

Министерство образования и науки Российской Федерации

**ФИЛИАЛ ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО
ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МЭИ»
в г. СМОЛЕНСКЕ**

Кафедра менеджмента и информационных технологий в экономике

Специальность 080801 Прикладная информатика (в экономике)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ДИПЛОМНОМУ ПРОЕКТУ

на тему

**«АВТОМАТИЗИРОВАННАЯ ИНФОРМАЦИОННАЯ СИСТЕМА ДЛЯ БИНАРНОЙ
КЛАССИФИКАЦИИ ОБЪЕКТОВ НА ОСНОВЕ ЛОГИСТИЧЕСКОЙ РЕГРЕССИИ»**

Студент
группы ПИЭ-10 _____ Н. А. Салов
(подпись, дата)

Руководитель
канд. техн. наук, доц. _____ Б. В. Окунев
(подпись, дата)

Смоленск 2015

АННОТАЦИЯ

Дипломный проект на тему «Автоматизированная информационная система для бинарной классификации объектов на основе логистической регрессии» выполнен студентом группы ПИЭ-10 филиала ФГБОУ ВПО «Национальный исследовательский университет «МЭИ» в г. Смоленске. Объем работы составляет 73 страницы. Работа содержит 67 рисунков и 5 таблиц подтверждающих и иллюстрирующих выводы дипломного проекта.

Основная часть данного дипломного проекта состоит из трех глав. В первой главе рассматриваются области применения и методы решения задачи бинарной классификации, теоретические и технологические аспекты применения логистической регрессии, проводится анализ существующих разработок и обоснование выбора технологии проектирования. Вторая глава посвящена проектированию автоматизированной информационной системы, формированию её функциональных возможностей, рассмотрению различного обеспечения задачи автоматизации и проведению тестирования разработанного программного продукта. В третьей главе проводится формирование технологической среды информационной системы, а также оценка эффективности проекта по ее внедрению.

Результаты данной работы распространяются под лицензией GNU GPL и доступны в репозитории по адресу <https://github.com/salov/dichotomy>.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	4
1 Аналитическая часть.....	6
1.1 Анализ областей применения и методов решения задачи бинарной классификации.....	6
1.2 Теоретические и технологические аспекты применения логистической регрессии	10
1.3 Анализ существующих разработок и обоснование выбора технологии проектирования.....	19
1.4 Выводы по главе.....	28
2 Проектная часть.....	30
2.1 Формирование функциональных возможностей разрабатываемой информационной системы.....	30
2.2 Информационное обеспечение задачи автоматизации	31
2.3 Алгоритмическое и программное обеспечение задачи автоматизации	38
2.4 Тестирование разработанной информационной системы	40
2.5 Выводы по главе.....	47
3 Методика внедрения и оценка эффективности проекта	49
3.1 Формирование технологической среды информационной системы	49
3.2 Обоснование эффективности проекта.....	65
3.3 Выводы по главе.....	67
ЗАКЛЮЧЕНИЕ	68
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	70
ПРИЛОЖЕНИЕ А - Алгоритмическое обеспечение задачи автоматизации....	73
ПРИЛОЖЕНИЕ Б – Листинг программы.....	77

ВВЕДЕНИЕ

Решение задач бинарной классификации позволяет повышать обоснованность и надежность принимаемых решений во множестве предметных областей. Автоматизация решения данной задачи позволяет повысить также и эффективность принятия решения, а также позволяет расширять сферу применения рассматриваемого инструмента, постепенно охватывая малый и средний бизнес. Кроме того, автоматизация позволяет снизить порог необходимых пользователю знаний в области математической статистики для применения инструмента бинарной классификации. Всё это делает актуальной цель, на достижение которой направлен данный дипломный проект.

Цель дипломного проекта – разработка автоматизированной системы для бинарной классификации объектов на основе логистической регрессии. Для достижения поставленной цели необходимо решить следующие задачи:

- изучить области применения и методы решения задач бинарной классификации;
- изучить теоретические и технологические аспекты логистической регрессии;
- проанализировать существующие разработки и выбрать технологию проектирования;
- сформировать функциональные требования к информационной системе;
- описать информационное и алгоритмическое обеспечение задачи автоматизации;
- разработать информационную систему в соответствии с разработанной спецификацией и протестировать ее;
- сформировать технологическую среду информационной системы;
- обосновать эффективность проекта.

Объектом исследования данной работы являются инструменты интеллектуального анализа данных. Предмет исследования – решение задач бинарной классификации объектов.

В качестве теоретической базы дипломного проекта были использованы монографии, научные статьи и электронные ресурсы, посвященные проблемам интеллектуального анализа данных, бинарной классификации, вопросам проектирования информационных систем.

В процессе исследования был проведен анализ специальной литературы, документов, периодики, авторитетных мнений специалистов в сфере интеллектуального анализа данных и проектирования информационных систем. Для этого использовались системно-функциональный, сравнительный, логический и исторический методы познания.

1 Аналитическая часть

1.1 Анализ областей применения и методов решения задачи бинарной классификации

При применении инструментов интеллектуального анализа данных часто приходится иметь дело с задачами, решением которых является определение возможности отнесения объектов к одному из двух рассматриваемых классов на основе определенного правила классификации. Класс таких задач называется задачами бинарной классификации объектов [1]. Решение задач данного класса становится актуальным в широком круге предметных областей. Проведение бинарной классификации необходимо в медицине для повышения эффективности диагностики заболеваний, в акушерстве, психиатрии, психологии и социологии, для расчета прогноза исхода оперативного лечения [2]. Кроме того, бинарная классификация нашла широкое распространение в банковском деле при построении рейтинга заемщика и управлении кредитными рисками [3], в маркетинге данная задача решается при моделировании поведения клиентов. Бинарная классификация часто проводится при изучении социальных и, в частности, демографических процессов [4].

Помимо описанных областей применения бинарная классификация может найти применение в таможенном деле. В частности, в рамках компетенции отдела применения системы управления рисками для определения и снижения степени риска, связанного с вероятностью несоблюдения таможенного законодательства таможенного союза. В данном случае бинарная классификация может применяться для определения товаров, транспортных средств международной перевозки, документов и лиц, подлежащих таможенному контролю. Также решение задачи может использоваться для определения форм таможенного контроля и степени проведения контроля [5]. Решение задачи бинарной классификации в таможенном деле способствует достижению следующих целей:

- обеспечение мер по защите национальной безопасности, жизни и здоровья человека, охране окружающей среды;

- повышение эффективности использования имеющихся в распоряжении таможенных органов ресурсов за счет сосредоточения внимания на областях повышенного риска;
- выявление, прогнозирование и предотвращение нарушений таможенного законодательства;
- ускорение проведения таможенных операций при перемещениях товаров через таможенную границу [5].

Как видно из описания задачи бинарной классификации объектов, сфера актуальности ее решения крайне широка. Современные тенденции и уровень развития информационно-телокоммуникационных и мобильных технологий способствует всё большему расширению этой сферы. Постепенно охватываются субъекты малого и среднего бизнеса за счет снижения издержек на решение таких задач, с одновременным повышением эффективности деятельности и резким улучшением конкурентоспособности экономических субъектов, применяющих в своей деятельности современные инструменты, в том числе решающих задачи бинарной классификации объектов.

В настоящее время существует множество методов решения задач бинарной классификации, наиболее распространенными из которых являются [1]:

- бинарные деревья решений (в частности, алгоритм CART);
- логистическая регрессия;
- искусственные нейронные сети.

Алгоритм CART строит бинарные деревья решений, содержащие по два потомка в каждом узле. Структура дерева представляет собой «листья» и «ветки». На «ветках» (ребрах) дерева записаны атрибуты, влияющие на целевую функцию, в «листьях» записаны значения целевой функции, а в остальных узлах - атрибуты, по которым различаются наблюдения. Чтобы классифицировать новое наблюдение, надо пройти по дереву до листа и выдать соответствующее значение. Каждый лист - значение целевой переменной, измененной в ходе движения от корня по листу. Каждый внутренний узел соответствует одной из входных переменных. Это процесс, повторяющийся на каждом из полученных подмножеств. Рекурсия

завершается тогда, когда подмножество в узле имеет те же значения целевой переменной, то есть не добавляет ценности для будущих классификаций [6].

Достоинствами данного метода являются:

- не требует подготовки данных (нормализации, добавления фиктивных переменных);
- возможность работы с категориальными и интервальными переменными;
- позволяет оценить модель при помощи статистических тестов;
- позволяет работать с большими объемами информации без специальных подготовительных процедур.

Кроме того, данный метод имеет определенные недостатки:

- практическое применение алгоритма основано на эвристических алгоритмах, оптимизирующих решения локально в каждом узле, что делает невозможным обеспечение оптимальности всего дерева в целом;
- высокая вероятность создания слишком сложных конструкций, недостаточно полно представляющих данные;
- сложность описания концептов моделью, что затрудняет понимание модели в целом [6].

Логистическая регрессия – разновидность множественной регрессии, которая применяется в анализе связей между несколькими независимыми переменными (регрессорами) и зависимой переменной (регрессией). Для решения задач, когда зависимая переменная является бинарной (в том числе и задач бинарной классификации) применяется бинарная логистическая регрессия. Данный метод имеет следующие преимущества [1, 9]:

- высокая робастность моделей на основе данного инструмента;
- технологическая простота модели – такие модели просты в понимании и интерпретации;
- способность принимать на вход данные любого рода;
- применимость инструмента ROC-анализа для оценки качества модели.

Недостатки метода логистической регрессии [1,9]:

- необходимость в достаточном объеме и приемлемом качестве обучающих данных;

- данному алгоритму не всегда удается построить регрессионную модель с приемлемой ошибкой.

Искусственная нейронная сеть (ИНС) – параллельно-распределенная система процессорных элементов (нейронов), выполняющих простейшую обработку данных, которая способна настраивать свои параметры в ходе обучения на эмпирических данных [8].

ИНС обладают следующими достоинствами:

- способность изменять веса связей между нейронами с помощью наборов обучающих примеров;

- параллельная обработка данных позволяет ускорять обработку информации;
- способность адаптировать веса связей к изменениям во внешнем окружении;
- отказоустойчивость за счет большого количества связей между нейронами.

Помимо достоинств аппарат ИНС имеет следующие недостатки:

- неоднозначность решения задач из-за эвристичности подходов к проектированию;

- необходимость многоцикловой настройки внутренних элементов и связей между ними;

- сложности с поиском достаточного количества обучающих примеров [8].

Решение большинства задач интеллектуального анализа данных является сложной и многоэтапной процедурой, а результаты применения тех или иных методов решения часто сложно однозначно интерпретировать. Каждый из подходов имеет свои преимущества и недостатки, поэтому одновременное использование нескольких моделей способствует эффективному решению задач [1].

В данном случае в работе рассматривается только один метод решения задачи бинарной классификации объектов – логистическая регрессия. Такое ограничение объясняется ограниченностью объема данной работы, а выбор именно этого инструмента объясняется технологической простотой и доступностью обучения и применения данной модели [7].

1.2 Теоретические и технологические аспекты применения логистической регрессии

Решение задач бинарной классификации выражается выходной переменной, которая имеет категориальный характер, то есть переменной, принимающей значения из некоторого ограниченного набора категорий. Такие наборы зачастую связаны с неисчисляемыми признаками, такими как, например, названия товаров и услуг, имена людей, исходы событий (да/нет) и тому подобные [1].

Являясь методом бинарной классификации, логистическая регрессия дает возможность оценивать вероятность принятия выходной переменной одного из двух возможных значений, то есть вероятность исхода. Под исходом в данном случае понимается показатель или признак, служащий объектом исследования. Например, при проведении кредитного скоринга вероятность исхода служит критерием целесообразности выдачи кредита [1].

Условное среднее $E(y|x)$, представляющее собой ожидаемое значение выходной переменной при заданном уровне входной, в случае логистической регрессии имеет вид, представленный в формуле (1.1).

$$E(y|x) = p(x) = \frac{1}{1+e^{-z}}, \quad (1.1)$$

где e – основание натурального логарифма, p – условное среднее, z определяется по формуле (1.2) [7].

$$z = \beta_0 + \sum_{i=1}^n \beta_i x_i, \quad (1.2)$$

где β_i – коэффициенты логистической регрессии, x_i – значения i -ой независимой (входной переменной) [7].

Функция, описанная уравнением (1.1) носит название логистической, а графики, описывающие поведение такой функции, называют сигмоидами, что

объясняется их S-образной формой. Одна из таких кривых представлена на рисунке 1.1.

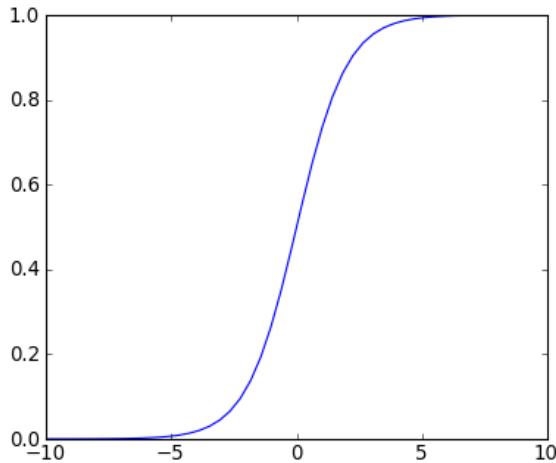


Рисунок 1.1 – Пример графика логистической регрессии

Логистическая регрессия определена на бесконечности, а область значений данной функции находится на отрезке $[0; 1]$. Такая область значений является решением проблемы ограничения вероятности исхода в необходимых пределах, что достигается при помощи логистического (логит) преобразования, обладающего свойствами линейности, непрерывности и определенности на бесконечности. Данное преобразование представлено в формуле (1.3).

$$g(x) = \ln \frac{p(x)}{1-p(x)} = \beta_0 + \beta_1 x, \quad (1.3)$$

где $p(x)$ – логистическая регрессия, β_i – коэффициенты логистической регрессии, x – значение независимой (входной переменной) [9].

Учитывая описанные выше свойства логистической регрессии, данная функция интерпретируется как вероятность приобретения выходной переменной значения 1 (положительный исход) или значения 0 (отрицательный исход). При этом, если $p(x)$ интерпретируется, как вероятность положительного исхода, то вероятность отрицательного исхода можно интерпретировать, как $1 - p(x)$.

В задачах бинарной классификации, решением которых является определение вероятности отнесения объекта к одному из двух классов, крайне важной становится проблема выбора точки отсечения, которая является численным значением порога вероятности, разделяющим два рассматриваемых класса. Важность данной проблемы объясняется тем, что занижение точки отсечения ведет к увеличению вероятности появления ошибки определения ложноположительных исходов, а завышения точки отсечения увеличивает вероятность ошибки определения ложноотрицательных исходов. Таким образом, актуальной становится задача выбора такой точки отсечения, чтобы модель наиболее точно классифицировала положительные и отрицательные исходы, одновременно минимизировав количество ложноположительных и ложноотрицательных классификаций [1].

Одним из инструментов, используемых для решения поставленной задачи, является ROC-анализ. Аббревиатура ROC расшифровывается как receiver operating characteristic (рабочая характеристика приемного устройства), что показывает появление данного инструмента в области задач бинарной классификации из области обработки сигналов в радиолокации [9].

Базовыми понятиями, которыми оперирует данный инструмент, являются ошибки I-го и II-го рода. В виду того, что модель логистической регрессии строится на основе некоторой обучающей выборки, и все входящие в такую выборку наблюдения соответствуют положительному и отрицательному исходу, то при работе модели возможно возникновение четырех вариантов развития ситуации, которые представлены в таблице сопряженности - таблица 1.1 [9].

Таблица 1.1 – Таблица сопряженности

		Исход в обучающей выборке
Исход, определенный моделью	Положительный	Отрицательный
Положительный	Истинно положительный случай	Ложно положительный случай (ошибка II-го рода)
Отрицательный	Ложно отрицательный случай (ошибка I-го рода)	Истинно отрицательный случай

Для целей ROC-анализа часто применяют относительные показатели, основанные на абсолютных, описанных в таблице 1.1. Так, доля истинно положительных случаев (TPR) рассчитывается по формуле (1.4).

$$TP_R = \frac{TP}{TP+FN} * 10_0 \%, \quad (1.4)$$

где TP – число истинно положительных случаев, FN – число ложно отрицательных случаев [1].

Второй используемый показатель – доля ложно положительных случаев, которая рассчитывается по формуле (1.5).

$$FP_R = \frac{FP}{FP+TN} * 10_0 \%, \quad (1.5)$$

где FP – число ложно положительных случаев, TN – число истинно отрицательных случаев [1].

Кроме перечисленных показателей для ROC-анализа большое значение имеют также показатели чувствительности (sensitivity) и специфичности (specificity). Чувствительность эквивалента доле истинно положительных случаев (TPR), а специфичность является долей истинно отрицательных случаев и рассчитывается по формуле (1.6).

$$Sp = \frac{TN}{TN+FP} * 10_0 \%, \quad (1.6)$$

где TN – число истинно отрицательных случаев, FP - число ложно положительных случаев, TN – число истинно отрицательных случаев [1].

При этом имеет место равенство, представленное в формуле (1.7).

$$FP_R = 10_0 - Sp, \quad (1.7)$$

где FPR – доля ложных положительных случаев, Sp – специфичность [1].

Исходя из сущности показателей чувствительности и специфичности, становится понятно, что модели с высокой чувствительностью часто проводят верную классификацию для положительных исходов и минимизирует ошибки I-го рода, а модель с высокой специфичностью, наоборот, часто верно классифицирует исследуемый объект при наличии отрицательного исхода и минимизирует ошибки II-го рода [1].

Основой ROC-анализа является построение ROC-кривой. Данная кривая строится путем изменения порога отсечения в интервале $[0; 1]$ с некоторым шагом Δx и расчета чувствительности и специфичности при каждом новом значении порога отсечения, что ведет к изменению количества верно и неверно классифицированных исходов на каждом шаге. Откладывая по оси абсцисс полученные значения чувствительности, а по оси ординат – значения FPR (100% – значение специфичности), получается ROC-кривая [9]. На рисунке 1.2 представлено несколько ROC-кривых, совмещенных в одной системе координат.

