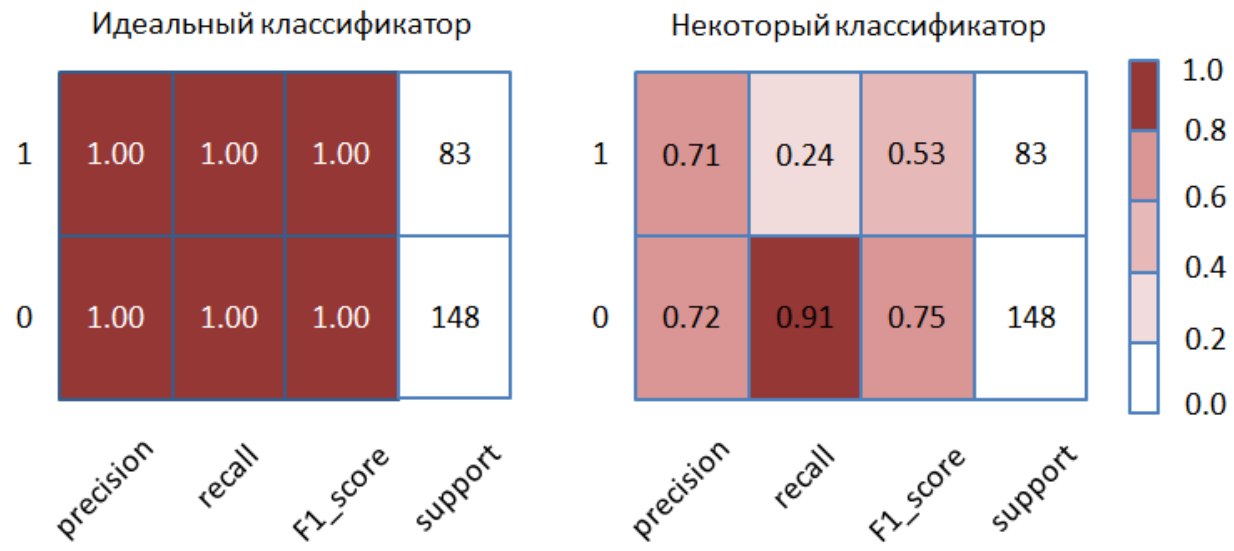

Рисунок 12 – Визуализация матрицы точностных показателей классификатора: слева – для идеального классификатора, справа – для некоторого существующего классификатора

На рисунке слева показан вид матрицы для идеального классификатора, а справа для некоторого существующего классификатора, при работе которого возникали ошибки. Для удобства визуального восприятия вводится цветовая школа, на основе которой окрашиваются ячейки таблицы со значениями.

2.3 Планирование вычислительных экспериментов

Эффективность работы классификатора оценивается на тестовом наборе данных, а его настройка проводится на тренировочных данных. Таким образом, исходный набор данных перед началом обучения классификатора необходимо поделить на два непересекающихся множества (тестовый набор данных и тренировочный набор данных) [10]. Обычно, доля тестового набора данных составляет 10% от исходного набора данных. Отбор записей в тестовую выборку данных, обычно, осуществляется случайным образом. В

этом случае, результат определения точностных параметров классификатора зависит от того, какие из записей попали в тестовую выборку данных.

Для того, чтобы сделать точностные показатели более объективными предложено следующее решение. Изначально исходная выборка данных делится случайным образом на 10 непересекающихся множеств приблизительно равного размера. На основе этих 10 множеств проводится 10 вычислительных экспериментов, в которых по очереди 1 множество выступает в роли тестовой выборки данных, а остальные множества в роли тренировочного набора данных.

В результате каждого из десяти вычислительных экспериментов определяются точностные показатели (*Accuracy*, *Recall*, *Precision*, *F1_score*, *Kappa*). Конечным результатом для исследуемого типа классификатора является определение средних значений этих показателей (рисунок 13).

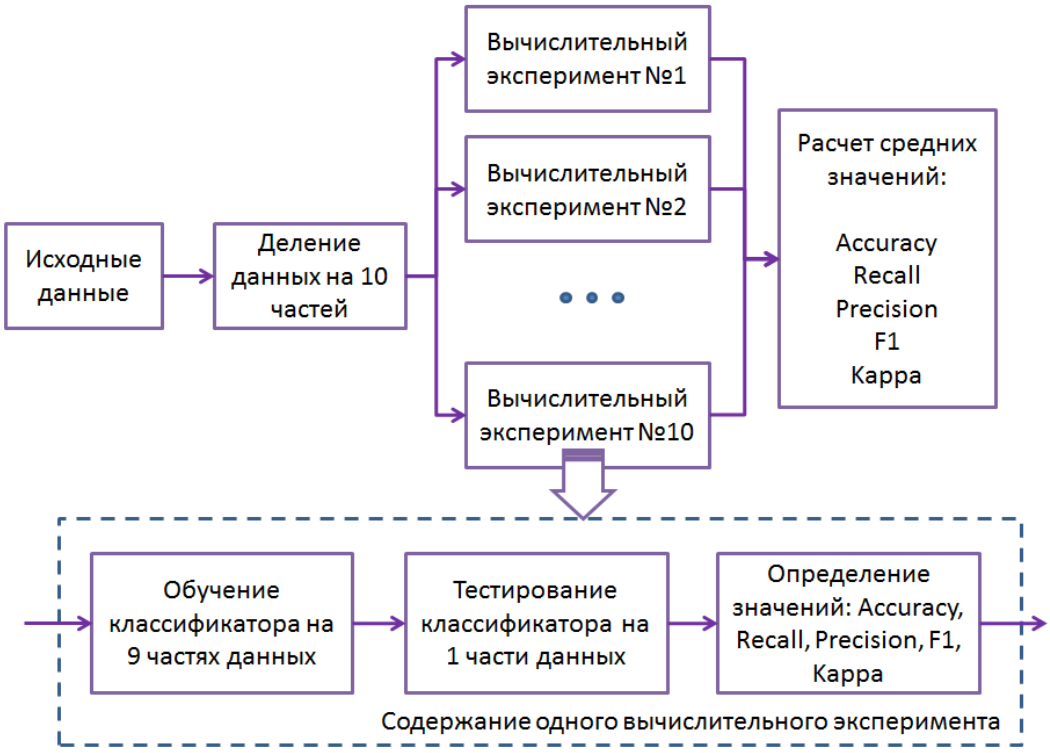

Рисунок 13 – Схема вычислительных экспериментов для одного типа классификаторов (10 вычислительных экспериментов на каждый тип классификатора)