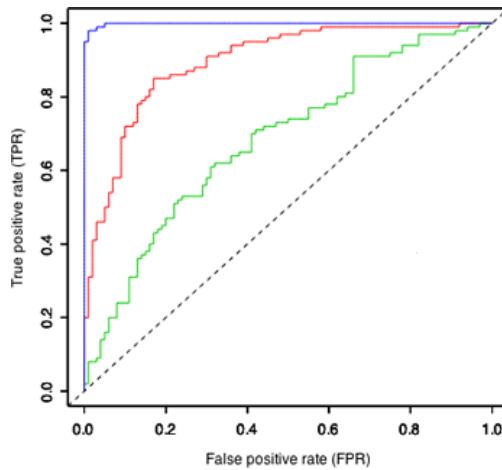


Рисунок 1.2 – ROC-кривые

Кроме того, на данном рисунке график дополнен биссектрисой угла начала координат. ROC-кривая идеального классификатора должна проходить через верхний левый угол, при этом, чем ближе кривая к верхнему левому углу, тем выше предсказательная способность соответствующей модели. Напротив, чем ближе

кривая к биссектрисе угла начала координат (диагональная прямая), тем ниже предсказательная способность модели, а сама прямая визуализирует бесполезный классификатор, чья классифицирующая способность не превышает эффективности метода случайного угадывания [1].

В некоторых случаях суждения о классифицирующих способностях нескольких моделей на основе визуального сравнения соответствующих им ROC-кривых затруднено. В таком случае для сравнения таких моделей возможно применение показателя площади под ROC-кривой (Area under curve - AUC), одним из способов вычисления которого является применение формулы площади криволинейной трапеции, представленной в формуле (1.8).

$$AU_C = \int f(x)dx = \sum_i \frac{Se_{i+1} + Se_i}{2} * (FP_{R^{i+1}} + FP_{R^i}), \quad (1.8)$$

где Se_i – значение чувствительности в i -ой точке, FPR_i – значение 100% - специфичность в i -ой точке [9].

Пример графической интерпретации площади под кривой представлен на рисунке 1.3.

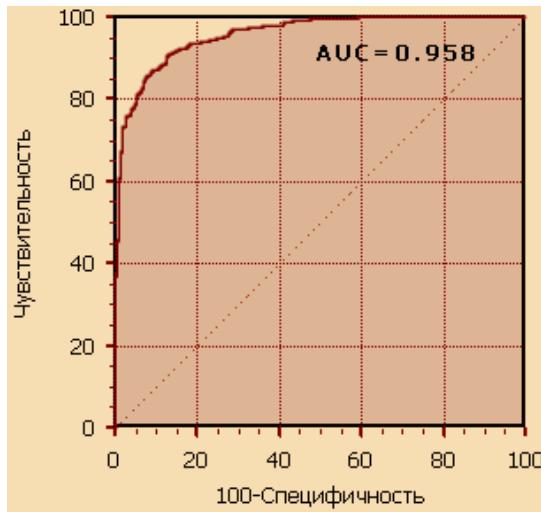


Рисунок 1.3 – Графическая интерпретация площади под ROC-кривой

Теоретически данный показатель может изменяться от 0,5, характеризуя бесполезную модель (когда ROC-кривая совпадает с диагональной прямой) до 1, характеризуя идеальную модель (когда ROC-кривая проходит через верхний левый угол). Учитывая то, что показатель площади под кривой не содержит информации о чувствительности и специфичности модели и предназначен в основном для сравнительного анализа, можно говорить о том, что чем больше данный показатель, тем большей классифицирующей способностью обладает модель. Косвенные суждения о качестве модели по данному показателю можно производить, пользуясь экспертными оценками, приведенными в таблице 1.2 [9].

Таблица 1.2 – Экспертные оценки качества модели по критерию площади под ROC-кривой

Интервал показателя AUC	Оценка качества модели
0,9 - 1	Отличное
0,8 – 0,9	Очень хорошее
0,7 – 0,8	Хорошее
0,6 – 0,7	Среднее
0,5 – 0,6	Неудовлетворительное

В идеальных условиях целесообразно выбирать ту модель, которая характеризуется 100% чувствительностью и специфичностью, но в реальных ситуациях добиться таких показателей представляется невозможным. Такая ситуация ставит задачу о нахождении оптимального порога отсечения. Выбор способа решения данной задачи индивидуален для каждой проблемной ситуации в виду специфики предметной области. Например, при диагностике некоторых заболеваний более целесообразно выбрать тот порог, в котором специфичность будет выше, так как вовремя выявить заболевание важнее, чем вероятность впустую потратить средства на лечение. Напротив, при других заболеваниях начало лечения при отсутствии уверенности в правильности диагноза может причинить больше вреда, чем ожидание результатов дополнительных исследований. В общем случае существует несколько критериев выбора порога отсечения, среди которых:

- требование минимальной величины чувствительности или специфичности, когда оптимальный порог находится в точке максимального значения того или другого показателя;

- требование максимальной суммарной чувствительности и специфичности;

- требование баланса между чувствительностью и специфичностью [1].

Для обеспечения приемлемого качества модели логистической регрессии необходима своевременная корректировка модели с точки зрения количественного и качественного состава ее регрессоров. Кроме того, для обеспечения возможности адаптации модели к изменяющимся условиям действительности необходима корректировка с точки зрения изменения коэффициентов модели.

Для обеспечения возможности адаптации модели предлагается осуществление обратной связи с пользователями системы классификации. После получения результатов работы модели и прошествии определенного интервала времени, который определяется индивидуально в зависимости от специфики рассматриваемой предметной области, необходимо получить значение реального исхода событий, который может как совпадать с данным моделью результатом, так и разниться с ним. После получения такой обратной связи необходимо дополнить обучающую выборку, использовавшуюся для обучения модели ранее, новым примером – в качестве значений регрессоров будут выступать заданные пользователем параметры классификации, а в качестве значения функции – реальный исход события. После такого дополнения обучающей выборки необходимо заново провести обучения модели на уже дополненной выборке. Такой подход позволит модели постоянно адаптироваться к реальным условиям ее применения.

Для корректировки модели с точки зрения количественного и качественного состава регрессоров предлагается использовать инструмент средних частных коэффициентов эластичности и стандартизованных коэффициентов. Коэффициенты регрессии β_i являются размерными величинами. В общем случае размерность коэффициента регрессии выражается в единицах измерения функции на единицу измерения регрессора x_i . Любое изменение единицы измерения регрессора

сказывается на коэффициенте регрессии. Стандартизованные же коэффициенты безразмерны, что делает возможным их сравнение. Расчет стандартизованных коэффициентов выполняется по формуле, представленной в формуле (1.9).

$$\beta'_i = \beta_i * \frac{S_i}{S_y}, \quad (1.9)$$

где β'_i – i-ый стандартизованный коэффициент, β_i – i-ый коэффициент регрессии, S_i – стандартное отклонение i-го регрессора, S_y – стандартное отклонение регрессии [11].

Такое сравнение наиболее актуально при оценке интенсивности влияния регрессоров на значение функции. Из-за различной размерности коэффициентов и различных средних значений регрессоров использование для сравнения коэффициентов регрессии в натуральном масштабе не представляется возможным – даже при большом значении коэффициента регрессии соответствующий регрессор может оказывать незначительное влияние, что объясняется, в основном, различной вариацией значений регрессоров. Стандартизованные же коэффициенты показывают, на какую часть стандартного отклонения изменяется среднее значение регрессии, если бы значение регрессора изменилось бы на стандартное отклонение при неизменности значений других регрессоров.

Средние частные коэффициенты эластичности показывают, на сколько процентов с среднем изменится значение регрессии при изменении регрессора на 1% от своего среднего уровня при неизменности значений других регрессоров. Средние частные коэффициенты эластичности рассчитываются по формуле, представленной в формуле (1.10).

$$\vartheta_i = \beta_i * \frac{\bar{x}_i}{\bar{y}}, \quad (1.10)$$

где ϑ_i – частный i-ый коэффициент эластичности, β_i – i-ый коэффициент модели, \bar{x}_i - среднее значение i-го регрессора, \bar{y} - среднее значение регрессии [11].

Комплексное использование двух описанных инструментов позволяет ранжировать регрессоры по степени их влияния на результат и на важность для модели в целом. Использование такого подхода позволяет корректировать модель с точки зрения количественного и качественного состава ее регрессоров, обеспечивая тем самым надлежащее качество, практическую классифицирующую способность модели и адаптировать модель к изменяющимся условиям действительности.

1.3 Анализ существующих разработок и обоснование выбора технологии проектирования

В связи с такой широкой сферой применения алгоритма бинарной классификации становится очевидным наличие некоторого количества программных средств, автоматизирующих применение данного подхода. Среди подобных программных средств, можно выделить ряд наиболее известных:

- СПАРК – система профессионального анализа рынков и компаний;
- STATISTICA, SPSS, R и прочие математические пакеты для статистического анализа, обладающие сходными характеристиками в рамках рассматриваемой задачи, поэтому было решено объединить их в общую группу под названием математические пакеты;
- MedCalc – статистический пакет для проведения биомедицинских исследований.

СПАРК – система, объединившая в единое целое разрозненные массивы информации об организациях, рассчитавшая скоринги и рейтинги, позволяющие оценить платежеспособность, благонадежность и вероятность банкротства любой организации, имеющейся в базе данных системы. Примеры окна результатов работы системы по оценке организации представлен на рисунках 1.4 и 1.5. Как видно из данного описания, данная система специализируется на классификации субъектов бизнеса по жестко заданным критериям классификации, что значительно сужает область применения данного программного продукта. Кроме того, стоимость лицензии достаточно велика [12].

Математические пакеты обычно позволяют производить широкий круг статистических расчетов, в том числе и проводить бинарную классификацию на основе логистической регрессии. При использовании таких программных средств возможно построение любых моделей по любым критериям классификации, с любым числом и составом регрессоров, используя собственные обучающие выборки. Кроме того, такие средства позволяют оценивать качество построенных моделей по широкому кругу параметров.



Рисунок 1.4 – Окно СПАРК по оценке кредитного риска

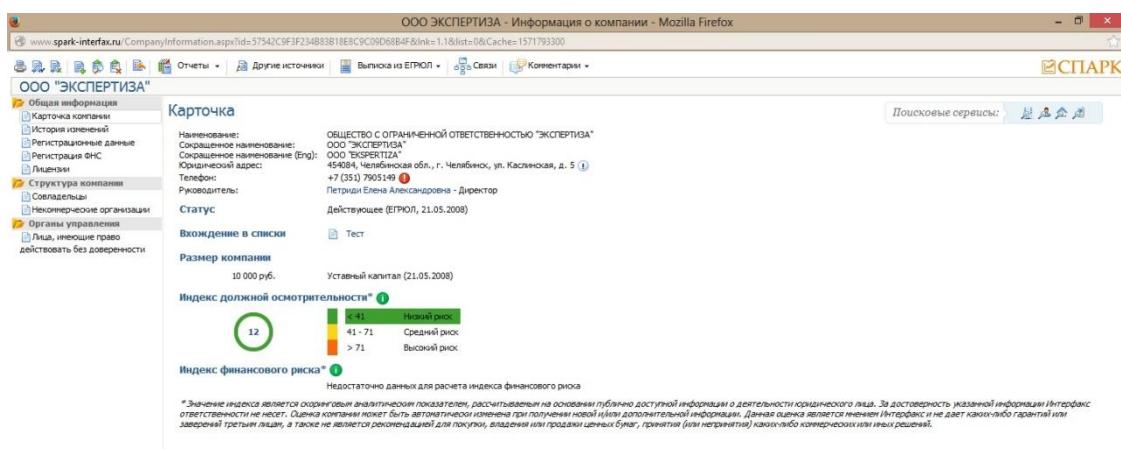


Рисунок 1.5 – Окно СПАРК по оценке финансового риска

Недостатком применения подобных программных продуктов является необходимость наличия специальных знаний в области математической статистики, а также высокая стоимость лицензий на использование. Помимо этого такие инструменты не имеют специализации на решение задач классификации и предлагают слишком широкий набор возможностей, что снижает эффективность их использования при решении задач бинарной классификации [13]. Примеры окон математического пакета STATISTICA для работы с логистической регрессией представлены на рисунках 1.6 и 1.7.

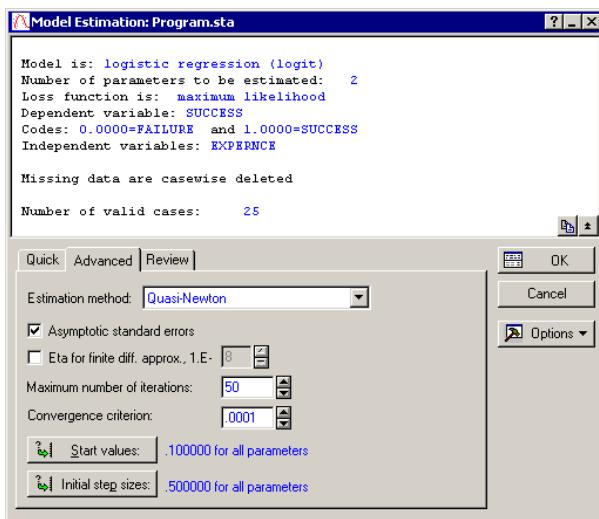


Рисунок 1.6 – Окно STATISTICA с установкой параметров поиска коэффициентов модели

Data: Model: Logistic regression (lo...)		
Model: Logistic regression Dep. var: SUCCESS Final loss: 12.7122870		
N=25	Const.B0	EXPERNCE
Estimate	-3.05970	0.161
Standard Error	1.25959	0.065
t(23)	-2.42912	2.485
p-level	0.02336	0.021
-95%CL	-5.66536	0.027
+95%CL	-0.45403	0.296
Wald's Chi-square	5.90061	6.173
p-level	0.01514	0.013
Odds ratio (unit ch)	0.04690	1.175
-95%CL	0.00346	1.027
+95%CL	0.63506	1.344
Odds ratio (range)		91.983
-95%CL		2.132
+95%CL		3968.996

Рисунок 1.7 – Окно STATISTICA с результатами поиска коэффициентов модели

MedCalc – статистический пакет для Windows, созданный с учетом требований для биомедицинских исследований. Предоставляет широкие возможности по статистическому анализу (регрессии, тесты, корреляции) и графическому представлению данных. Примеры окна системы, предназначенного для работы с логистической регрессией представлен на рисунке 1.8 и 1.9. Недостатком данной системы является специализация в области медицины, что сужает область ее применения, кроме того, являясь настольным приложением, данный пакет не обеспечивает должный уровень обратной связи с моделью [13, 14].

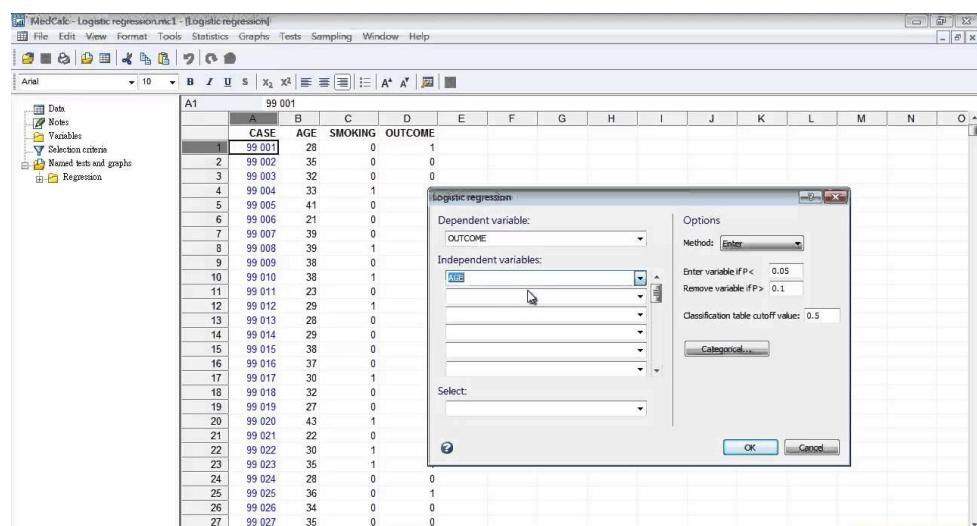


Рисунок 1.8 – Окно MedCalc с настройками построения модели

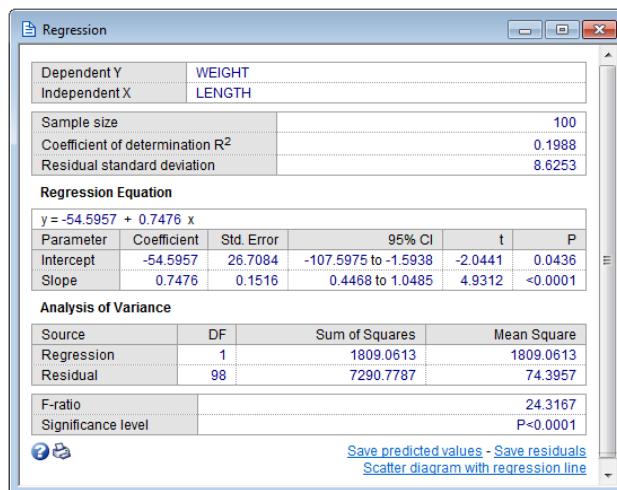


Рисунок 1.9 – Окно MedCalc с результатами построения регрессии

Для сравнения данных программных средств были сформулированы следующие описательные характеристики

- лицензия: важность данной характеристики объясняется тем, что стоимость программного средства часто является одним из решающих факторов при принятии решения о целесообразности использования того или иного программного продукта, что особенно актуально для физических лиц и субъектов малого и среднего бизнеса;

- необходимость наличия специальных знаний в области математической статистики: зачастую заинтересованные в решении задач классификации субъекты являются квалифицированными специалистами своей предметной области, не имея специальных познаний и практического опыта применения методов математической статистики, что создает проблему эффективного использования программного средства для решения классификационной задачи с минимальными затратами времени; кроме того, отсутствие знаний статистических методов может привести к возникновению ошибки, что также снижает эффективность применения программного средства;

- предоставление возможности обучения моделей на собственных выборках: важность данной характеристики связана с тем, что такая возможность позволяет максимально расширить сферу применения программного продукта, а также максимально учитывать специфику предметной области для каждого использующего субъекта; кроме того, такая возможность позволяет своевременно и гибко реагировать на изменения в специфике функционирования моделируемых ситуаций;

- архитектура: выбор клиент-серверной архитектуры позволяет накапливать базу моделей и предоставлять доступ к использованию данных моделей пользователям со всего мира;

- осуществление обратной связи по результатам использования модели: обеспечение такой возможности позволяет оказывать своевременное корректирующее воздействие на модель, повышая ее классифицирующую способность; в совокупности с выбором клиент-серверной архитектуры (большое

количество пользователей совершают больше сеансов обратной связи) такой подход позволяет существенно улучшать качество моделей.

Таким образом, описание программных средств, используемых в том числе и для проведения бинарной классификации на основе логистической регрессии, представлено в таблице 1.3 [12, 13, 14].

Таблица 1.3 – Сравнение программных средств

Программное средства	Характеристика				
	Лицензия	Необходимость специальных знаний	Обучение на собственных выборках	Архитектура	Обратная связь
Математические пакеты	Коммерческие лицензии, крайне высокая стоимость	Нужны знания в области математической статистики и опыт использования математических пакетов	Возможно	Настольные приложения с централизованной архитектурой	Возможно в виде последующего включения в выборку результатов отдельных наблюдений
СПАРК	Коммерческая подписка (от 25 тыс. руб. в месяц)	Возможно использование без специальных знаний	Невозможно	Веб-архитектура	Невозможна
MedCalc	Коммерческая лицензия (от 445\$ - одиночная пожизненная лицензия, до 1780\$ в год за сетевую лицензию)	Необходимы базовые познания в математической статистики	Возможно	Настольное приложения с централизованной архитектурой	Возможно в виде последующего включения в выборку результатов отдельных наблюдений

Как видно из данной таблицы, на настоящий момент нет известных программных средств, которые бы полностью удовлетворяли обозначенным критериям. Такая ситуация делает актуальной задачу разработки автоматизированной информационной системы для бинарной классификации объектов на основе логистической регрессии в виде свободного открытого приложения, на базе веб-архитектуры, позволяющего производить обучение

моделей на собственных выборках и осуществлять обратную связь от всех пользователей системы для повышения качества используемых моделей.

Исходя из характеристики поставленной задачи, становится ясно, что для ее решения необходимо создание программного продукта собственными силами. На начальном этапе необходимо определить и технологию проектирования и средства разработки. В качестве архитектуры было решено использовать клиент-серверный вариант на основе веб-технологий. Такое решение позволит эффективно предоставлять доступ к приложению широкому кругу пользователей, одновременно аккумулируя модели в одном месте и позволяя улучшать существующие модели за счет интенсивного потока обратной связи ото всех пользователей, что представляется крайне затруднительным при файл-серверной архитектуре или при использовании разрозненных клиентов доступа у различных пользователей системы. При использовании веб-технологий установка и функционирование приложения будет осуществляться на одной физической машине, обеспечивая доступность из любой точки мира, что становится возможным, благодаря современному уровню развития информационных технологий. Кроме этого такой подход делает возможным работу с приложением при помощи любого способа доступа – начиная от маломощных персональных компьютеров и заканчивая мобильными устройствами [15].

В качестве языка программирования был выбран язык PHP, что объясняется имеющимися у него преимуществами [16]:

- открытый исходный код – использование данного языка бесплатно и не требует никаких выплат даже при разработке коммерческих проектов;
- кроссплатформенность: PHP стабильно работает практически на всех операционных системах – Windows, Unix, Linux, MacOS;
- простота интеграции: подавляющее большинство веб-серверов и СУБД имеют простые и надежные средства интеграции с PHP;
- большое сообщество разработчиков: такое преимущество означает возможность быстрого решения возникающих при разработке проблем; кроме того

это означает, что существует большое количество библиотек, позволяющих упростить рутинные действия;

- динамичность развития: хотя язык существует уже давно и работает стабильно, работы по его развитию продолжаются и язык получает постоянные обновления и улучшения;
- опыт разработки: имеется опыт разработки коммерческих проектов с использованием данного языка.

В качестве языка разметки был выбран HTML5. Этот объясняется тем, что данный язык и спецификация являются стандартом де-факто при разработке веб-приложений. В настоящее время серьезных альтернатив такому подходу не существует [17]. В качестве языка клиентского программирования был выбран JavaScript. Это также объясняется его монополистическим положением в рассматриваемой области применения. Кроме того, имеется опыт разработки коммерческих систем с использованием данного языка. В качестве языка оформления и форматирования страниц был выбран CSS, так как серьезных альтернатив этому языку не имеется.

В качестве СУБД было решено использовать PostgreSQL – свободную объектно-реляционную систему управления базами данных. Это объясняется следующими ее преимуществами [18]:

- открытый исходный код: использование данной СУБД не связано ни с какими лицензионными отчислениями даже при ее использовании в коммерческих проектах (в отличие, например, от MySQL);
- развитое сообщество и поддержка: имеется сообщество профессионалов и энтузиастов, которое может помочь при решении проблем, возникающих в процессе разработки;
- надежность и стабильность: надежность данной СУБД подтверждена опытом использования в крупных проектах (Cisco, Skype, МГУ);
- кроссплатформенность: PostgreSQL работает на операционных системах Unix, Linux, Windows;

- интегрируемость с языками программирования: существует поддержка работы с данной СУБД из большинства современных языков программирования, в том числе PHP;

- опыт использования: имеется опыт разработки коммерческих систем с использованием СУБД PostgreSQL.

Для повышения эффективности разработки программного средства было принято решение использовать несколько фреймворков. Это позволит сократить время разработки, уменьшив объем выполняемых рутинных операций, таких как организация роутинга системы, построения слоя абстракции работы с базой данных и тому подобных действий.

В качестве PHP-фреймворка был выбран Laravel 4.2, что объясняется его следующими преимуществами [19]:

- открытый исходный код: нет необходимости в выплате лицензионных платежей при разработке даже коммерческих проектов;

- поддержка современных возможностей PHP: данный фреймворк написан при четком следовании тенденциям развития языка и полностью использует все перспективные возможности, предоставляемые PHP (ООП, функции-замыкания, и тому подобное);

- исчерпывающая документация: вся информация, необходимая для изучения фреймворка, доступна в официальной документации и изложена в структурированном и понятном виде;

- активное сообщество: благодаря развивающемуся и активному сообществу возможно быстрое получение дополнительной информации и помощи в решении проблем, возникающих при разработке;

- большое количество компонент: используя менеджер зависимостей PHP (composer), данный фреймворк предоставляет возможность удобного использования готовых компонент из эко-системы PHP для реализации часто используемого функционала;

- опыт разработки: имеется опыт разработки коммерческих проектов с использованием данного фреймворка.

В качестве CSS-фреймворка был выбран Twitter Bootstrap, так как он имеет следующие преимущества [20]:

- открытый исходный код: возможно использование в коммерческих проектах без выплаты лицензионных отчислений;
- адаптивность: предоставляет удобный механизм построения адаптивных веб-страниц, что является де-факто обязательным требованием при разработке современных приложений;
- возможности персонализации: предоставляется удобный механизм персонализации компонентов фреймворка для удовлетворения нужд дизайнера;
- опыт использования: имеется опыт использования данного фреймворка при разработке коммерческих проектов.

В качестве javascript-фреймворка было принято использовать jQuery ввиду имеющихся у него преимуществ [21]:

- открытый исходный код: возможно использование в коммерческих проектах без выплаты лицензионных отчислений;
- скорость и понятность: код, написанный с использованием данного фреймворка, зачастую более читаем, чем код, написанный на чистом javascript;
- кросбраузерность: данный фреймворк корректно работает в подавляющем большинстве веб-браузеров;
- распространность и активное сообщество: данный фреймворк часто является стандартом де-факто при разработке веб-приложений, что объясняет наличие большого сообщества; кроме того, большинству веб-разработчиков код, написанный с использованием данного фреймворка понятен.

1.4 Выводы по главе

В аналитической части дипломного проекта были рассмотрены области применения и методы решения задачи бинарной классификации. Были выделены преимущества и недостатки методов при их применении для решения рассматриваемой задачи. В результате анализа было принято решение рассмотреть в

рамках данной работы только метод логистической регрессии, обладающий технологической простотой и простотой применения и обучения модели. Поэтому были рассмотрены теоретические аспекты логистической регрессии, а также технологические особенности ее применения для задач бинарной классификации. Помимо этого были описаны инструменты, позволяющие обеспечивать качество моделей логистической регрессии с точки зрения количественного и качественного состава ее регрессоров, а также инструменты обеспечения адаптируемости модели к изменяющимся условиям действительности с точки зрения изменения коэффициентов модели. Кроме того, был проведен анализ областей применения логистической регрессии и существующих разработок в данной области. В результате анализа было определено, что на настоящий момент нет известных программных средств, удовлетворяющих требованиям, актуальным для проведения бинарной классификации объектах в различных предметных областях.

Таким образом, было определена объективная необходимость в разработке программного средства, автоматизирующего процесс бинарной классификации объектов на основе логистической регрессии. В связи с этим были обоснованы решения по выбору технологии проектирования такой системы и средств разработки.

2 Проектная часть

2.1 Формирование функциональных возможностей разрабатываемой информационной системы

Цель разрабатываемой информационной системы – автоматизация процесса бинарной классификации объектов. В рамках данной цели основной задачей является обеспечение автоматизированной возможности отнесения изучаемого объекта к одному из двух рассматриваемых в рамках решаемой задачи классов.

Для формирования функциональных возможностей были выделены следующие категории пользователей информационной системы:

- гость (пользователь, не авторизовавшийся в приложении);
- пользователь (пользователь, авторизовавшийся в приложении и обладающий ролью «пользователь»);
- администратор (пользователь, авторизовавшийся в приложении и обладающий ролью «администратор»).

С точки зрения гостя, система будет обладать следующим функционалом:

- предоставление информации о системе;
- возможность зарегистрироваться в системе;
- возможность войти в систему в качестве зарегистрированного пользователя.

С точки зрения пользователя функционал разрабатываемой системы будет следующим:

- возможность выйти (разавторизоваться) из приложения;
- предоставление информации о системе;
- напоминание о задачах, ждущих обратной связи, целью которой является обеспечение адаптируемости модели к изменяющимся условиям действительности, и задачах, срок подтверждения для которых истек;
- предоставление списка задач, для которых требуется обратная связь;
- восстановление пароля через электронную почту;
- просмотр списка проблемных ситуаций;
- просмотр и выбор из списка задач классификации;

- просмотр информации о решаемых задачах классификации;
 - решение выбранной задачи классификации;
 - осуществление обратной связи (ввод реального исхода события);
- поиск в приложении проблемной ситуации по коду ОКВЭД (общероссийский классификатор видов экономической деятельности, был использован для того, чтобы упростить систематизацию поиск нужной проблемы среди всего многообразия проблемных ситуаций), названию, а также поиск решаемой задачи по ее названию.

Администратор имеет доступ ко всем функциональным возможностям пользователя, кроме того, с его точки зрения будет доступен дополнительный функционал:

- добавление проблемных ситуаций в каталог проблемных ситуаций;
- удаление из каталога проблемных ситуаций;
- редактирование проблемных ситуаций;
- просмотр списка задач, недоступных для использования по критерию значения специфичности и чувствительности в точке порога отсечения;
- добавление решаемых задач, в том числе и обучение модели задачи;
- скачивание основной обучающей выборки решаемой задачи;
- скачивание шаблона для создания обучающих выборок;
- получение уведомлений о деактивизации моделей по электронной почте (деактивированные модели не видны пользователю в списке решаемых задач и недоступны для использования);
- просмотр информации о модели – значение чувствительности и специфичности в пороге отсечения, площадь под ROC-кривой, время корректности решения (до обратной связи), регрессоры, единицы измерения регрессоров, коэффициенты модели, коэффициенты эластичности и стандартизованные коэффициенты.

2.2 Информационное обеспечение задачи автоматизации

Входной информацией для разрабатываемой системы является, в основном, информация, необходимая для обучения моделей, используемых для решения задач классификации. Состав данной информации зависит от предметной области решаемой задачи, но можно обобщенно выделить следующие необходимые для всех предметных областей компоненты:

- время корректности решения задачи (время, в течение которого актуален результат решения задачи и после истечения которого пользователю будет закрыта возможность использовать функционал решения задач, если он не совершил обратную связь);
- минимальное значение чувствительности и специфичности в пороге отсечения (выбор значения показателя зависит от требуемой от модели надежности в рамках решаемой задачи);
- обучающая выборка (набор объектов с их характеристиками, для которых априорно известно, к какому из двух классов они принадлежат).

Перечень входных документов, на основании которых собираются перечисленные компоненты, напрямую зависят от предметной области и не могут быть описаны стандартизовано. Например, для проведения задач классификации отделом применения системы управления рисками Федеральной таможенной службы Российской Федерации, основным источником входных данных является Центральный реестр субъектов внешнеэкономической деятельности (ЦРСВЭД). Посредством ЦРСВЭД осуществляется сбор и хранение согласованных сведений о субъектах, когда-либо вступавших в правоотношения с таможенными органами Российской Федерации, а также предоставление подразделениям таможенных органов объективной информации о деятельности субъектов внешнеэкономической деятельности. Кроме того, в качестве источника данных используется также центральный реестр товаров, который обеспечивает доступ к эталонным сведениям о товарах, перемещаемых через государственную границу, а также предоставляет подразделениям таможенных органов РФ сведения для проверки информации, заявляемой участниками внешнеэкономической деятельности в таможенных

документах, контроля применения к данным товарам мер экономического регулирования, управления рисками и других таможенных целей [5].

Таким образом, на данном этапе необходимо произвести моделирование данных. Первый этап такого моделирования – моделирование на логическом уровне. Для этого будет произведено построение логической модели данных (IDEF1X) с помощью CASE-средства ErWin. Логическая модель отражает логические связи между элементами данных вне зависимости от их содержания и среды хранения. Логическая модель может быть трех типов: реляционная, иерархическая и сетевая. В данном случае будет использоваться реляционная модель данных [22]. В таблице 2.1 представлены сущности и связи между сущностями.

Таблица 2.1 – Сущности и связи логической модели данных

Сущности	Связи между сущностями
Пользователи (ключ – код пользователя)	- пользователь обладает одной или несколькими ролями
Роли (ключ – код роли)	- проблемная ситуация является наследником другой проблемной ситуации или является ситуацией верхнего уровня;
Проблемные ситуации (ключ – код проблемной ситуации)	- проблемная ситуация содержит решаемые задачи;
Решаемые задачи (ключ – код решаемой задачи)	- решаемая задача обладает сроком корректности решения;
Сроки корректности решения (ключ – код срока корректности решения)	- задачи решаются пользователями.
Решенные пользователем задачи (ключ – код решенной пользователем задачи)	

Логическая модель данных для разрабатываемого приложения представлена на рисунке 2.1.



Рисунок 2.1 – Логическая модель данных

Следующим этапом моделирования данных является построение физической модели. Физическая модель, определяющая размещение данных, методы доступа и технику индексирования, называется внутренней моделью системы. Физическая модель данных для выбранной СУБД (PostgreSQL), построенная по стандарту IDEF1X изображена на рисунке 2.2. Для увеличения производительности системы и более логичного ее представления необходимо применить денормализацию [22].

Следует отметить, что имеющееся некоторое расхождение между существующими в логической модели данных и таблицами в физической, вызвано процессом денормализации данных [22].

База данных разрабатываемого программного продукта состоит из 7 таблиц, каждая из которых отличается своей структурой: количеством столбцов (атрибутов) и их логическими типами. Сама база данных разрабатывается в СУБД PostgreSQL.

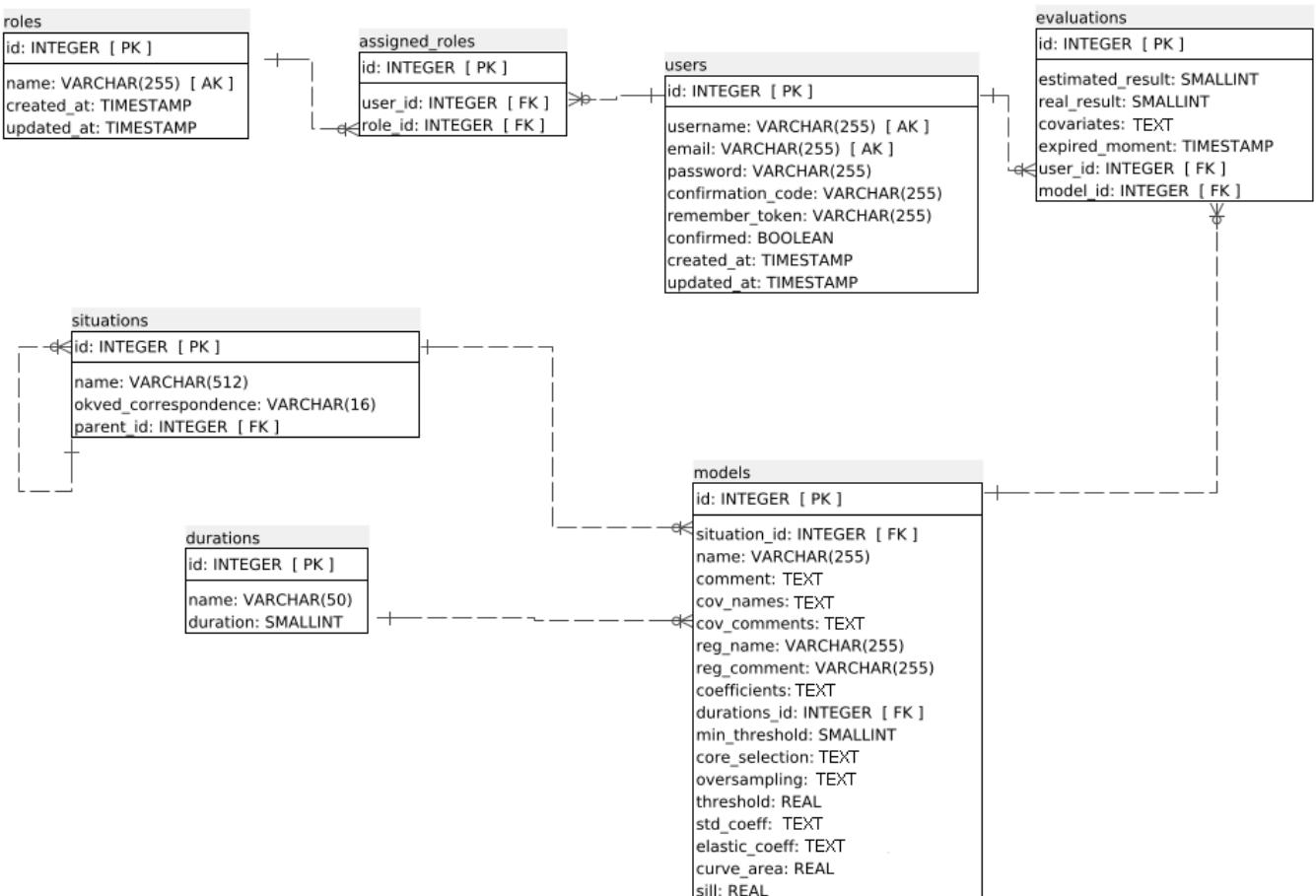


Рисунок 2.2 – Физическая модель базы данных

Для реализации автоинкрементных полей были использованы последовательности, имена которых состоят из названия таблицы и суффикса `id_seq` (`id` – название поля, для которого они используются, `seq` – сокращение от `sequence` – последовательность). Поэтому для полей, где необходимо использование автоинкремента, значения по умолчанию будут задаваться как `«nextval(<имя_последовательности>::regclass)»`. Список последовательностей представлен на рисунке 2.3.