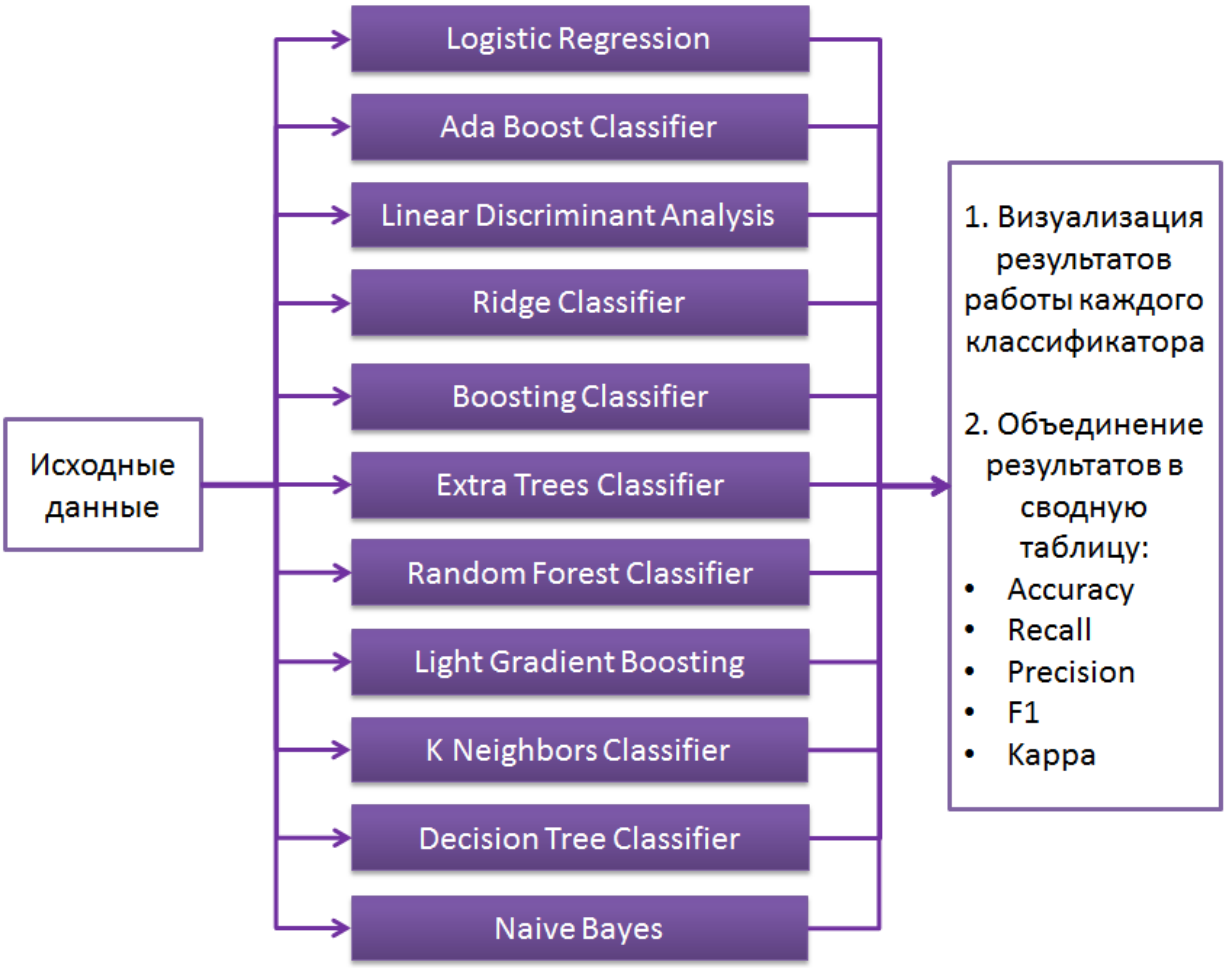

Рисунок 14 – Схема тестирования различных типов классификатора и объединение результатов в сводную таблицу

Для того, чтобы сформировать сводную с точностными показателями необходимо повторить вычислительные эксперименты для всех типов классификаторов (рисунок 14).

2.4 Набор данных для тестирования предложенного подхода

Набора данных, на котором будет проводиться тестирование разных методов построения классификатора, имеет 8 числовых признаков. Описание признаков представлено в таблице 1. Источник данных – общедоступный репозиторий данных (github). Имеющийся набор данных представлен в

обезличенном виде, для его использования не требуется получение специальных разрешений.

Таблица 1 – Описание признаков в выборке данных

Название признака	**Описание**	**Тип признака**
Number of times pregnant	Количество беременностей	числовой
Plasma glucose concentration a 2 hours in an oral glucose tolerance test	Концентрация глюкозы в сыворотке крови через 2 часа при оральном глюкозотолерантном тесте	числовой
Diastolic blood pressure (mm Hg)	Диастолическое артериальное давление (мм рт.ст.)	числовой
Triceps skin fold thickness (mm)	Толщина кожной складки над трицепсом (мм)	числовой
2-Hour serum insulin (mu U/ml)	2-часовой сывороточный инсулин	числовой
Body mass index (weight in kg/(height in m)^2)	Индекс массы тела (вес в кг/(рост в метрах)^2)	числовой
Diabetes pedigree function	Функция, значение которой зависит от наличия диабета у родственников	числовой
Age (years)	Возраст (годы)	числовой
Class variable	Метка класса	категориальный

В представленном наборе данных информация о 769 пациентах с подозрением на диабет. К классу “1” относятся пациенты с подтвержденным

диабетом, а к классу “0” относятся пациенты с подтвержденным отсутствием диабета.

Имеющаяся выборка данных представлена в виде csv файла.

2.5 Выводы по главе

Описан подход для определения лучшего метода построения классификатора для выбранных исходных данных. Данный подход включает в себя: разделение исходных данных на 10 приблизительно равных не пересекающихся множеств; проведение для каждого метода построения классификатора 10 вычислительных экспериментов, в которых по очереди 1 множество выступает в роли тестовой выборки данных, а остальные множества в роли тренировочного набора данных; расчёт на основе вычислительных экспериментов для каждого метода средних показателей точности (*Accuracy*, *Recall*, *Precision*, *F1_score*, *Kappa*), которые объединяются в сводную таблицу.

Для упрощения сравнения эффективности различных классификаторов предложено использование следующих средств визуализации: график ROC, график Precision-Recall, визуальная интерпретация матрицы ошибок, столбчатая диаграмма ошибок классификации и визуальная интерпретация матрицы точностных показателей классификатора.

Теперь необходимо реализовать предложенные идеи в виде программного обеспечения.

3 Разработанное программное обеспечение для сравнения методов построения классификаторов

3.1 Особенности разработанного программного обеспечений

При выполнении исследований, представленных в бакалаврской работе, на языке Python в среде программирования Google Colab было спроектировано, разработано и протестировано программное обеспечение для определения лучшего классификатора для заданных пользователем данных. При этом программное обеспечение обладает следующими особенностями:

- наличие графического интерфейса, реализованного посредством виджетов;
- наличие возможности работы с данными представленными в виде csv-файла;
- поддержка следующих методов для построения бинарных классификаторов: Logistic Regression (логистическая регрессия), Ada Boost Classifier (метод AdaBoost), Linear Discriminant Analysis (линейный дискриминантный анализ), Ridge Classifier (классификатор Ridge), Gradient Boosting Classifier (градиентный бустинг), Extra Trees Classifier, Random Forest Classifier (случайный лес), Light Gradient Boosting Machine, K Neighbors Classifier (K ближайших соседей), Decision Tree Classifier (деревья решений), Naïve Bayes (наивный байесовский классификатор), SVM - Linear Kernel (метод опорных векторов), Quadratic Discriminant Analysis (квадратичный дискриминантный анализ);
- реализованы следующие средства визуализации работы классификаторов: графики ROC, график Precision-Recall, визуализация матрицы ошибок, столбчатая диаграмма результатов

классификации, визуализация матрицы точностных показателей классификатора.

В разработанном программном обеспечении использованы следующие библиотеки:

- widgets – библиотека для создания пользовательского интерфейса посредством форм и виджетов.
- numpy – библиотека, содержащая набор метаматематических функций и – добавляющая возможность работы с массивами;
- pandas – библиотека содержащая методы для работы с табличными данными;
- matplotlib – библиотека содержащая в себе методы для визуализации данных;
- scikit-learn – библиотека, реализующая различные методы построения классификаторов;
- XGBoost – библиотека, реализующая метод построения классификатора основанного на градиентном бустинге;
- LightGBM – библиотека, реализующая метод Light Gradient Boosting Machine;
- pycaret – библиотека содержащая методы для работы с данными в формате json.

Рассмотрим логику работы программного обеспечения. Сначала (в блоке «Вспомогательный код») производится импорт всех перечисленных выше библиотек. Затем с помощью метода uplod() создается и отображается на экран виджет для загрузки файла. Виджет содержит в себе кнопку «Обзор», при нажатии на которую откроется стандартное окно с возможность выбора файла для загрузки. В процессе загрузки файла виджет будет отображать тип файла, его размер и процент загрузки файла. Для дальнейшей работы указатель на загруженный файл сохраняется в переменную data_file (рисунок 15).

Данных из загруженного файла конвертируется с помощью метода read_csv() и сохраняются в переменную data (тип переменной - DataFrame). Библиотека Pandas предоставляет для переменных типа DataFrame множество встроенных методов, облегчающих вывод и анализ содержащихся в них данных.

Затем производится вывод содержимого переменной data на экран. Это необходимо для визуального контроля корректности загруженных данных (рисунок 16)