Последовательность
<code>assigned_roles_id_seq</code>
<code>durations_id_seq</code>
<code>evaluations_id_seq</code>
<code>models_id_seq</code>
<code>permission_role_id_seq</code>
<code>permissions_id_seq</code>
<code>roles_id_seq</code>
<code>situations_id_seq</code>
<code>users_id_seq</code>

Рисунок 2.3 – Список используемых последовательностей

Таблица `assigned_roles` (рисунок 2.4) хранит информацию об имеющихся у пользователей ролях: ключ, код пользователя (внешний ключ), код роли (внешний ключ).

Column	Тип данных	Not Null	По умолчанию	Ограничения
<code>id</code>	<code>integer</code>	NOT NULL	<code>nextval('assigned_roles_id_seq'::regclass)</code>	
<code>user_id</code>	<code>integer</code>	NOT NULL		
<code>role_id</code>	<code>integer</code>	NOT NULL		

Рисунок 2.4 – Таблица `assigned_roles`

Таблица `durations` (рисунок 2.5) используется для хранения информации о сроках корректности решения: ключ, название срока и продолжительность срока в часах.

Таблица `evaluations` (рисунок 2.6) содержит информацию о решенных пользователями задачах: ключ, вычисленный моделью результат, реальный результат, значения регрессоров, время истечения корректности решения, код

пользователя (внешний ключ) и код модели (внешний ключ).

Column	Тип данных	Not Null	По умолчанию	Ограничения
<code>id</code>	integer	NOT NULL	<code>nextval('durations_id_seq'::regclass)</code>	○ ↗
<code>name</code>	character varying(50)	NOT NULL		
<code>duration</code>	smallint	NOT NULL		

Рисунок 2.5 – Таблица durations

Column	Тип данных	Not Null	По умолчанию	Ограничения
<code>id</code>	integer	NOT NULL	<code>nextval('evaluations_id_seq'::regclass)</code>	○ ↗
<code>estimated_result</code>	smallint	NOT NULL		
<code>real_result</code>	smallint			
<code>covariates</code>	text	NOT NULL		
<code>expired_moment</code>	timestamp without time zone	NOT NULL		
<code>user_id</code>	integer	NOT NULL		○ ↗
<code>model_id</code>	integer	NOT NULL		○ ↗

Рисунок 2.6 – Таблица evaluations

Таблица models (рисунок 2.8) используется для хранения информации о решаемых задачах: ключ, код проблемной ситуации (внешний ключ), название, дополнительная информация, названия регрессоров, единицы измерения регрессоров, название регрессии, единицы измерения регрессии, коэффициенты, код времени корректности решения (внешний ключ), минимальное значение чувствительности и специфичности в пороге отсечения, основная обучающая выборка, дополнительная обучающая выборка, реальные значение чувствительности и специфичности в пороге отсечения, стандартизованные коэффициенты, частные коэффициенты эластичности, площадь под ROC-кривой, порог отсечения.

Column	Тип данных	Not Null	По умолчанию	Ограничения
<code>id</code>	integer	NOT NULL	<code>nextval('models_id_seq'::regclass)</code>	○ ↗
<code>situation_id</code>	integer	NOT NULL		○ ↗
<code>name</code>	character varying(255)	NOT NULL		
<code>comment</code>	text			
<code>cov_names</code>	text	NOT NULL		
<code>cov_comments</code>	text	NOT NULL		
<code>reg_name</code>	character varying(255)	NOT NULL		
<code>reg_comment</code>	character varying(255)	NOT NULL		
<code>coefficients</code>	text	NOT NULL		
<code>durations_id</code>	smallint	NOT NULL		○ ↗
<code>min_threshold</code>	smallint	NOT NULL		
<code>core_selection</code>	text	NOT NULL		
<code>oversampling</code>	text			
<code>threshold</code>	real	NOT NULL		
<code>std_coeff</code>	text	NOT NULL		
<code>elastic_coeff</code>	text	NOT NULL		
<code>curve_area</code>	real	NOT NULL		
<code>sill</code>	real	NOT NULL		

Рисунок 2.8 – Таблица models

Таблица roles (рисунок 2.10) служит для хранения информации о ролях: ключ, название роли, время создания и изменения.

Column	Тип данных	Not Null	По умолчанию	Ограничения
<code>id</code>	integer	NOT NULL	<code>nextval('roles_id_seq'::regclass)</code>	ключ
<code>name</code>	character varying(255)	NOT NULL		1
<code>created_at</code>	timestamp without time zone	NOT NULL		
<code>updated_at</code>	timestamp without time zone	NOT NULL		

Рисунок 2.10 – Таблица roles

Таблица situations (рисунок 2.11) служит для хранения информации о проблемных ситуациях: ключ, название проблемной ситуации, соответствие ОКВЭД и код родительской ситуации (внешний ключ).

Column	Тип данных	Not Null	По умолчанию	Ограничения
<code>id</code>	integer	NOT NULL	<code>nextval('situations_id_seq'::regclass)</code>	ключ
<code>name</code>	character varying(512)	NOT NULL		
<code>okved_correspondence</code>	character varying(16)	NOT NULL	"::character varying	
<code>parent_id</code>	integer			ключ

Рисунок 2.11 – Таблица situations

Таблица users (рисунок 2.12) служит для хранения информации о пользователях системы: ключ, имя пользователя, E-mail, пароль, код подтверждения, токен запоминания, флагок подтверждения регистрации время создания и время изменения записи.

Column	Тип данных	Not Null	По умолчанию	Ограничения
<code>id</code>	integer	NOT NULL	<code>nextval('users_id_seq'::regclass)</code>	ключ
<code>username</code>	character varying(255)	NOT NULL		1
<code>email</code>	character varying(255)	NOT NULL		1
<code>password</code>	character varying(255)	NOT NULL		
<code>confirmation_code</code>	character varying(255)	NOT NULL		
<code>remember_token</code>	character varying(255)			
<code>confirmed</code>	boolean	NOT NULL	false	
<code>created_at</code>	timestamp without time zone	NOT NULL		
<code>updated_at</code>	timestamp without time zone	NOT NULL		

Рисунок 2.12 – Таблица users

Кроме того, для работы системы необходимо две служебные таблицы.

Таблица migrations , хранящая информацию о проведенных миграциях базы данных: имя миграции и порядковый номер. Такой подход используется для обеспечения возможности версионного контроля состояния базы данных. Таблица password_reminders также является служебной и используется для временного хранения информации, необходимой для восстановления пароля пользователем: ключ, E-mail, токен (подтверждение аутентичности) и время создания записи.

2.3 Алгоритмическое и программное обеспечение задачи автоматизации

Наибольший интерес с точки зрения алгоритмической сложности в рамках разрабатываемого программного средства представляет алгоритм обучения модели логистической регрессии. Коэффициенты линейной регрессии могут быть получены с помощью метода наименьших квадратов (МНК). Для модели логистической регрессии таких решений не существует. Оптимальным методом оценки коэффициентов в таком случае является метод максимального правдоподобия. Суть данного метода состоит в поиске таких значений коэффициентов, для которых максимальна вероятность появления [1].

Для более подробного рассмотрения данного метода необходимо рассмотреть функцию правдоподобия. Данная функция определяет вероятность появления значений параметров $\beta_1, \beta_2 \dots \beta_n$ для заданного значения x. Тогда задача заключается в поиске значений этих параметров, максимизирующих функцию правдоподобия. Для этого строятся оценки максимального правдоподобия, для которых значения параметров наиболее подходят для наблюдаемых данных. Функция правдоподобия имеет вид, представленный в формуле (2.1).

$$L(Y_1, Y_2, \dots, Y_k, \theta) = p(Y_1, \theta) * \dots * p(Y_k, \theta) \quad (2.1)$$

где Y_k – значение выходной переменной в k-ом наблюдении, θ – оцениваемые коэффициенты [9].

Решение задачи упрощается, если максимизировать натуральной логарифм от

рассматриваемой функции. Это становится возможным, так как обе эти функции достигают своего максимума при одном значении θ [7]. Вид логарифма функции правдоподобия представлен в формуле (2.2).

$$L^* = \sum_{i \in I_1} \ln P_i(W) + \sum_{i \in I_0} \ln(1 - P_i(W)) = \sum_{i=1}^k [Y_i \ln P_i(W) + (1 - Y_i) \ln(1 - P_i(W))], \quad (2.2)$$

где P_i – вероятность появления единицы, W – вектор коэффициентов регрессии, I_0 , I_1 – множества наблюдений, для которых $Y_i = 0$ и $Y_i = 1$ соответственно [9].

Градиент функции правдоподобия представлен на формуле (2.3).

$$g = \sum_i (Y_i - P_i) X_i, \quad (2.3)$$

где X_i – строка матрицы регрессоров [9].

Гессиан функции правдоподобия представлен в формуле (2.4).

$$H = - \sum_i P_i (1 - P_i) X_i^T X_i, \quad (2.4)$$

где X_i^T – транспонированная строка матрицы регрессоров [9].

Гессиан функции правдоподобия всюду отрицательно определенный, поэтому логарифмическая функция правдоподобия всюду вогнута. Такое поведение Гессиана позволяет использовать для поиска максимума метод Ньютона-Рафсона, достаточное условие сходимости которого при этом выполняется [23]. Итерационная формула для использования данного метода представлена в формуле (2.5).

$$b_t = b_{t-1} + (X' V_{t-1} X)^{-1} X' (y - p_{t-1}), \quad (2.5)$$

где t – номер итерации, b_t – вектор коэффициентов на итерации t , X – матрица регрессоров, y – вектор значений регрессии из обучающей выборки, p_{t-1} – вектор

значений регрессии, вычисленных моделью с коэффициентами b_{t-1} , V_{t-1} – диагональная матрица со значениями по диагонали, вычисленными по формуле (2.6) [23].

$$V_{t-1}^{i,i} = p_{i,t-1}(1 - p_{i,t-1}). \quad (2.6)$$

Итерации необходимо продолжать до тех пор, пока b_t не станет отличаться от b_{t-1} на слишком малую величину ε (целесообразно принять равным 0.001). Кроме того, целесообразно останавливать итерации в случае, когда СКО между вычисленными моделью и реальными значениями функции регрессии становятся больше n -раз подряд (n целесообразно принять равным 4) [7]. Также целесообразно остановить вычисления, если b_t больше b_{t-1} на некоторую большую величину M (целесообразно принять равным 1000) [7]. Таким образом, блок-схема алгоритма обучения модели логистической регрессии представлена на рисунке А.1 в приложении А. Кроме того, с точки зрения алгоритмической сложности также интересен алгоритм поиска порога отсечения модели. Блок-схема данного алгоритма представлена на рисунке А.2 в приложении А.

Информационная структура приложения, отражающая способ организации информационных материалов и связей между ними, позволяющих материалам взаимодействовать друг с другом, представлена на рисунке А.3 в приложении А.

Диаграмма классов, отражающая классы системы, их атрибуты, методы и связи между ними, представлена на рисунке А.4 в приложении А. Данная диаграмма построена с точки зрения реализации и содержит классы, используемые непосредственно в программном коде [24].

2.4 Тестирование разработанной информационной системы

В ходе тестирования программы применялись две методики [7, 8]. Первая из них – методика «черного ящика». Такое тестирование имеет целью выяснение обстоятельств, в которых поведение программы не соответствует спецификации.

Тестовые данные в таком случае используются только в соответствии со спецификацией программы (без учета знаний о ее внутренней структуре) [25].

Для проведения тестирования по данной методики были использованы фиктивные данные, они были внесены в информационную систему в достаточном объеме. При проверке корректности работы результаты были признаны корректными. Результаты тестирования были признаны удовлетворительными, окончательное тестирование по методике «черного ящика» показало адекватность результатов работы программы эталонным [26].

В качестве второй методики была использована методика «белого ящика» - стратегия тестирования, управляемого логикой программы, позволяющая исследовать внутреннюю структуру программы. В этом случае тестирующий получает тестовые данные путем анализа логики программы (при этом не стоит забывать использовать спецификации программы) [25]. Были проверены основные маршруты потока передачи управления. Была проведена проверка на обработку некорректных (с точки зрения спецификации) входных данных.

Также осуществлялась проверка работы программы в нормальном, исключительном и экстремальном режимах. Для проведения тестирования в перечисленных режимах работы было решено разделить тестирование клиентской и административной части.

Для тестирования административной части в нормальном режиме работы была добавлена новая проблемная ситуация. Для этого был открыт каталог проблемных ситуаций, нажата кнопка «Добавить ситуацию». Были заполнены поля «Название» и «Соответствие ОКВЭД», после чего была нажата кнопка «Добавить». Результат этого действия представлен на рисунке 2.13. Кроме того была добавлена решаемая задача. Для этого был открыт список решаемых задач, нажата кнопка «Добавить задачу». Были заполнены необходимые поля, а также загружен файл обучающей выборки. После этого была нажата кнопка «Добавить». Результат этого действия представлен на рисунках 2.14 и 2.15.

The screenshot shows the dichotomy application interface. At the top, there is a navigation bar with links for 'dichotomy', 'Главная' (Home), 'Управление' (Management), 'Решение задач классификации' (Classification Task Solving), 'Вы вошли как admin' (You logged in as admin), 'Выход' (Logout), and 'Поиск' (Search). A red notification box at the top left states: 'У Вас имеются задачи, срок подтверждения которых истек.' (You have tasks whose confirmation deadline has expired.) Below it, another message says: 'Возможность решения задач классификации временно заблокирована для Вас до момента, пока Вы не осуществите обратную связь.' (The ability to solve classification tasks is temporarily blocked for you until you provide feedback.) A green success message at the bottom left says: 'Запись успешно добавлена!' (Record added successfully!).

▼ Каталог проблемных ситуаций

Главная / Каталог проблемных ситуаций				
Добавить ситуацию				
Название	Соответствие ОКВЭД	Редактировать реквизиты	Удалить	Решаемые задачи
Сельское, лесное хозяйство, охота, рыболовство и рыбоводство	Раздел А.			
Добыча полезных ископаемых	Раздел В.			
Обрабатывающие производства	Раздел С.			
Обеспечение электрической энергией, газом и паром; кондиционирование воздуха	Раздел D.			

Рисунок 2.13 – Результат добавления новой проблемной ситуации

The screenshot shows the dichotomy application interface. At the top, there is a navigation bar with links for 'dichotomy', 'Главная' (Home), 'Управление' (Management), 'Решение задач классификации' (Classification Task Solving), 'Вы вошли как admin' (You logged in as admin), 'Выход' (Logout), and 'Поиск' (Search). A red notification box at the top left states: 'У Вас имеются задачи, срок подтверждения которых истек.' (You have tasks whose confirmation deadline has expired.) Below it, another message says: 'Возможность решения задач классификации временно заблокирована для Вас до момента, пока Вы не осуществите обратную связь.' (The ability to solve classification tasks is temporarily blocked for you until you provide feedback.) A green success message at the bottom left says: 'Запись успешно добавлена!' (Record added successfully!).

Below the navigation bar, a breadcrumb trail shows: Главная / Каталог решаемых задач / Сельское, лесное хозяйство, охота, рыболовство и рыбоводство

A blue 'Добавить задачу' (Add task) button is visible. The main area displays a table with two rows:

Название	Редактирование параметров	Удалить задачу
адssssssssssss		
Тестовая задача		

Рисунок 2.14 – Сообщение об успешном добавлении новой задачи

The screenshot shows the dichotomy application interface. At the top, there is a navigation bar with links for 'dichotomy', 'Главная' (Home), 'Управление' (Management), 'Решение задач классификации' (Classification Task Solving), 'Вы вошли как admin' (You logged in as admin), 'Выход' (Logout), and 'Поиск' (Search). A red notification box at the top left states: 'У Вас имеются задачи, срок подтверждения которых истек.' (You have tasks whose confirmation deadline has expired.) Below it, another message says: 'Возможность решения задач классификации временно заблокирована для Вас до момента, пока Вы не осуществите обратную связь.' (The ability to solve classification tasks is temporarily blocked for you until you provide feedback.)

Below the navigation bar, a breadcrumb trail shows: Главная / Каталог решаемых задач / Сельское, лесное хозяйство, охота, рыболовство и рыбоводство / Тестовая задача

A blue 'Скачать основную обучающую выборку' (Download main training sample) button is visible. The main area displays a table with three rows:

Тестовая задача	
Информация	Результат: Благонадежность клиента (1 - Да 0 - Нет)
Минимальный порог отсечения 	0.5376
Фактический порог отсечения 	0.5376
Площадь под кривой 	0.5376
Время корректности решения 	0.5376

Below this, a table shows various parameters and their values:

Название	Свободный член	Возраст клиента	Пол	Состоит ли в браке	Количество измерений	Годовой доход	Опыт работы	Срок проживания в регионе	Недвижимость	Месячный платеж
Единицы измерения	-	Количество полных лет	1-Мужской; 0-Женский.	1-Да; 0-Нет.	Количество человек	Денежные единицы (тысяч \$)	Лет (с любой точностью)	Количество целых лет	Денежные единицы (тысяч \$)	Денежные единицы (\$)
Коэффициенты	-3.4914515024051	-0.0030510402472793	0.65942661073325	-0.23531195031044	-1.876481029739	0.00071536612486261	0.003181003648114	0.0093942440654338	0.010803480506039	-0.00089148256199528
% эластичные коэффициенты	-0.2136171319611	0.87009535425989	-0.28237434037253	-3.978905693671	19.319265136892	0.079612731099278	0.25541099372276	0.33202406713056	-7.3738125019921	
Стандартизованные коэффициенты	-0.043645269449346	0.63039195937587	-0.23146718475229	-3.5662769905188	10.709561148451	0.0336047715656	0.20168808038407	0.42016338979197	-2.9798422824176	

Рисунок 2.15 – Результат добавления новой задачи

Как видно из данного рисунка, работа административной части в нормальном режиме функционирования проходит корректно. Для тестирования административной части в исключительном режиме работы были выделены следующие примеры исключительных ситуаций:

- оставление обязательных полей формы незаполненными;
- загрузка файла обучающей выборки, не соответствующего спецификации (например, количество наблюдений меньше чем 10^*n , где n – количество регрессоров);
- попытка перейти в административную часть, авторизовавшись под пользователем, не имеющим роли администратора.

Результат моделирования первой ситуации представлен на рисунке 2.16.

A screenshot of a web application interface. At the top, there is a navigation bar with links: Главная / Каталог проблемных ситуаций / Добавление ситуации. Below this is a form titled 'Добавление ситуации'. It contains two input fields: 'Название' (Name) and 'Соответствие ОКВЭД' (Correspondence to OKVED). Both fields are currently empty. At the bottom of the form are two buttons: 'Добавить' (Add) in blue and 'Отмена' (Cancel) in grey. A red message box at the bottom of the page states: 'Поле Название обязательно для заполнения...' (The Name field is mandatory).