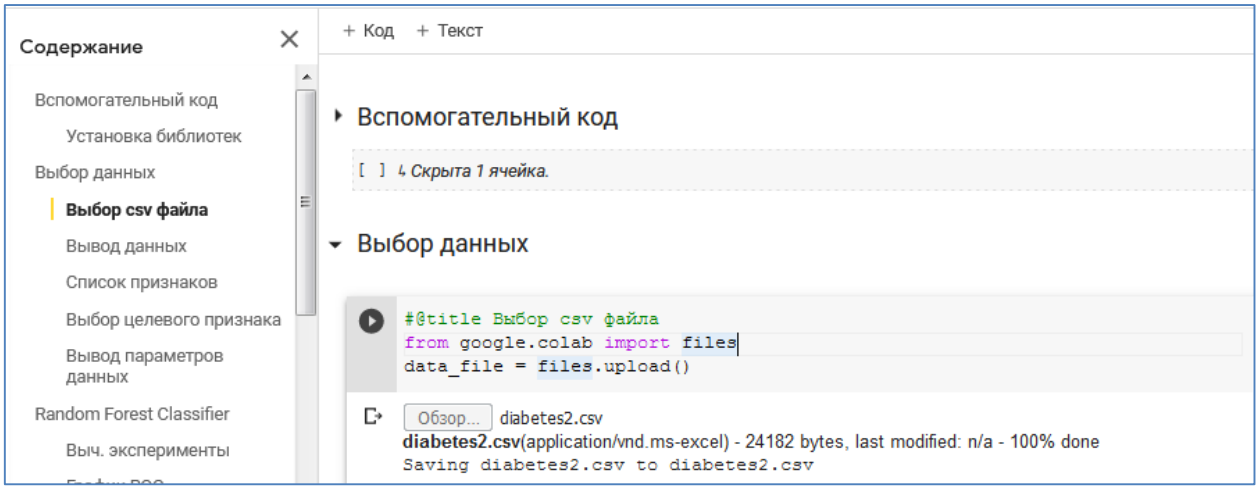

Рисунок 15 – Использование виджета для загрузки csv файла с исходными данными

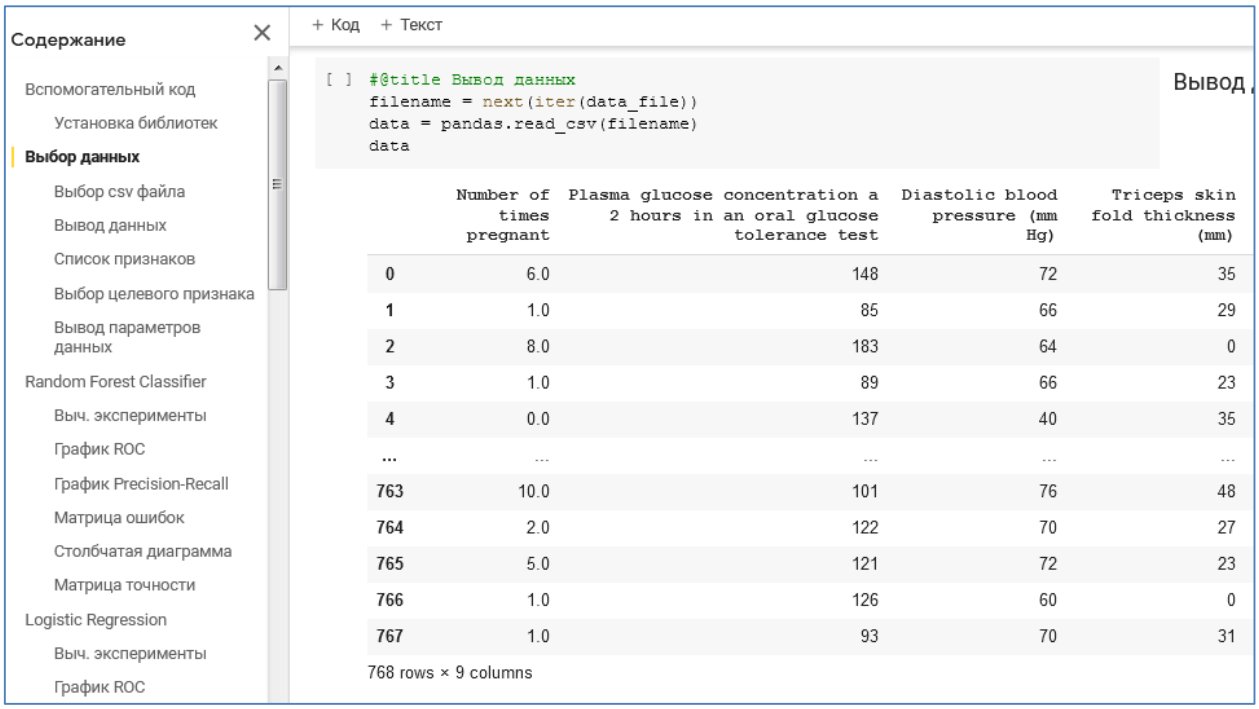

Рисунок 16 – Вывод загруженных данных на экран

Затем с помощью метода columns.value.tolist() осуществляется сбор названий столбцов (признаков) в исходных данных с последующим сохранением результатов в переменную var_list (тип переменной – List).

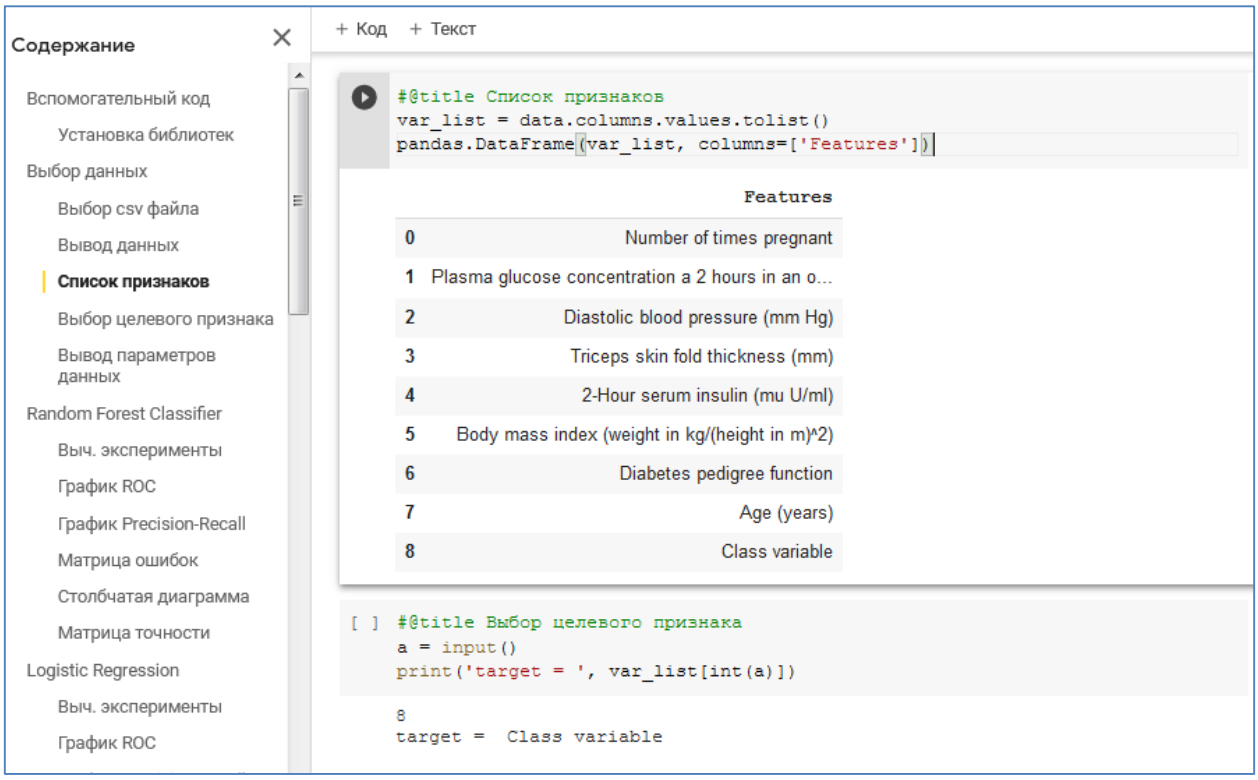

Рисунок 17 – Задание 8 столбца (Class variable), как целевого признака

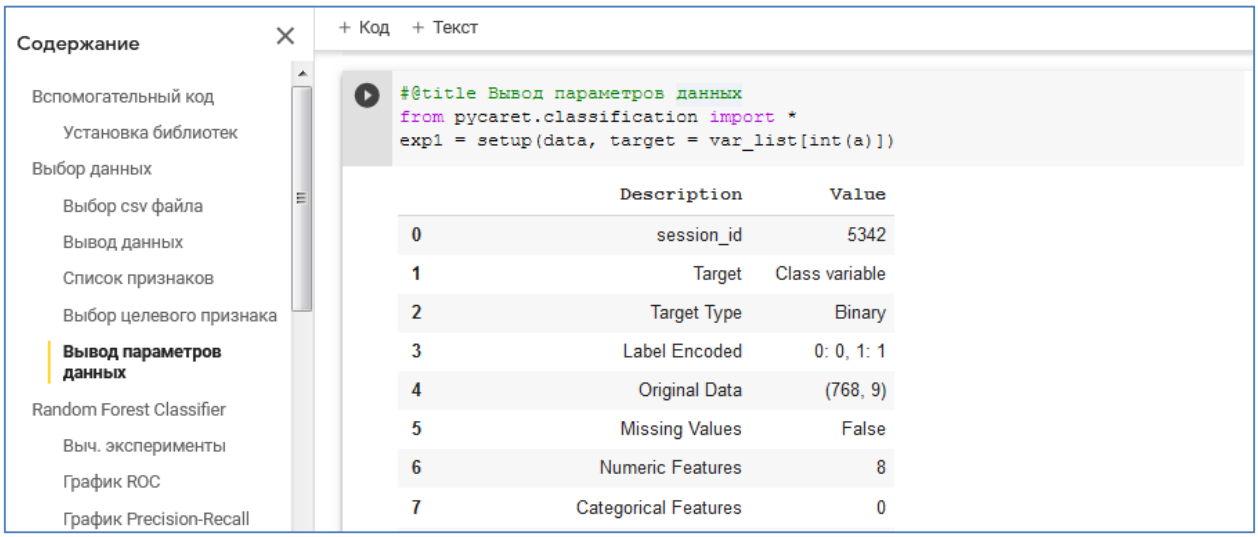

Рисунок 18 – Определение основных параметров данных

С помощью метода DataFrame() осуществляется конвертирование содержимого переменной var_list в переменную с типом DataFrame. Это

выполняется для визуализации названий столбцов в виде таблицы (рисунок 17). Далее пользователю предлагается выбрать номер столбца, значения из которого впоследствии будут использованы в качестве меток класса (целевой столбец). Запрос на ввод номера столбца реализован с помощью метода input().

С помощью стандартных методов из библиотеки Pandas осуществляется вывод статистической информации о данных хранящихся в переменной Data: размер таблицы, количество категориальных признаков, количество числовых признаков и т.д. (рисунок 18).

На следующем этапе осуществляется проведение вычислительных экспериментов по тестированию методов построения классификаторов и визуализация результатов работы классификаторов на тестовых данных.

В программном обеспечении методы тестируются друг за другом последовательно без остановки. После завершения тестирования методов формируется сводная таблица. В сводной таблице на первой строчке находится метод, обеспечивший построение наиболее точного классификатора (рисунок 20).

Так как количество анализируемых методов построения классификаторов значительное (13 штук), то, для удобства обзора результатов тестирования, предусмотрено наличие левой навигационной панели (рисунок 19).

В данной панели ссылки на результаты тестирования сгруппированы по применяемым методам построения классификаторов. При выборе в панели нужного пункта будет осуществлен вывод на экран запрашиваемого элемента отчета (рисунок 19).

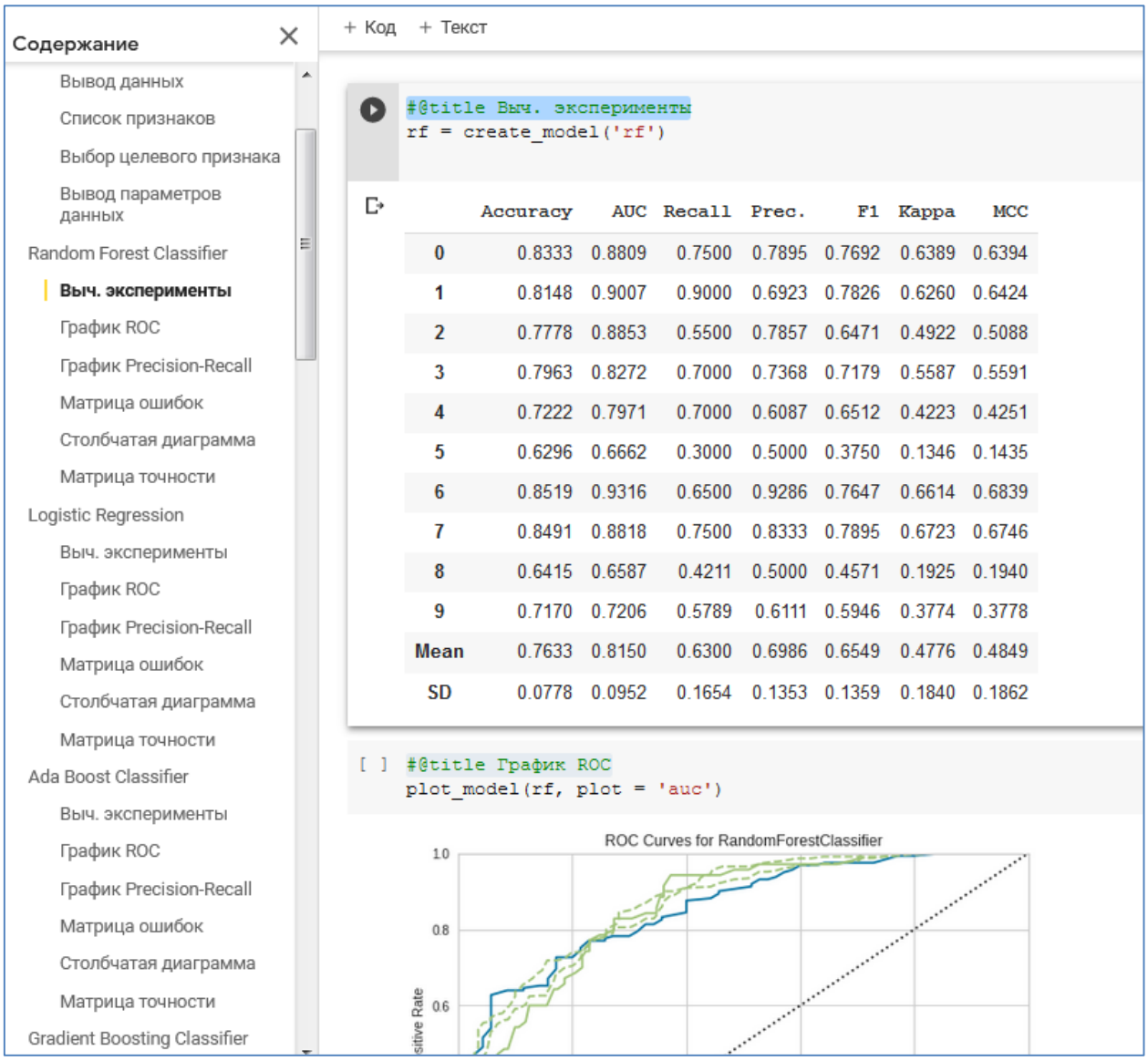

Рисунок 19 – Последовательное проведение вычислительных экспериментов и визуализация результатов

3.2 Тестирования программного обеспечения на медицинских данных

Рассмотрим пример работы программного обеспечения на медицинском наборе данных, который описывался во втором разделе работы.

Как видно из вывода программы, представленном на рисунке 20, наилучшим методом построения бинарного классификатора является Gradient Boosting (градиентный бустинг). Данный метод занимает первую строчку данной таблицы, так как классификатор, полученный с использованием данного метода, обладает наилучшими значениями *Accuracy*, *Recall*, *F1_score*, *Kappa*, *MCC* по сравнению с другими классификаторами.

```
#@title Результаты сравнения
result = compare_models()
```

Результаты ср

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
gbc	Gradient Boosting Classifier	0.7726	0.8259	0.6705	0.7074	0.6827	0.5063	0.5121	0.129
ridge	Ridge Classifier	0.7688	0.0000	0.5697	0.7477	0.6402	0.4759	0.4897	0.018
nb	Naive Bayes	0.7687	0.8128	0.6289	0.7009	0.6576	0.4855	0.4909	0.018
ada	Ada Boost Classifier	0.7687	0.8007	0.6358	0.7125	0.6629	0.4900	0.4988	0.114
lda	Linear Discriminant Analysis	0.7669	0.8330	0.5697	0.7425	0.6383	0.4722	0.4855	0.020
rf	Random Forest Classifier	0.7633	0.8150	0.6300	0.6986	0.6549	0.4776	0.4849	0.518
lr	Logistic Regression	0.7577	0.8297	0.5547	0.7284	0.6230	0.4508	0.4644	0.054
qda	Quadratic Discriminant Analysis	0.7390	0.8051	0.5692	0.6727	0.6150	0.4199	0.4244	0.018
et	Extra Trees Classifier	0.7335	0.8022	0.5645	0.6710	0.6075	0.4085	0.4162	0.465
lightgbm	Light Gradient Boosting Machine	0.7297	0.7921	0.6103	0.6422	0.6188	0.4114	0.4172	0.054
knn	K Neighbors Classifier	0.7037	0.7347	0.5203	0.6230	0.5596	0.3413	0.3492	0.121
dt	Decision Tree Classifier	0.6628	0.6384	0.5450	0.5460	0.5428	0.2766	0.2786	0.019
svm	SVM - Linear Kernel	0.5980	0.0000	0.2724	0.4427	0.2895	0.0615	0.0730	0.020