Рисунок 2.16 – Оставление пустым обязательного поля Название

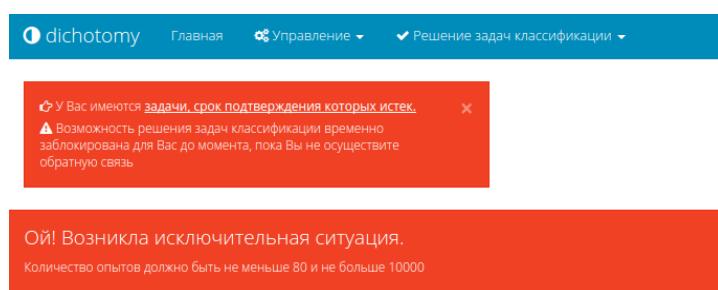


Рисунок 2.17 – Загрузка файла выборки с недостаточным числом наблюдений

Попытка перейти в административную часть под пользователем без роли администратора ведет к переадресации на главную страницу приложения, что

нельзя продемонстрировать на рисунке.

Как видно из представленных рисунков и описания ситуаций, тестируемая система работает корректно при исключительных режимах работы. Это достигнуто за счет валидации ввода на стороне клиента (средствами HTML5) и сервера (средствами PHP), кроме того, была внедрена система выброса и перехвата исключений.

Для тестирования пользовательской части в нормальном режиме было произведено решение тестовой задачи (определение благонадежности заемщика). Для этого была выбрана необходимая задача, были заполнены значения параметров и нажата кнопка «Решить». Результат этого действия представлен на рисунке 2.18.

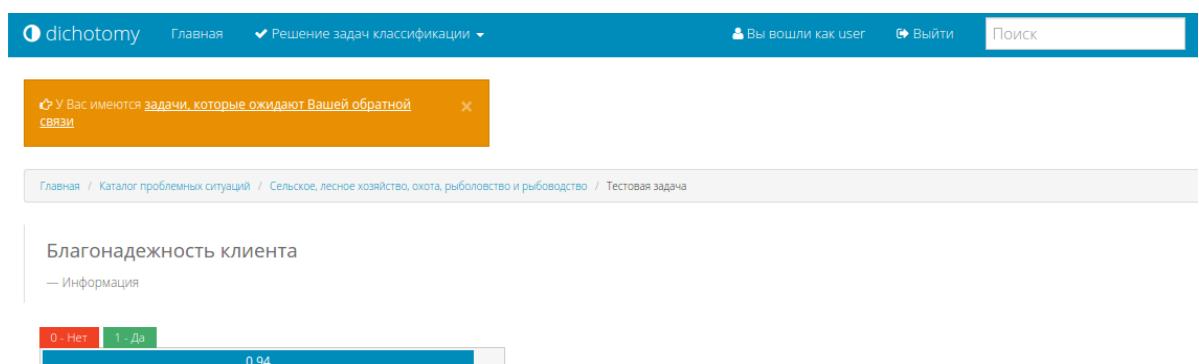


Рисунок 2.18 – Результат решения задачи

Кроме того, была проведена обратная связь. Для этого была выбрано решение, ждущее обратной связи, был введен фиктивный исход данной задачи. Результат данного действия представлен на рисунке 2.19.

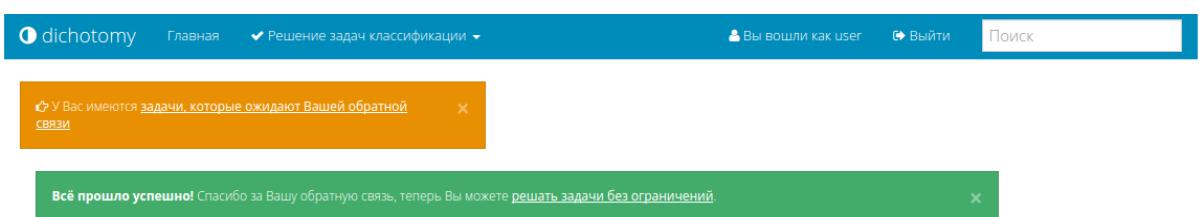


Рисунок 2.19 – Результат совершения обратной связи

Также был протестирован поиск по системе. Для этого в строку поиска была

введена фраза «деятельность» и нажата кнопка Enter. Результат данного действия представлен на рисунке 2.20.

The screenshot shows a search interface with three main sections:

- Совпадения по коду ОКВЭД**: A red box displays the message "Совпадений по коду ОКВЭД не найдено." (No matches found for OKVED code).
- Совпадения по названию проблемной ситуации**: A table lists various business activities with their corresponding OKVED codes:

Лесоводство и прочая лесохозяйственная деятельность	Код ОКВЭД 02.1
Лесоводство и прочая лесохозяйственная деятельность	Код ОКВЭД 02.10
Водоснабжение, водоводческие, организация сбора и утилизации отходов, деятельность по ликвидации загрязнений	Раздел Е
Складское хозяйство и вспомогательная транспортная деятельность	Код ОКВЭД 52
Деятельность почтовой связи и курьерская деятельность	Код ОКВЭД 53
Деятельность почтовой связи прочая и курьерская деятельность	Код ОКВЭД 53.2
Деятельность почтовой связи прочая и курьерская деятельность	Код ОКВЭД/53.20
Деятельность по обработке данных, предоставление услуг по размещению информации, деятельность порталов в информационно-коммуникационной сети Интернет	Код ОКВЭД 63.1
Деятельность по обработке данных, предоставление услуг по размещению информации и связанная с этим деятельность	Код ОКВЭД/63.11
Страхование, перестрахование, деятельность негосударственных пенсионных фондов, кроме обязательного социального обеспечения	Код ОКВЭД/65
- Здесь показаны не все совпадения по названию проблемной ситуации, их слишком много.** (Here are not all the matches for the problem situation name, there are too many.)
- Совпадения по названию решаемой проблемы**: A red box displays the message "Совпадений по названию решаемой проблемы не найдено." (No matches found for the name of the solved problem).

Рисунок 2.20 – Результаты поиска по системе

Как видно из данных рисунков, пользовательский функционал в нормальном режиме работы выполняется корректно.

Для тестирования пользовательской части в исключительном режиме работы были выделены следующие примеры исключительных ситуаций:

- оставление обязательных полей формы незаполненными;
- попытка перейти по несуществующему адресу.

Результат моделирования первой ситуации представлен на рисунке 2.21.

The screenshot shows a form with several input fields, each with an error message indicating it is required:

- Поле Возраст клиента обязательно для заполнения..
- Поле Пол обязательно для заполнения..
- Поле Состоит ли в браке обязательно для заполнения..
- Поле Количество иждивенцев обязательно для заполнения..
- Поле Годовой доход обязательно для заполнения..
- Поле Опыт работы обязательно для заполнения..
- Поле Срок проживания в регионе обязательно для заполнения..
- Поле Недвижимость обязательно для заполнения..
- Поле Месячный платеж обязательно для заполнения..

Рисунок 2.21 – Оставление обязательных полей пустыми

Результат моделирования второй ситуации представлен на рисунке 2.22.



Рисунок 2.22 – Результат перехода по несуществующему адресу

Как видно из данных рисунков, система корректно обрабатывает все исключительные ситуации.

Также были учтены и проработаны ситуации, связанные с экстремальным режимом работы: невозможность подключиться к базе данных, невозможность выполнить запрос, а также все исключения, которые по какой-либо причине не были перехвачены приложением. В данном случае пользователю представляется информация о возникновении ошибки с предложением связаться с администратором по электронной почте. Результат обработки подобных ситуаций представлен на рисунке 2.23.

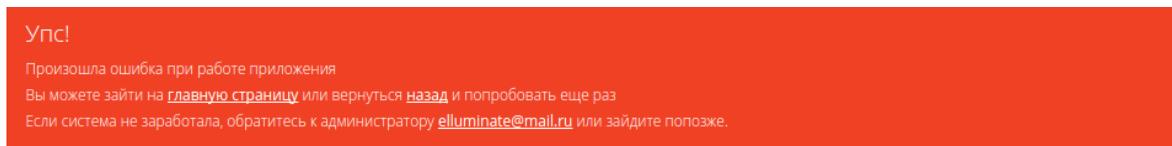


Рисунок 2.23 – Обработка экстремальной ситуации

Как видно из данного рисунка, приложение адекватно обрабатывает экстремальную ситуацию.

Таким образом, был сделан вывод о работоспособности системы во всех режимах работы, что положительно характеризует ее качество.

2.5 Выводы по главе

Во второй части дипломного проекта было подробно описано проектирование автоматизированной информационной системы для бинарной классификации объектов на основе логистической регрессии.

Сначала были сформированы и описаны функциональные требования к проектируемой системе с точки зрения всех групп пользователей – гостя, зарегистрированного пользователя и администратора системы. Далее было рассмотрено информационное обеспечение задачи автоматизации: были описаны концептуальные характеристики входной и выходной информации, приведен пример для одной из возможных областей применения системы. Была разработана логическая и физическая модели базы данных информационной системы, описаны особенности реализации структуры базы данных в объектно-реляционной СУБД PostgreSQL.

После этого было описано алгоритмическое обеспечение информационной системы. Здесь были описаны наиболее интересные с точки зрения алгоритмической сложности алгоритмы системы – метод максимального правдоподобия и поиск порога отсечения модели. Был рассмотрен математический аппарат данных алгоритмов и построены их блок-схемы. Затем было рассмотрено программное обеспечение задачи автоматизации – была описана информационная структура приложения и составлена диаграмма классов системы с точки зрения ее реализации.

Кроме того было проведено тестирование системы, в результате которого были исправлены возникающие в процессе функционирования ошибки. По результатам тестирования был сделан вывод о корректной работе разработанной системы во всех режимах функционирования как в административной, так и в пользовательской части, что положительно характеризует качество информационной системы.

3 Методика внедрения и оценка эффективности проекта

3.1 Формирование технологической среды информационной системы

Разработанная автоматизированная информационная система поддерживает многопользовательский режим работы в сетевой среде. Для обеспечения требований безопасности было реализовано разделение ролей пользователей. Поэтому целесообразно выделить категории пользователей системы:

- гость (посетитель, не авторизовавшийся в приложении);
- зарегистрированный пользователь (посетитель, авторизовавшийся в приложении, и обладающий ролью «Пользователь»);
- администратор системы (посетитель, авторизовавшийся в приложении, и обладающий ролью «Администратор»).

Гость имеет возможность ознакомиться с информацией о системе и выполнить авторизацию. Пользователь может просматривать каталог проблемных ситуаций и списки решаемых задач, решать задачи и совершать обратную связь. Администратор обладает всеми полномочиями пользователя и, кроме того, может добавлять проблемные ситуации, решаемые задачи, просматривать список неактивных задач.

В виду того, что использовать систему будут субъекты, уровень компетенции и опыт которых в сфере информационных технологий различен, то система спроектирована в стиле минимализма, с обеспечением удобности, предсказуемости и наглядности интерфейса.

Необходимо также рассмотреть аппаратные и программные требования, необходимые для функционирования приложения. Так как приложение построено на веб-архитектуре, то целесообразно рассмотреть отдельно требования к серверной и клиентской частям.

Работа с приложением со стороны клиенты возможна как с компьютеров, так и с мобильных устройств. Основное требование – наличие доступа в Интернет или локальную сеть (в зависимости от расположения приложения), а также возможность запуска современных браузеров (Internet Explorer 7+, Firefox, Opera, Chrome, Chromium, Safari) [20].

На серверной части должна быть установлена сетевая операционная система (Windows, Unix, Linux-подобные системы). Основное требование – возможность установки веб-сервера, поддерживающего возможность переадресации вопросов, а также возможность работы с PHP версией не ниже 5.4 и установленным расширением MCrypt (к таким веб-серверам относится Apache, Nginx, lighttpd, IIS и тому подобные) [19]. Требования к аппаратной части зависят от планируемой нагрузки на систему и рассчитываются в каждой конкретной ситуации отдельно.

Для обучения работы с программой планируется использование нижеприведенной инструкции.

При переходе на страницу приложения доступны пункты меню «Главная», «Войти» и «Зарегистрироваться» (рисунок 3.1).



Рисунок 3.1 – Меню для неавторизованного пользователя

Для того, чтобы использоваться приложение, необходимо авторизоваться. В случае, если пользователь еще не зарегистрирован в системе, то ему необходимо зарегистрироваться. Для этого необходимо выбрать пункт меню «Зарегистрироваться», после чего будет показана форма, представленная на рисунке 3.2.

Рисунок 3.2 – Регистрация в системе

Необходимо заполнить все поля, как это показано на рисунке 3.2 и нажать кнопку «Создать новую учетную запись». После этого на указанную почту будет отправлено письмо с инструкцией по активации аккаунта. Пример такого письма представлен на рисунке 3.3.

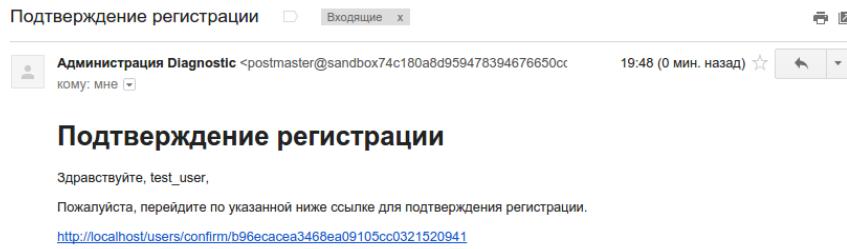


Рисунок 3.3 – Письмо для подтверждения регистрации

После перехода по указанной в письме ссылке аккаунт будет активирован и пользователь увидит форму, представленную на рисунке 3.4.

Вход в систему

Email / Имя пользователя
Email / Имя пользователя

Пароль
Пароль
(я забыл пароль)

Запомнить меня

Ваш аккаунт подтвержден! Теперь вы можете войти в систему.

Войти

Рисунок 3.4 – Форма входа в приложение

После этого необходимо ввести имя пользователя (или E-mail) и пароль, указанные при регистрации, в форму и нажать кнопку «Войти». В случае, если пользователь уже был зарегистрирован в системе, то необходимо сразу выбрать пункт меню «Войти», после чего действия будут аналогичными, как и для первого

входа в приложение после регистрации. После входа пользователю становятся доступны дополнительные пункты меню, а также окно поиска (рисунок 3.5).

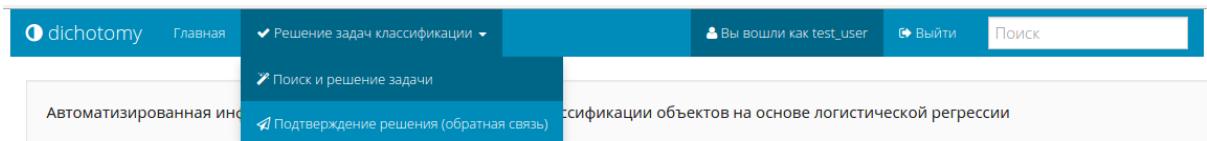


Рисунок 3.5 – Меню зарегистрированного пользователя

Если пользователь забыл пароль, то необходимо перейти по ссылке «я забыл пароль» на форме входа, после чего будет показана форма восстановления пароля (рисунок 3.6).

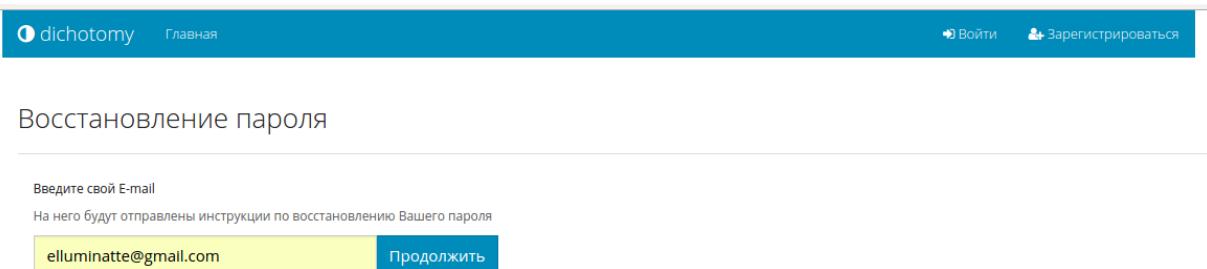


Рисунок 3.6 – Форма восстановления пароля

Необходимо ввести свой E-mail, указанный при регистрации и нажать кнопку «Продолжить». После чего на указанную почту будет отправлено электронное письмо, пример содержания которого представлен на рисунке 3.7.

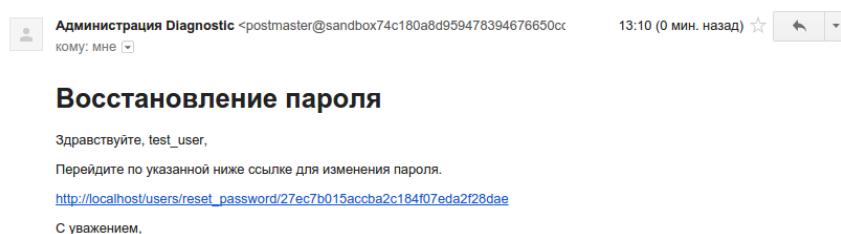


Рисунок 3.7 – Письмо о восстановлении пароля

Необходимо перейти по указанной в письме ссылке, в результате чего будет

показана форма сброса пароля (рисунок 3.8).

The screenshot shows a password reset form with the following fields:

- Пароль**: A text input field containing '*****'.
- Повторите пароль**: A text input field containing '*****'.
- Продолжить**: A blue button at the bottom.

Рисунок 3.8 – Форма сброса пароля

Необходимо ввести пароль и его подтверждение в поля в формы, после чего нажать кнопку «Продолжить». После чего пароль будет изменен и авторизация в приложении станет доступна с использованием нового пароля.

Для просмотра списка проблемных ситуаций, поиска и решения задачи классификации необходимо воспользоваться пунктом меню «Решение задач классификации» => «Поиск и решение задачи». После этого будет показан каталог проблемных ситуаций (рисунок 3.9).

The screenshot shows a catalog of problem situations with the following structure:

Название	Соответствие ОКВЭД	Решаемые задачи
Сельское, лесное хозяйство, охота, рыболовство и рыбоводство	Раздел А.	
Добыча полезных ископаемых	Раздел В.	
Обрабатывающие производства	Раздел С.	
Обеспечение электрической энергией, газом и паром; кондиционирование воздуха	Раздел D.	

Рисунок 3.9 – Каталог проблемных ситуаций

Синий цвет названия проблемной ситуации и символ стрелки указывает на то,

что у данной проблемной ситуации имеются вложенные ситуации. Для перехода ко вложенным ситуациям нужно кликнуть левой кнопки мыши на названии ситуации. Если у ситуации вложенных уровней не имеется, то символ стрелки не отображается, а цвет названия становится темно-серым (рисунок 3.10).

The screenshot shows a web-based application with a blue header bar. The header includes the logo 'dichotomy', navigation links 'Главная' (Home), 'Решение задач классификации' (Classification Task Solving) with a dropdown arrow, user information 'Вы вошли как test_user', a 'Выход' (Logout) button, and a search bar 'Поиск'. Below the header, the page title is '▼ Каталог проблемных ситуаций'. A breadcrumb navigation bar shows the path: 'Главная / Каталог проблемных ситуаций / Сельское, лесное хозяйство, охота, рыболовство и рыбоводство / Растениеводство и животноводство, охота и предоставление соответствующих услуг в этих областях / Выращивание однолетних культур'. The main content area displays a table with three rows of data:

Название	Соответствие ОКВЭД	Решаемые задачи
Выращивание зерновых (кроме риса), зернобобовых культур и семян масличных культур	Код ОКВЭД 01.11	
Выращивание риса	Код ОКВЭД 01.12	
Выращивание овощей, бахчевых, корнеплодных и клубнеплодных культур, грибов и трюфелей	Код ОКВЭД 01.13	

Рисунок 3.10 – Последний уровень вложенности каталога проблемных ситуаций

Если для конкретной проблемной ситуации имеются задачи классификации, то их количество отображается на синем фоне в колонке «Решаемые задачи» (первая строка на рисунке 3.6). В противном случае в колонке «Решаемые задачи» отображается символ крестика на сером фоне. В случае, когда имеются задачи классификации, для перехода к их списку необходимо кликнуть по синему фону в колонке «Решаемые задачи» для соответствующей проблемной ситуации. В результате этого будет показан список решаемых задач (рисунок 3.11).

The screenshot shows the same application interface as in Figure 3.10. The page title is '▼ Список решаемых задач'. The main content area displays a table with two rows of data:

Название
adssssssssssss
Тестовая задача

Рисунок 3.11 – Список решаемых задач

Для решения какой-либо задачи из списка необходимо кликнуть левой кнопкой мыши по ее названию, после чего будет показана форма ввода параметров задачи классификации (рисунок 3.12).

The screenshot shows a user interface for a 'Test task'. At the top, there is a header with the title 'Тестовая задача' and two informational lines: 'Информация' and 'Время корректности решения: час'. Below this is a section titled 'Введите значения параметров для решения задачи'. This section contains several input fields with dropdown menus for selecting values:

- Возраст клиента**: Количество полных лет (dropdown: 1-100 лет)
- Пол**: 1-Мужской; 0-Женский. (dropdown: 1-Мужской; 0-Женский)
- Состоит ли в браке**: 1-Да; 0 - Нет. (dropdown: 1-Да; 0 - Нет)
- Количество иждивенца**: Количество человек (dropdown: 1-10 человек)
- Годовой доход**: Денежные единицы (тысяч \$) (dropdown: 1-100 тысяч \$)
- Опыт работы**: Лет (с любой точностью) (dropdown: 1-100 лет)
- Срок проживания в регионе**: Количество целых лет (dropdown: 1-10 лет)
- Недвижимость**: Денежные единицы (тысяч \$) (dropdown: 1-10 тысяч \$)
- Месячный платеж**: Денежные единицы (\$) (dropdown: 1-1000 долларов)

At the bottom of the form are two buttons: 'Решить' (Solve) and 'Отмена' (Cancel).

Рисунок 3.12 – Форма ввода параметров задачи классификации

Над формой выводится название задачи, а также дополнительная информация и время корректности решения (время, в течение которого корректен результат решения задачи и по истечении которого пользователю будет закрыта возможность использовать функционал по решению задач). Для решения задачи необходимо ввести все параметры. В полях для ввода и под ними выводится информация о единицах измерения параметра, а также список его возможных значений, когда это необходимо. После заполнения всех полей формы необходимо нажать кнопку «Решить».

Результат представляет собой значение от 0 до 1. Положительный результат означает, что значение результата превысило используемый в данной задаче порог отсечения (границу между положительным и отрицательным результатами). Отрицательный результат означает, что значение результата оказалось меньше порога отсечения.

После решения задачи классификации пользователь будет видеть на всех страницах приложения напоминание о необходимости совершить обратную связь

(рисунок 3.14).

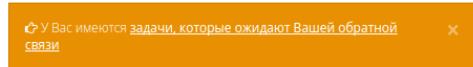


Рисунок 3.14 – Напоминание о необходимости совершить обратную связь

В случае, когда срок корректности решенных задач истек, пользователь увидит другое напоминание (рисунок 3.15). Кроме того, доступ к решению других задач будет закрыт до тех пор, пока пользователь не совершил обратную связь для всех задач, срок корректности решения для которых истек.

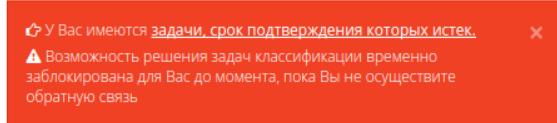


Рисунок 3.15 – Напоминание о наличии задач, срок корректности которых истек

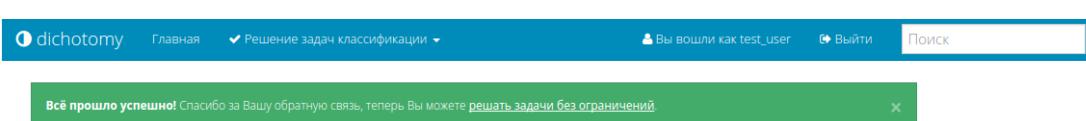
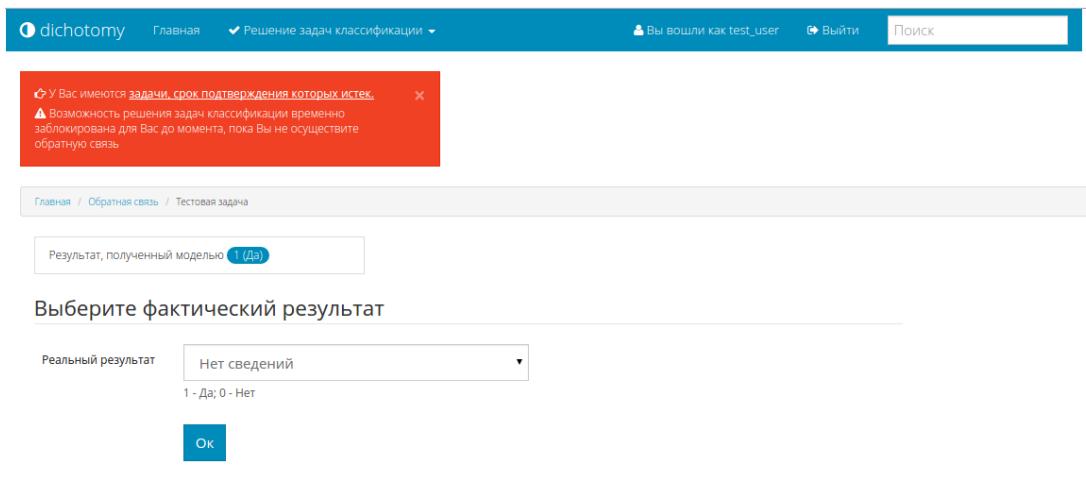
Для совершения обратной связи пользователю необходимо перейти по ссылке, отображаемой в напоминаниях (рисунки 3.14 и 3.15) или выбрать пункт меню «Решение задач классификации» => «Подтверждение решения (обратная связь)». После этого будет показан список решений, которые ждут обратной связи (рисунок 3.16).

Задача	Окончание времени корректности решения	Перейти к обратной связи
Тестовая задача	2015-03-14 21:51:44	

Рисунок 3.16 – Список решений, ожидающих обратной связи

В данном списке указана решенная задача и время окончания корректности решения. Для совершения обратной связи необходимо нажать кнопку в колонке «Перейти к обратной связи» напротив нужной задачи. После чего будет показана форма совершения обратной связи (рисунок 3.17).

Перед формой отображается результат, вычисленный моделью (0 – Нет, 1 – Да). Ниже расположен выпадающий список, из которого нужно выбрать реальный исход решаемой задачи или указать, что у пользователя нет сведений об исходе задачи. Кроме того, при нажатии на ссылку «Показать значения параметров», можно увидеть введенные при решении задачи параметры классификации. После ввода реального результата, необходимо нажать кнопку «Ок». В результате обратная связь будет осуществлена и будет выведено уведомление об этом. В случае, если задача, срок корректности решения которых истек, у пользователя больше нет, то доступ к решению задач будет немедленно восстановлен (рисунок 3.18).



Также в приложении реализован поиск по названию проблемной ситуации, коду ОКВЭД и решаемой задачи. Для использования поиска необходимо

ввести в окно поиска (правый верхний угол) фразу поиска и нажать кнопку Enter. После этого будут выведены результаты поиска, пример которых представлен на рисунке 3.19.

The screenshot shows a search interface with the following details:

- Top Navigation:** dichotomy, Главная, Управление, Решение задач классификации, Вы вошли как admin, Выход, Поиск.
- Section 1: Совпадения по коду ОКВЭД**
 - Message: Совпадений по коду ОКВЭД не найдено.
 - Table:
| | |
| --- | --- |
| Деятельность финансовых и страховых организаций | Раздел К |
| Деятельность по предоставлению финансовых услуг, кроме услуг по страхованию и пенсионному обеспечению | Код ОКВЭД 60 |
| Деятельность вспомогательная в сфере финансовых услуг и страхования | Код ОКВЭД 60.01 |
| Деятельность инвестиционных фондов и аналогичных финансовых организаций | Код ОКВЭД 60.02 |
| Деятельность по предоставлению прочих финансовых услуг, кроме услуг по страхованию и пенсионному обеспечению | Код ОКВЭД 60.03 |
| Деятельность инвестиционных фондов и аналогичных финансовых организаций | Код ОКВЭД 60.04 |
| Деятельность финансовой аренды (leasing/убыльзанту) | Код ОКВЭД 60.05 |
| Предоставление прочих финансовых услуг, кроме услуг по страхованию и пенсионному обеспечению, не включенных в другие группировки | Код ОКВЭД 60.06 |
| Деятельность вспомогательная в сфере финансовых услуг, кроме страхования и пенсионного обеспечения | Код ОКВЭД 60.07 |
| Управление финансовыми рынками | Код ОКВЭД 60.11 |
 - Note: Здесь показаны не все совпадения по названию проблемной ситуации, их слишком много. Пожалуйста, уточните условия поиска.
- Section 2: Совпадения по названию решаемой проблемы**
 - Message: Совпадений по названию решаемой проблемы не найдено.

Рисунок 3.19 – Результаты поиска

Как видно из данного рисунка, в случае, если результатов по критерию поиска нет, будет выведено уведомление об этом. При наличии результатов они будут выведены списком с возможностью перехода к ним по ссылке. Кроме того, если результатов более 10, то будет выведено уведомление о необходимости уточнить поисковую фразу.

В случае если пользователь авторизовался под пользователем, обладающим ролью «администратор», то ему доступны также дополнительные административные пункты меню (рисунок 3.19), но ему также доступен весь функционал роли «пользователь», кроме того, в случае наличия решений задач, срок корректности которых истек, администратор может продолжать пользоваться всем функционалом системы.

The screenshot shows the administrator menu with the following sections:

- Top Navigation:** dichotomy, Главная, Управление, Решение задач классификации, Вы вошли как admin, Выход, Поиск.
- Left Sidebar:** Каталог проблемных ситуаций, Несколько задач.
- Bottom Bar:** У Вас имеются задачи, которые требуют решения.

Рисунок 3.19 – Меню администратора

Для управления каталогом проблемных ситуаций необходимо выбрать пункт меню «Управление» => «Каталог проблемных ситуаций», в результате чего будет показан каталог, представленный на рисунке 3.20.

The screenshot shows a web application interface for managing a catalog of problem situations. At the top, there is a navigation bar with links for 'dichotomy', 'Главная' (Home), 'Управление' (Management), 'Решение задач классификации' (Classification Task Resolution), 'Вы вошли как admin' (You logged in as admin), 'Выход' (Logout), and 'Поиск' (Search). Below the navigation bar, the title 'Каталог проблемных ситуаций' (Catalog of Problem Situations) is displayed. A breadcrumb navigation shows 'Главная / Каталог проблемных ситуаций'. A blue button labeled 'Добавить ситуацию' (Add Situation) is visible. The main content area displays a table with three rows of data:

Название	Соответствие ОКВЭД	Редактировать реквизиты	Удалить	Решаемые задачи
1 Сельское, лесное хозяйство, охота, рыболовство и рыбоводство	Раздел А.			
1 Добыча полезных ископаемых	Раздел В.			
1 Обрабатывающие производства	Раздел С.			

Рисунок 3.20 – Управление каталогом проблемных ситуаций

Для перехода к следующему уровню вложенности необходимо перейти по ссылке названия нужной проблемной ситуации, где будет показан аналогичный каталог.

Для добавления новой проблемной ситуации на текущем уровне иерархии необходимо нажать кнопку «Добавить ситуацию», в результате чего будет показана форма добавления ситуаций, представленная на рисунке 3.21.

The screenshot shows a form for adding a new situation. At the top, there is a navigation bar with links for 'dichotomy', 'Главная' (Home), 'Управление' (Management), 'Решение задач классификации' (Classification Task Resolution), 'Вы вошли как admin' (You logged in as admin), 'Выход' (Logout), and 'Поиск' (Search). Below the navigation bar, the title 'Добавление ситуации' (Add Situation) is displayed. A breadcrumb navigation shows 'Главная / Каталог проблемных ситуаций / Добавление ситуации'. The form has two input fields: 'Название' (Name) and 'Соответствие ОКВЭД' (OKVED Correspondence). At the bottom, there are two buttons: 'Добавить' (Add) and 'Отмена' (Cancel).

Рисунок 3.21 – Форма добавления ситуации

После заполнения полей необходимо нажать кнопку «Добавить», после чего ситуация будет добавлена, о чем говорит уведомление, представленное на рисунке 3.22.



Рисунок 3.22 – Результат добавления проблемной ситуации

Для изменения реквизитов проблемной ситуации необходимо нажат кнопку с изображением карандаша в колонке «Редактировать реквизиты», после чего будет показана форма редактирования реквизитов, аналогичная форме добавления с тем лишь отличием, что поля формы будут уже заполнены.

Для удаления проблемной ситуации необходимо нажать кнопку с изображением крестика в колонке «Удалить». При этом необходимо помнить о том, что при удалении проблемной ситуации будут также удалены и все вложенные ситуации. При нажатии на кнопку «Удалить» будет выведено подтверждение, изображенное на рисунке 3.23.

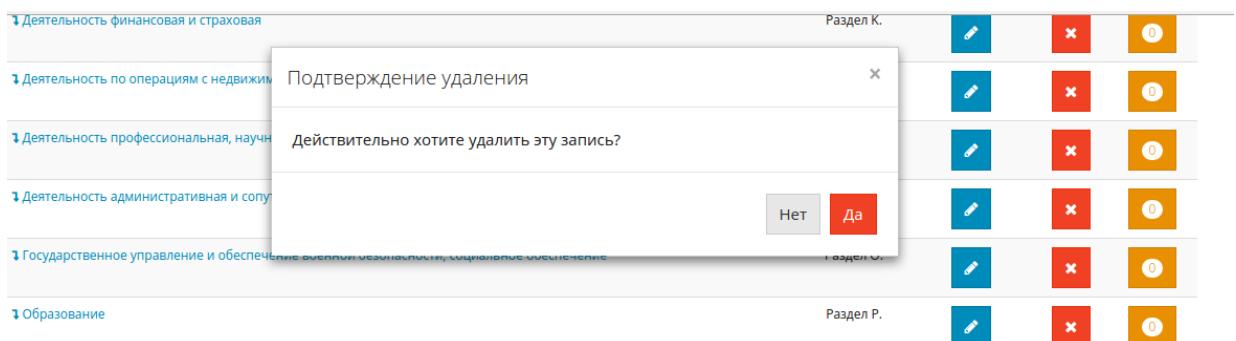


Рисунок 3.23 – Подтверждение удаления

В случае если пользователь уверен в необходимости удаления, то нужно нажать кнопку «Да». В противном случае нужно отменить удаление нажатием кнопки «Нет».

Для перехода к списку решаемых задач той или иной проблемной ситуации необходимо нажать на кнопку в колонке «Решаемые задачи» напротив нужной ситуации. При этом число в данной кнопке отражает количество уже имеющихся задач. При переходе к списку задач будет показан список, представленный на рисунке 3.24.

Название	Редактирование параметров	Удалить задачу
adssssssssssss		
Тестовая задача		

Рисунок 3.24 – Управление списком решаемых задач

Для просмотра модели задачи нужно нажать на название задачи, после чего будет выведена страница, изложенная на рисунке 3.25.

Название	Свободный член	Возраст клиента	Пол	Состоит ли в браке	Количество иждивенцев	Годовой доход	Опыт работы	Срок проживания в регионе
Единицы измерения	-	Количество полных лет	1 -Мужской; 0 - Женский.	1 -Да; 0 - Нет.	Количество человек	Денежные единицы (тысяч \$)	Лет (с любой точностью)	Количество целых лет
Коэффициенты	-3.4914443304935	-0.0030519305391835	0.6594259863075	-0.23531184513765	-1.8764778969169	0.00071536491570002	0.0031809371409225	0.0093042642933681
Эластичные коэффициенты	-	-0.21330570328546	0.87565324454682	-0.28179911393266	-3.9707953867549	19.309024903813	0.079461882896101	0.25527035498159
Стандартизованные коэффициенты	-	-0.043711631374981	0.63023110974707	-0.23151444951097	-3.5665797187412	10.768264686935	0.033650841150658	0.20162045905623

Рисунок 3.25 – Информация о модели задачи

Здесь можно скачать основную обучающую выборку в формате xls, нажав кнопку «Скачать основную обучающую выборку». Ниже представлено название задачи, дополнительная информация, а также информация о выходной переменной. Кроме того демонстрируется информация о минимальном пороге отсечения, фактическом пороге, площади под кривой и времени корректности решения задачи. Ниже представлена информация о единицах измерения регрессоров, найденные коэффициенты, частные коэффициенты эластичности и стандартизованные коэффициенты.

Для добавления новой задачи необходимо нажать кнопку «Добавить задачу» в списке решаемых задач. После этого будет показана форма добавления новой задачи (рисунок 3.26).