Рисунок 20 – Результат сравнения разных методов построения бинарных классификаторов

На втором месте на Ridge Classifier, а на третьем – Naïve Bayes (наивный байесовский классификатор). Наихудшим методом для построения классификатора на имеющихся данных оказался SVM (метод опорных векторов). Причины того, что какой-то из методов оказывается лучше другого, связаны непосредственно с анализируемыми данными (наличие аномальных значений, наличие неинформативных признаков, малый размер обучающей выборки и т.д.)

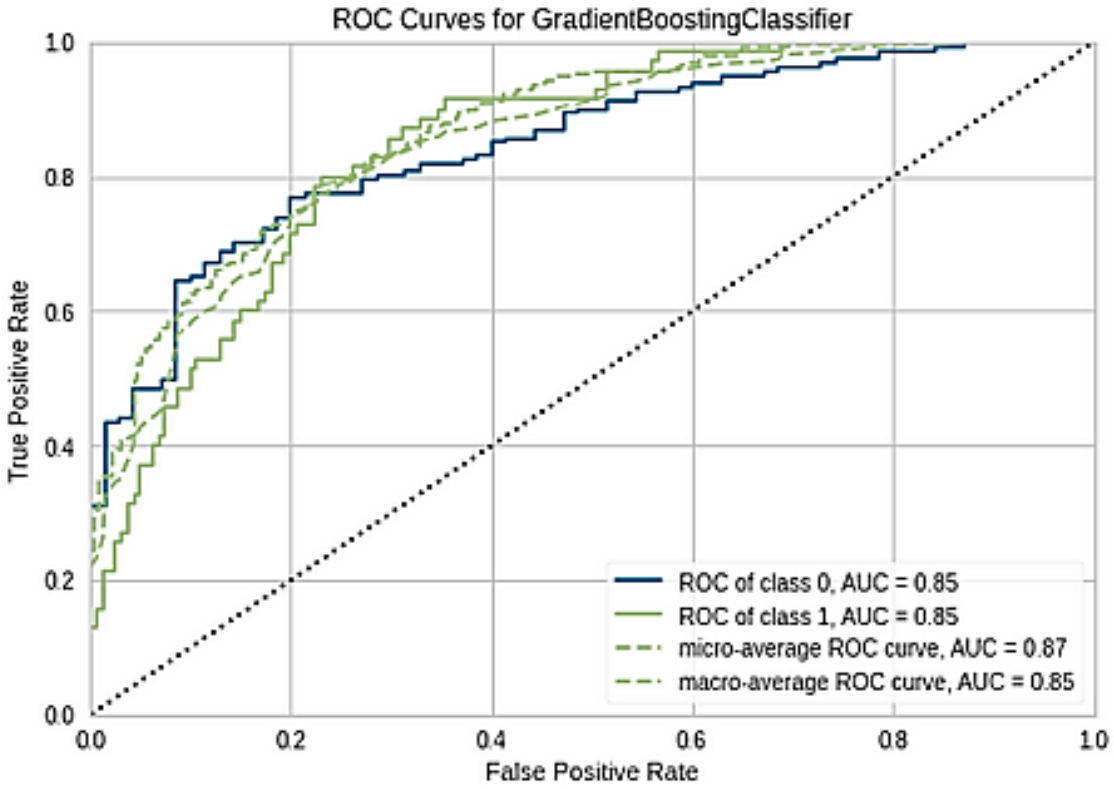

Рисунок 21 – График ROC для классификатора Gradient Boosting Classifier

Результаты визуализации работы классификатора Gradient Boosting Classifier на тестовом наборе данных, сгенерированные разработанным программным обеспечением, показаны ниже:

- график ROC показан на рисунке 21;
- график Precision-Recall показан на рисунке 22;
- визуализация матрицы ошибок показана на рисунке 23;
- столбчатая диграмма работы классификатора на тестовой выборке данных показана на рисунке 24;
- визуализация матрицы точности показана на рисунке 25.

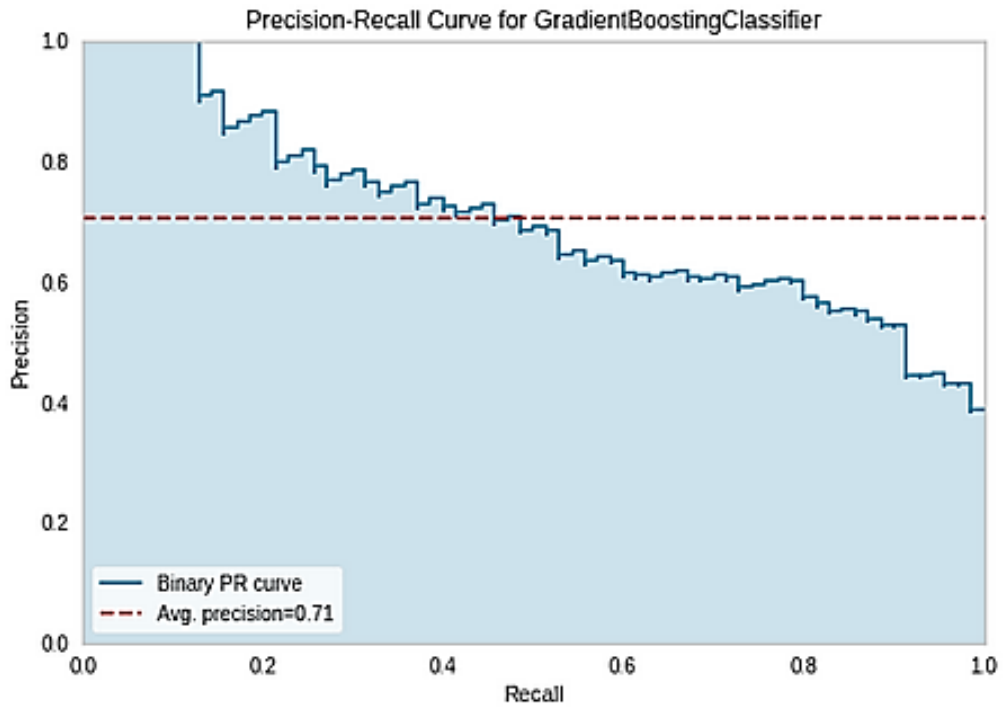

Рисунок 22 – График Precision-Recall для классификатора Gradient Boosting Classifier