The screenshot shows the 'dichotomy' application interface. At the top, there is a navigation bar with links for 'Главная', 'Управление', 'Решение задач классификации', 'Вы вошли как admin', 'Выход', and a search bar. Below the navigation bar, the URL path is shown: Главная / Каталог проблемных ситуаций / Сельское, лесное хозяйство, охота, рыболовство и рыбоводство / Добавление задачи. The main content area is titled 'Добавление задачи'. It contains several input fields and controls:

- 'Название' (Name) field with placeholder 'Название'.
- 'Время корректности решения' (Time of correct solution) dropdown menu showing 'час' (hour) with the value '75' highlighted.
- 'Минимальный порог отсечения' (Minimum threshold for pruning) slider with a value of 75.
- 'Информация о задаче' (Information about the task) text area.
- 'Файл обучающей выборки' (Training set file) button with placeholder 'Выберите файл' (Select file) and message 'Файл не выбран' (File not selected).
- 'Добавить' (Add) and 'Отмена' (Cancel) buttons at the bottom.

Рисунок 3.26 – Добавление новой задачи

Здесь необходимо ввести название задачи, время корректности ее решения, минимальный порог отсечения, дополнительную информацию о задаче, а также загрузить файл обучающей выборки. Нажав на ссылку «Скачать шаблон файла обучающей выборки» можно скачать xls-файл с правилами заполнения обучающей выборки, а также с примером правильного заполнения. После ввода в форму всех необходимых данных необходимо нажать кнопку «Добавить». В результате

добавления будет выведено информационное сообщение, представленное на рисунке 3.27 и задача появится в списке.

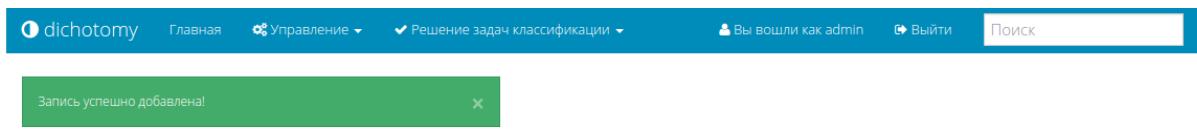


Рисунок 3.27 – Результат добавления задачи

Для изменения параметров задачи необходимо нажать кнопку с символом карандаша в колонке «Редактирование параметров» в списке задач. После этого будет показана форма редактирования задачи (рисунок 3.28).

A screenshot of a web application interface for editing task parameters. At the top, there is a navigation bar with links for 'dichotomy', 'Главная' (Home), 'Управление' (Management), 'Решение задач классификации' (Solving classification tasks), 'Вы вошли как admin' (You logged in as admin), 'Выход' (Logout), and 'Поиск' (Search). Below the navigation bar, the URL path is shown: Главная / Каталог проблемных ситуаций / Сельское, лесное хозяйство, охота, рыболовство и рыбоводство / Тестовая задача / Редактирование параметров. The main content area has a title 'Редактирование параметров задачи'. It contains several input fields: 'Название' (Name) with the value 'Тестовая задача'; 'Время корректности решения' (Time of correct solution) with a dropdown menu showing 'час' (hour) and a value '75'; 'Минимальный порог отсечения' (Minimum threshold for pruning) with a slider set to 75; 'Информация о задаче' (Information about the task) with a text area containing 'Информация'; and a file upload section for 'Файл обучающей выборки' (Training sample file) with a note: 'Выберите файл' (Select file), 'Файл не выбран' (File not selected), and 'Внимание! При загрузке файла модель будет переобучена. При этом данные дополнительной выборки будут утеряны.' (Attention! When loading the file, the model will be retrained. In this case, the data of the additional sample will be lost.). At the bottom, there are two buttons: 'Изменить' (Change) and 'Отмена' (Cancel).

Рисунок 3.28 – Форма редактирования задачи

Форма редактирования задачи аналогична форме добавления, за исключением того, что поля формы уже заполнены (кроме файла обучающей выборки). Если будет загружен файл обучающей выборки, то модель задачи будет переобучена, а также будут удалены пользовательские вычисления. Если нет необходимости переобучать модель, то нужно оставить данное поле незаполненным. После заполнения всех необходимых полей нужно нажать кнопку «Изменить».

Если в процессе переобучения моделей задач или совершения обратной связи

порог отсечения модели станет ниже минимального порога, заданного администратором, то модель будет деактивирована – она станет не видна пользователям, а в административном интерфейсе появится значок замка (рисунок 3.29).

The screenshot shows a section titled 'Список решаемых задач' (List of solvable tasks). A single row is visible, representing a task named 'Благонадежность заемщика1111'. The row includes a blue 'Редактирование параметров' (Edit parameters) button with a pencil icon and a red 'Удалить задачу' (Delete task) button with a trash icon. Above the table, a breadcrumb navigation path is shown: Главная / Каталог проблемных ситуаций / Деятельность финансовая и страховая / Деятельность по предоставлению финансовых услуг, кроме услуг по страхованию и пенсионному обеспечению / Предоставление займов и прочих видов кредита.

Рисунок 3.29 – Отображение деактивированной задачи в административном интерфейсе

Кроме того, на почту администратора будет отправлено электронное письмо с уведомлением о деактивации задачи, пример которого представлен на рисунке 3.30.

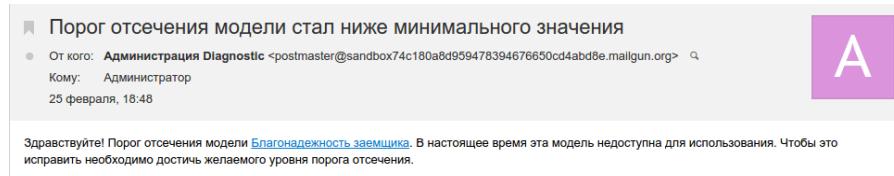


Рисунок 3.30 – Письмо о деактивации модели

Также для просмотра всех деактивированных моделей можно воспользоваться пунктом меню «Управление» => «Неактивные задачи». В результате будет показан список всех неактивных задач, имеющихся в приложении (рисунок 3.31).

The screenshot shows a section titled 'Список неактивных задач' (List of inactive tasks). A table displays two rows of inactive tasks: 'Благонадежность заемщика1111' and 'Тестирование11'. The table has a header row with a 'Название' (Name) column. At the top of the page, there is a navigation bar with links for 'dichotomy', 'Главная' (Home), 'Управление' (Management), 'Решение задач классификации' (Classification task solving), 'Вы вошли как admin' (You logged in as admin), 'Выход' (Logout), and 'Поиск' (Search). A breadcrumb path 'Главная / Неактивные задачи' is also present.

Рисунок 3.31 – Список неактивных задач

Таким образом, при использовании приведенной выше инструкции обеспечивается эффективное использование всех возможностей разработанной программы, как для пользователя, так и для администратора системы.

3.2 Обоснование эффективности проекта

Цель внедрения современных информационных технологий – повышение эффективности деятельности объекта внедрения, будь то организация, индивидуальный предприниматель, орган государственной власти или местного самоуправления. Внедрения информационных технологий представляет собой проект, поэтому для оценки эффективности внедрения используются методы оценки эффективности инвестиций. В настоящее время существует ряд таких методов, которые условно подразделяются на следующие группы [27]:

- классические методы оценки инвестиционных проектов, предполагающие расчет таких показателей как чистый приведенный доход (Net Present Value, NPV), внутренняя норма доходности (Internal Rate of Return, IRR), срок окупаемости (Payback), добавленная стоимость (Economic Value Added, EVA);
- затратные методы оценки, основными из которых можно назвать определение совокупной стоимости владения (Total Cost of Ownership, TCO) и его производные, такие как истинная стоимость владения (Real Cost of Ownership, RCO), совокупная стоимость владения приложениями (Total Cost of Application Ownership, TCA);
- комплексные методы оценки набора финансовых и нефинансовых показателей эффективности (Key Performance Indicators, KPI), такие как сбалансированная система показателей Нортон и Каплана (Balanced Scorecard, BSC), модель «стейкхолдер» и пирамида результативности Линча и Кросса.

В виду того, что разработанная система может быть внедрена во множество отдельных разнородных организаций, а также то, что на данный момент внедрение системы не производилось, целесообразно воспользоваться затратным методом оценки эффективности.

Эффект от реализации проекта в основном косвенный и выражается в повышении обоснованности и надежности принимаемых решений за счет применения объективных математических методов. Возможно также появление прямых эффектов, но это зависит от объекта внедрения информационной системы и не может быть рассчитано на данном этапе.

Для расчета затрат необходимо учесть затраты, понесенные на необходимое программное обеспечение, а также затраты, связанные с оплатой работы. Для разработки информационной системы нужно следующее программное обеспечение: Ubuntu Linux, Open Office, Nginx, PostgreSQL, PHP, Sublime Text. Все перечисленное программное обеспечение является свободным и затрат на его покупку и эксплуатацию не требуется.

Для расчета затрат, связанных с оплатой работы необходимо определить трудоемкость разработки. В разработке данной информационной системы принимал участие один человек. Поэтапная оценка трудоемкости представлена в таблице 3.1.

Таблица 3.1 – Поэтапная трудоемкость процесса разработки

Наименование работы	Исполнитель	Трудоемкость, чел-дней
Изучение проблемы и возможных методов решения	Разработчик	20
Составление технического задания	Разработчик	7
Выбор средств разработки	Разработчик	7
Разработка информационной системы	Разработчик	60
Тестирование информационной системы	Разработчик	7
Подготовка документации	Разработчик	7
Итого		108

Средняя заработка веб-разработчика равна 30 000 рублей [28]. Для расчета затрат на заработную плату разработчика информационной системы необходимо воспользоваться формулой (3.1).

$$33_{\Pi} = \frac{03_{\Pi}}{N_d} * T_3, \quad (3.1)$$

где ЗЗП – затраты на заработную плату разработчика, ОЗП – заработка плата за месяц основной работы, Нд – количество рабочих дней в месяц, Тз – трудовые затраты при работе над проектом (в днях).

Таким образом, затраты на заработную плату разработчика информационной системы, принимая во внимание число рабочих дней в месяц равным 21, равна $30000 \text{ руб./день} * 108 \text{ дней} \approx 154285 \text{ руб.}$

Кроме того необходимо учесть также затраты на оплату машинного времени. Они зависят от времени работы на ЭВМ, себестоимости машино-часа работы ЭВМ и включают в затраты на электроэнергию [27]. Для проектирования информационной системы использовался персональный компьютер, потребляющий 0,3 кВт/ч. Стоимость 1 кВт электроэнергии стоит 3,01 рублей. Стоимость машино-часа работы рассчитывается по формуле (3.2.).

$$C_{\text{мч}} = \Pi * C_{\text{kVt}} , \quad (3.2)$$

где Π – потребляемая ЭВМ электроэнергия, C_{kVt} – стоимость 1 кВт энергии [27].

Тогда стоимость машино-часа работы в данном случае равна $0,3 \text{ кВт/ч} * 3,01 \text{ руб} = 0,903 \text{ руб./час.}$

ЭВМ в данном случае работала по 8 часов в каждый человеко-день, значит общее время работы ЭВМ равно $8 \text{ ч.} * 108 \text{ д.} = 864 \text{ часа.}$ Себестоимость электроэнергии рассчитывается по формуле (3.3).

$$C_{\text{эл}} = T_{\text{ЭВМ}} * C_{\text{мч}}, \quad (3.3)$$

где $T_{\text{ЭВМ}}$ – время работы ЭВМ, $C_{\text{мч}}$ – стоимость машино-часа [27].

Тогда суммарные затраты на электроэнергию составят $864 \text{ ч.} * 0,903 \text{ руб.час.} \approx 780 \text{ рублей.}$

Общие затраты на разработку информационной системы в данном случае складываются из затрат на заработную плату, а также затрат на электроэнергию. Таким образом, общие затраты составят 780 руб. + 154285 руб. = 155 065 рублей.

С учетом данных затрат в конкретных случаях внедрения системы возможен расчет эффективности проекта. Учитывая то, что данные затраты были понесены единовременно и то, что возможно многократное использование данной системы во множестве организаций, а также то, что программное обеспечение распространяется под свободной лицензией, был сделан вывод о том, что в большинстве случаев внедрения разработанной системы удастся обеспечить высокую эффективность проекта.

3.3 Выводы по главе

Третья глава дипломного проекта была посвящена формированию технологической среды и анализу эффективности проекта. В рамках формирования технологической среды были сформулированы программные и аппаратные требования к клиентской и серверной частям. Кроме того, были определены категории пользователей, а также составлена инструкция для эффективного использования возможностей системы.

При анализе эффективности проекта были рассмотрены основные подходы к оценке эффективности проектов, был выбран метод для применения в рамках данной работы. В результате длительность разработки составила 108 дней, затраты на разработку системы составили 155 065 рублей.

ЗАКЛЮЧЕНИЕ

В результате проделанной работы была разработана автоматизированная информационная система для бинарной классификации объектов на основе логистической регрессии.

В аналитической главе были рассмотрены области применения и методы решения задач бинарной классификации. В ходе анализа было определено, что решение подобных задач является актуальным вопросом в широчайшем круге предметных областей, а одним из наиболее релевантных методов решения является применение логистической регрессии. Поэтому были подробно рассмотрены теоретические и технологические особенности применения данного аппарата к решению проблем бинарной классификации. Кроме того, были рассмотрены уже существующие разработки. По результатам данного анализа был сделан вывод о целесообразности разработки новой системы. Исходя из специфики решаемой задачи, также была выбрана и обоснована технология проектирования информационной системы.

В проектной части диплома были сформированы функциональные требования к проектируемой информационной системе с точки зрения всех групп пользователей. Также было рассмотрено информационное обеспечение задачи автоматизации. В результате описания информационного обеспечения были описаны концептуальные характеристики выходной и входной информации, а также были разработаны логическая и физическая модели базы данных. При описании алгоритмического обеспечения задачи были рассмотрены наиболее интересные алгоритмы, описан их математический аппарат и построены блок-схемы. При рассмотрении программного обеспечения была описана информационная структура приложения, а сформирована диаграмма классов с точки зрения реализации системы. По результатам проектирования было проведено тестирование системы, в ходе которого был сделан вывод о надлежащем качестве разработанной информационной системы.

В третьей части диплома была сформирована технологическая среда системы

– описаны требования к клиентской и серверной частям, определены категории пользователей и сформирована инструкция пользователя, позволяющая эффективно использовать все функциональные возможности системы для всех категорий пользователей. Кроме того, была проведена оценка эффективности проекта затратным методом, которая показала, что затраты на разработку системы составили 155 065 рублей. Было определено, что за счет распространения под свободной лицензией и возможностью многократного использования системы, эффективность проекта по внедрению информационной системы в большинстве случаев будет обеспечена на высоком уровне.

Результаты данной работы могут быть использованы широким кругом пользователей. Это становится возможным благодаря тому, что все результаты опубликованы под свободной лицензией. Это резко повышает эффективность использования системы за счет минимизации затрат.

Таким образом, данный дипломный проект полностью соответствует техническому заданию и разработанная в ходе его работы, информационная система, отвечает заданным спецификациям.

В заключение можно сделать вывод, что задачи, поставленные в начале проектирования, были решены. Основная цель дипломного проектирования была достигнута посредством выполнения поставленных задач.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Паклин, Н.Б. Бизнес-аналитика: от данных к знаниям [Текст]: Учеб. Пособие / Н.Б. Паклин, В.И. Орешков. – Спб.: Питер, 2013 – 704 с.: ил.
2. Вараксин, А.Н. Регрессионные модели для анализа пространственно-временных данных в задачах медико-экологического мониторинга [Текст] / А.Н. Вараксин, Т.А. Маслакова, В.Н. Чуканов // Экологические системы и приборы. – 2010. - № 8. – С. 49-52.
3. Груздев, А.В. Метод бинарной логистической регрессии в банковском скоринге. Риск-менеджмент в кредитной организации [Текст] / А.В. Груздев // Риск-менеджмент в кредитной организации. – 2012. - № 2. – С. 76-91.
4. Константина, Е.Д. Методология системного анализа взаимосвязей между факторами риска и здоровьем населения в задаче устойчивого развития [Текст] / Е.Д. Константина, А.Н. Вараксин // Международный электронный журнал. Устойчивое развитие: наука и практика. – 2010. - № 2(5). – С. 68-85.
5. Российская Федерация. Законы. Федеральный закон о таможенном регулировании в Российской Федерации [Текст] // Российская газета. – 2010. – 29 нояб. – С. 243-315.
6. Кнут, Д. Искусство программирования. Том 4, А. Комбинаторные алгоритмы. Часть 1 [Текст] / Д. Кнут. – М.: Вильямс, 2012. – 960 с.
7. Hastie, T. The Elements of Statistical Learning: Data Mining, Inference, and Prediction [Текст] / T. Hastie, R. Tibshirani, J. Friedman. – Springer, 2009. – 746 с.
8. Круглов, В.В. Нечеткая логика и искусственные нейронные сети [Текст] / В.В. Круглов, М.И. Дли, Р.Ю. Голунов. – М.: ФИЗМАТЛИТ, 2001. – 224 с.
9. Логистическая регрессия и ROC-анализ - математический аппарат [Электронный ресурс] // Режим доступа: <http://www.basegroup.ru/library/analysis/regression/logistic/>.
10. Кремер, Н.Ш. Эконометрика [Текст] / Н.Ш. Кремер, Б.А. Путко. – М.: Юнити-Дана, 2008. – 312 с.
11. Рассел, Д. Регрессионный анализ [Текст] / Д. Рассел. – М.: Книга по

требованию, 2013. – 90 с.

12. Система профессионального анализа рынка и компаний [Электронный ресурс] // Режим доступа: <http://www.spark-interfax.ru/Front/Index.aspx>.
13. Обзор статистических программ [Электронный ресурс] // Режим доступа: <http://www.sciencefiles.ru/section/46/>.
14. MedCalc - User-friendly statistical software [Электронный ресурс] // Режим доступа: <http://www.medcalc.org/>.
15. Марманис Х. Алгоритмы интеллектуального Интернета. Передовые методики сбора, анализа и обработки данных [Текст] / Х. Марманис, Д. Бабенко. - М.: Символ-Плюс, 2011. – 480 с.
16. PHP: Documentation [Электронный ресурс] // Режим доступа: <http://php.net/docs.php>.
17. McLaughlin, B. What Is HTML5? A New Way to Look at the Web [Текст] / B McLaughlin. - O'Reilly Media, 2011. – 10 с.
18. PostgreSQL: Documentation [Электронный ресурс] // Режим доступа: <http://www.postgresql.org/docs/>.
19. Laravel 4.2: Documentation [Электронный ресурс] // Режим доступа: <http://laravel.com/docs/4.2>.
20. About Bootstrap [Электронный ресурс] // Режим доступа: <http://getbootstrap.com/about/>.
21. jQuery Learning Cente [Электронный ресурс] // Режим доступа: <http://learn.jquery.com/>.
22. Вендро, А.М. Проектирование программного обеспечения экономических информационных систем [Текст] / А.М. Вендро. – М.: Финансы и статистика, 2005. – 544 с.
23. Maximum-Likelihood Estimation of the Logistic Regression Model [Электронный ресурс] // Режим доступа: <http://socserv.socsci.mcmaster.ca/jfox/Courses/UCLA/logistic-regression-notes.pdf>.
24. Ларман, К. Применение UML 2.0 и шаблонов проектирования [Текст] / К. Ларман. - М.: Вильямс, 2006. — 736 с.