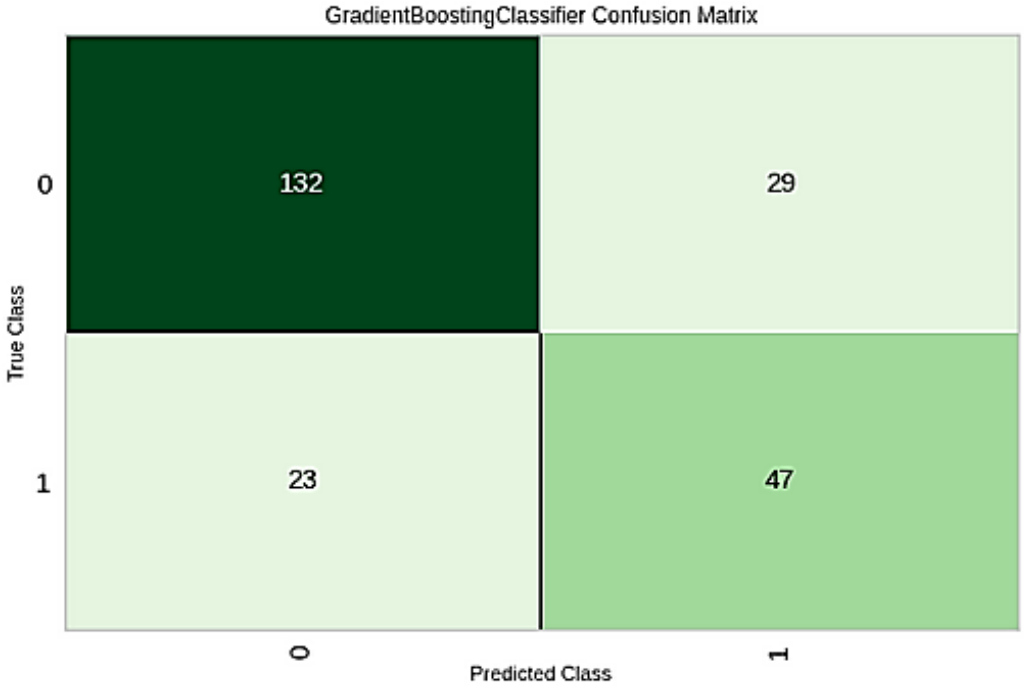

Рисунок 23 – Матрица ошибок для классификатора Gradient Boosting Classifier

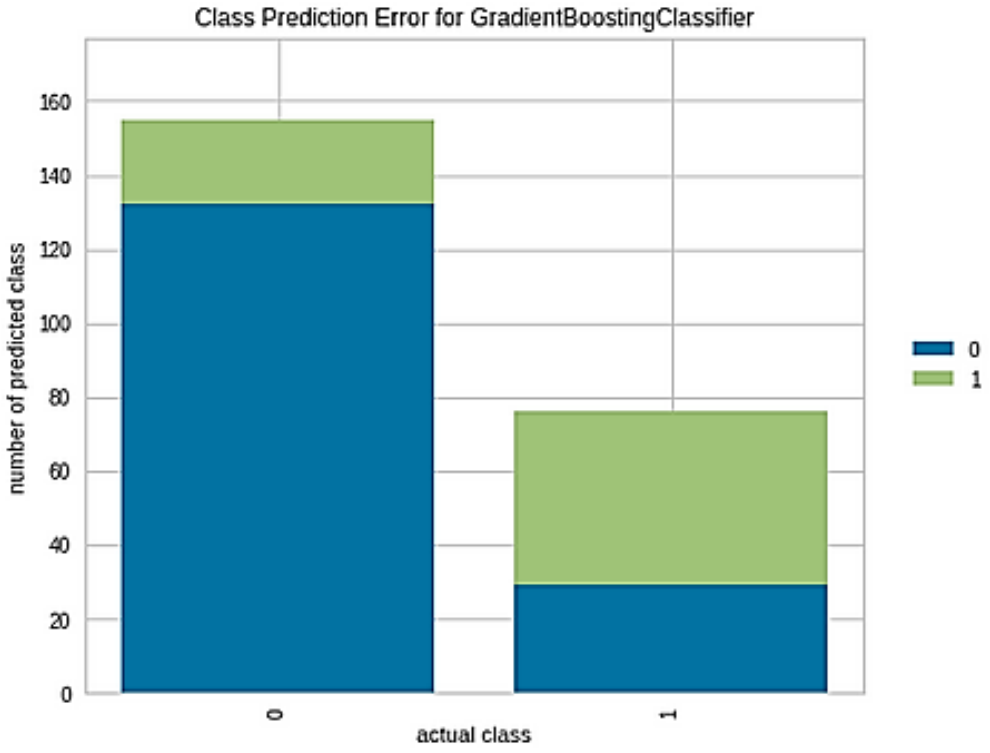

Рисунок 24 – Столбчатая диаграмма для Gradient Boosting Classifier

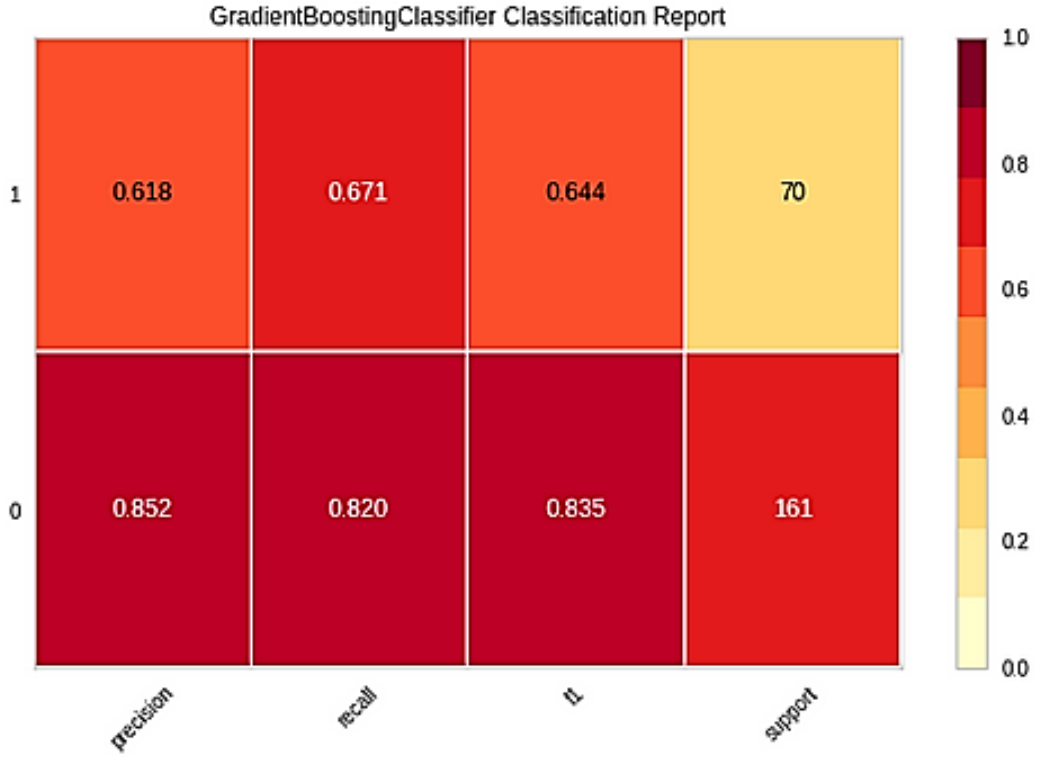

Рисунок 25 – Матрица точности для Gradient Boosting Classifier

Как видно из рисунков 20-25, разработанное программное обеспечение позволяет визуализировать работу выбранного классификатора на тестовом наборе данных.

3.3 Выводы по главе

Разработанное программное обеспечение позволяет производить сравнение эффективности методов построения классификаторов на выбираемом пользователе наборе данных. При этом реализована поддержка следующих методов: Logistic Regression (логистическая регрессия), Ada Boost Classifier (метод AdaBoost), Linear Discriminant Analysis (линейный дискриминантный анализ), Ridge Classifier (классификатор Ridge), Gradient Boosting Classifier (градиентный бустинг), Extra Trees Classifier, Random Forest Classifier (случайный лес), Light Gradient Boosting Machine, K Neighbors Classifier (K ближайших соседей), Decision Tree Classifier (деревья решений), Naïve Bayes (наивный байесовский классификатор), SVM - Linear Kernel (метод опорных векторов), Quadratic Discriminant Analysis (квадратичный дискриминантный анализ).

Показан пример использования разработанного программного обеспечения на медицинском наборе данных. В нашем случае наилучшим методом построения классификатора является Gradient Boosting Classifier (градиентный бустинг).

Заключение

На основании проведенных исследований были сделаны следующие выводы:

С развитием машинного обучения появляется все больше различных алгоритмов построение классификаторов, основанных на разных математических принципах. При этом заранее не известно, какой из алгоритмов обеспечит построение (обучение) наилучшей модели, наиболее точно описывающей имеющиеся данные. Поэтому актуальной проблемой является разработка программного обеспечения для автоматизации поиска наилучшего метода построения модели (бинарного классификатора).

Описан подход для определения лучшего метода построения классификатора для выбранных исходных данных. Данный подход включает в себя: разделение исходных данных на 10 приблизительно равных не пересекающихся множеств; проведение для каждого метода построения классификатора 10 вычислительных экспериментов, в которых по очереди 1 множество выступает в роли тестовой выборки данных, а остальные множества в роли тренировочного набора данных; расчет на основе вычислительных экспериментов для каждого метода средних показателей точности (*Accuracy*, *Recall*, *Precision*, *F1_score*, *Kappa*) и объединение их в сводную таблицу.

Для упрощения сравнения эффективности различных классификаторов предложено использование следующих средств визуализации: график ROC, график Precision-Recall, визуальная интерпретация матрицы ошибок, столбчатая диаграмма ошибок классификации и визуальная интерпретация матрицы точностных показателей классификатора.

На языке Python разработано программное обеспечение для сравнения эффективности методов построения классификаторов на выбираемом пользователем наборе данных. При этом реализована поддержка следующих методов: Logistic Regression (логистическая регрессия), Ada Boost Classifier

(метод AdaBoost), Linear Discriminant Analysis (линейный дискриминантный анализ), Ridge Classifier (классификатор Ridge), Gradient Boosting Classifier (градиентный бустинг), Extra Trees Classifier, Random Forest Classifier (случайный лес), Light Gradient Boosting Machine, K Neighbors Classifier (K ближайших соседей), Decision Tree Classifier (деревья решений), Naïve Bayes (наивный байесовский классификатор), SVM - Linear Kernel (метод опорных векторов), Quadratic Discriminant Analysis (квадратичный дискриминантный анализ).

На имеющемся наборе медицинских данных по диагнозам диабета, состоящем из 769 записей наилучшим методом построения бинарного классификатора оказался - Gradient Boosting Classifier. Точностные показатели классификатора, определенные на тестовой выборке данных: *Accuracy*=0.7726, *AUC*=0.8259, *Recall*=0.6705, *Precision*=0.7074, *F1_score*=0.6827, *Kappa*=0.5063, *MCC*=0.5121.

Таким образом, все поставленные в работе задачи выполнены, и цель достигнута.

Список используемой литературы

1. Болодурина, И.П. Разработка модели классификации потоков данных программно-управляемой инфраструктуры виртуального центра обработки данных / И.П. Болодурина, Д.И. Парфенов // Информационные технологии моделирования и управления. №2, 2017. – ООО "Издательство "Научная книга" (Воронеж), 2017. – с. 151-159. – Текст : непосредственный.

2. 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. – Текст : непосредственный.

3. 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. – Текст : непосредственный.

4. 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. – Текст : непосредственный.

5. 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. – Текст : непосредственный.

6. 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. – Текст : непосредственный.

7. 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. – Текст : непосредственный.

8. 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. – Текст : непосредственный.

9. 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. – Текст : непосредственный.

10. 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. – Текст : непосредственный.

11. 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. – с. 177-180. – Текст : непосредственный.

12. Осколков, В.М. Применение параллельных вычислений для прогнозирования на основе алгоритма машинного обучения Random Forest / В.М. Осколков, Н.И. Шаханов, И.А. Варфоломеев, О.В. Юдина, Л.Н. Виноградова, Е.В. Ершов // Сборник трудов конференции Оптико-электронные приборы и устройства в системах распознавания образов, обработки изображений и символьной информации. Распознавание, Курск, 16–19 мая 2017 года. – Юго-Западный государственный университет (Курск), 2017. – с. 267-269. – Текст : непосредственный.

13. 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. – Текст : непосредственный.

14. 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. – Текст : непосредственный.

15. 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. – Текст : непосредственный.

16. 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. 22–26 мая 2017. – Северо-Кавказский федеральный университет (Ставрополь), 2017. – с. 361-363. – Текст : непосредственный.

17. 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. – Текст : непосредственный.

18. 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. – Текст : непосредственный.

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

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

21. 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. – Текст : непосредственный.

22. 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. – Текст : непосредственный.