25. Рэшка, Д. Тестирование программного обеспечения [Текст] / Д. Рэшка, Э. Дастин, Д. Пол. – М.: Лори, 2012. – 568 с.
26. Применение логистической регрессии в медицине и скоринге [Электронный ресурс] // Режим доступа: http://www.basegroup.ru/library/practice/logis_medic_scoring/.
27. Виленский, П. Л. Оценка эффективности инвестиционных проектов. Теория и практика [Тест] / П.Л. Виленский, В.Н. Лившиц, С.А. Смоляк. – М.: Дело, Академия народного хозяйства, 2008. – 1104 с.
28. Сколько получает Веб программист в России? Средняя зарплата Веб программист в России, статистика Trud.com [Электронный ресурс] // Режим доступа: <http://russia.trud.com/salary/692/3322.html>.

ПРИЛОЖЕНИЕ А

Алгоритмическое обеспечение задачи автоматизации

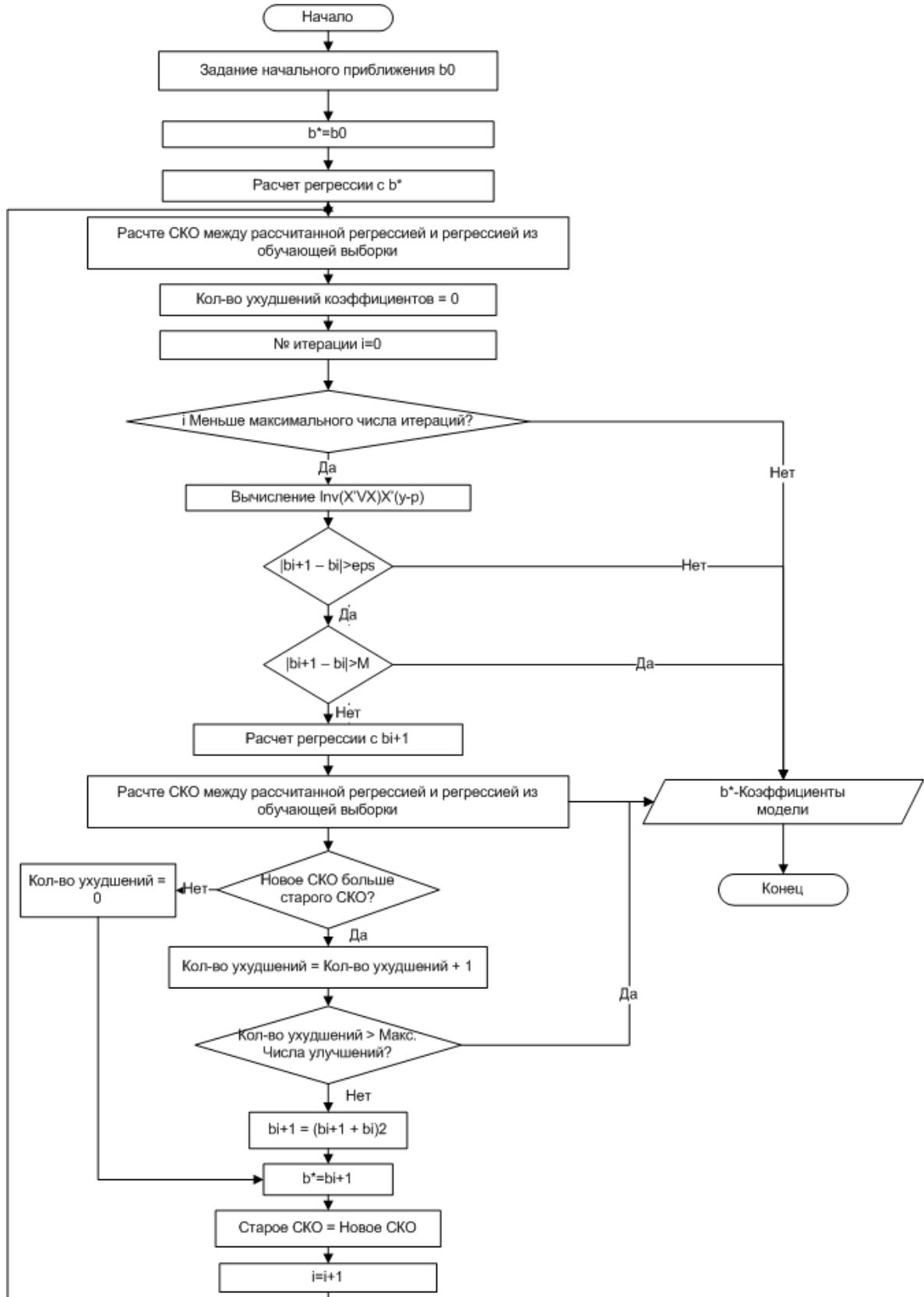


Рисунок А.1 – Блок-схема алгоритма обучения модели логистической регрессии

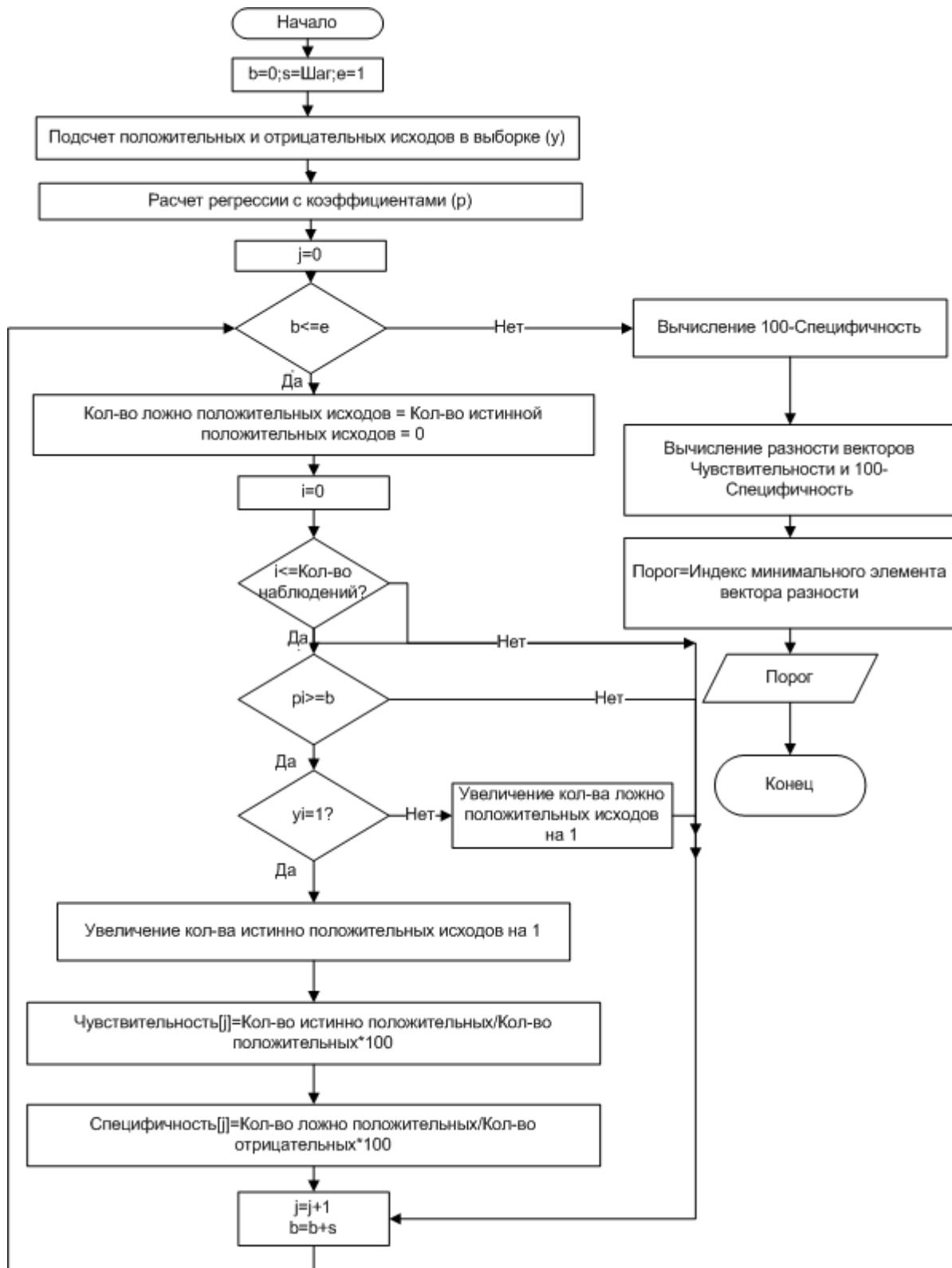


Рисунок А.2 – Блок-схема алгоритма поиска порога отсечения модели

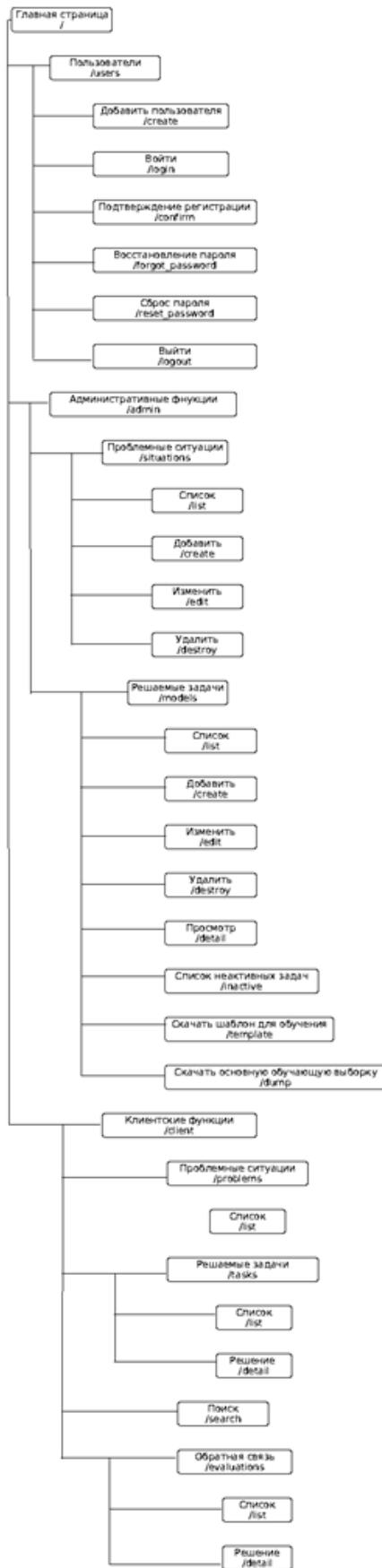


Рисунок А.3 – Информационная структура приложения

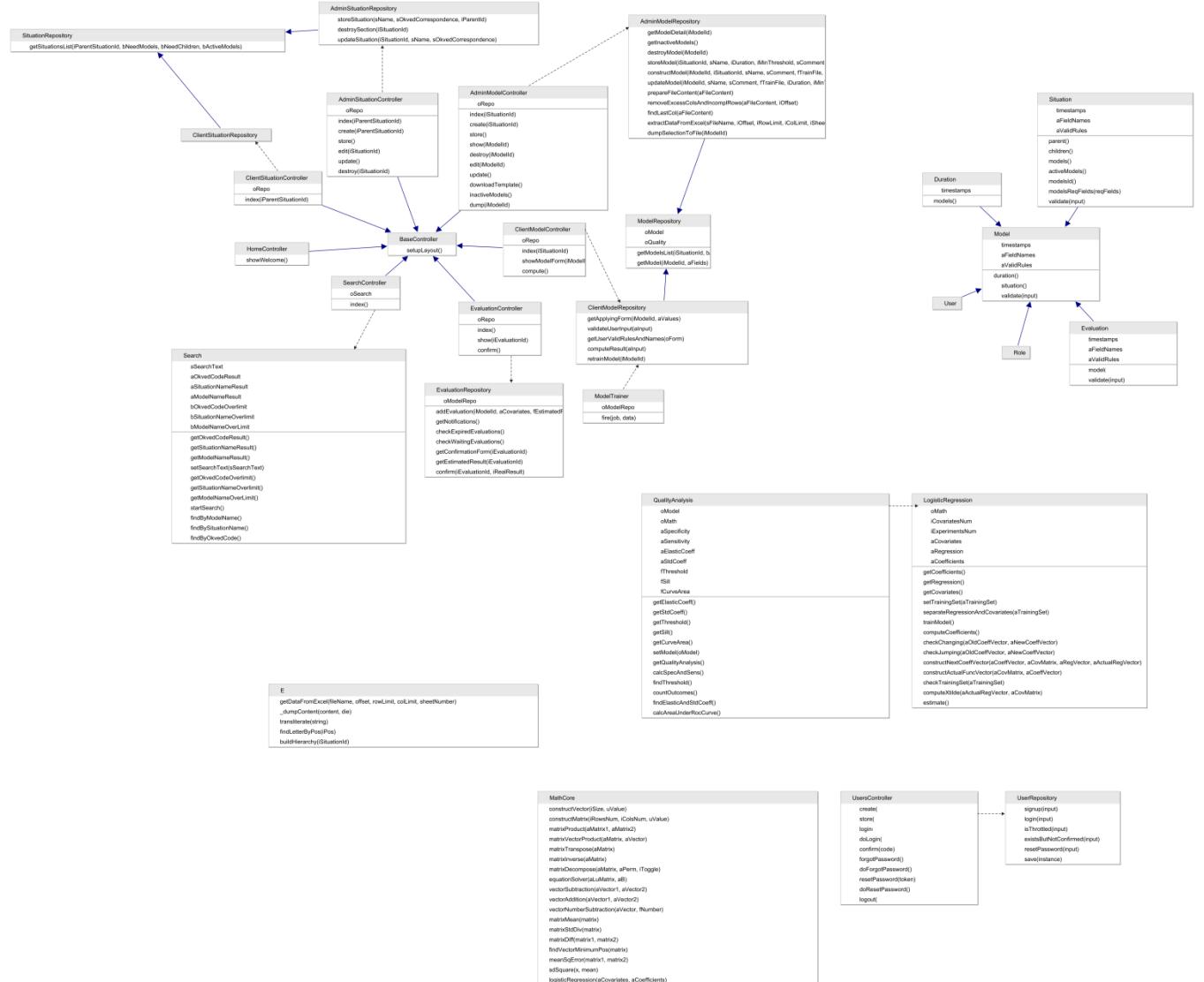


Рисунок А.4 – Диаграмма классов разрабатываемого приложения

ПРИЛОЖЕНИЕ Б

Листинг программы

```
App/classes/Elluminate/Engine/E.php
<?php
/**
 *          Created      by      PhpStorm.
 *          User:        Date:    illuminate
 *          Date:        Time:    06.01.15
 *          Time:        15:19
 */
namespace Illuminate\Engine;
use Illuminate\Exception\FileException;
use Symfony\Component\HttpFoundation\File\Exception\FileException;

/**
 *          Class      E
 *          @package    Illuminate\Engine
 */
class E
{
    /**
     * построчно собирает данные из файла xls или xlsx
     * @param $fileName - имя файла
     * @param int $offset - с какой строки надо читать
     * @param int $rowLimit - сколько строк нужно прочитать
     * @param int $colLimit - сколько столбцов нужно прочитать
     * @param int $sheetNumber - номер листа книги файла
     * @return array - массив данных, содержащихся в файле
     */
    public static function getDataFromExcel($fileName, $offset = 0, $rowLimit = 0, $colLimit = 0, $sheetNumber = 0) {
        $data = [];
        try {
            $inputFileType = PHPExcel_IOFactory::identify($fileName);
            $objReader = PHPExcel_IOFactory::createReader($inputFileType);
            // видит непустую ячейку только там, где есть реальное
            // значение, а не стиль и т.д.
            $objReader->setReadDataOnly(true);
            $objPHPExcel = $objReader->load($fileName);
            catch (\Exception $e) {
                throw new FileException($e);
            }
            $sheet = $objPHPExcel->getSheet($sheetNumber);
            // чтобы не считать ячейки, где есть, например, стиль.
            // должно помочь вместе с setReadDataOnly
            // если заданы ограничения, то читаем до них, иначе -
            // читаем всё, что есть
            $highestRow = $rowLimit ? $rowLimit : $sheet->getHighestDataRow();
            $highestColumn = $colLimit ? $colLimit : $sheet->getHighestDataColumn();

            for ($row = $offset; $row <= $highestRow; $row++) {
                $rowData = $sheet->rangeToArray('A' . $row . ':' . $highestColumn);
                $data[] = $rowData;
            }
        } catch (\Exception $e) {
            return $data;
        }
    }

    /**
     * функция для отладки, выводит на экран содержимое
     * переменной
     * @param $content - переменную, которую нужно вывести
     * @param bool $die - закончить ли выполнение программы
     * после вывода значения
     */
    public static function _dumpContent($content, $die = true) {
        if (is_array($content) || is_object($content)) {
            echo "<XMP>";
            print_r($content);
            echo "</XMP>";
        } else {
            var_dump($content);
        }
        if($die)
            exit;
    }

    /**
     * транслитерирует строку
     * @param $string - входная строка
     * @return mixed|string - результат транслитерации, кирилл.
```

```

СИМВОЛЫ => ЛАТИНСКИЕ, ПРОБЕЛЫ =>
*/
public static function transliterate($string) {
    $string = (string)$string;
    //массив альтернатив
    $rus = array(' ', 'а', 'б', 'в', 'г', 'д', 'е', 'ё', 'ж', 'з', 'и', 'й', 'к',
    'л', 'м', 'н', 'о', 'п', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ш', 'щ', 'ы', 'ъ',
    'э', 'ю', 'я', ' ', 'А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ё', 'Ж', 'З', 'И', 'Й', 'К',
    'Л', 'М', 'Н', 'О', 'П', 'С', 'Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ы', 'Ъ',
    'Э', 'Ю',
    $eng = array(' ', 'a', 'b', 'v', 'g', 'd', 'e', 'yo', 'zh', 'z', 'i', 'y', 'k',
    'l', 'm', 'n', 'o', 'p', 'r', 's', 't', 'u', 'f', 'h', 'c', 'ch', 'sh', 'sch', ' ', 'i', '',
    'e', 'yu', 'ya', ' ', 'A', 'B', 'V', 'G', 'D', 'E', 'Yo', 'j', 'Z', 'I', 'K', 'L',
    'M', 'N', 'O', 'P', 'R', 'S', 'T', 'U', 'F', 'H', 'C', 'Ch', 'Sh', 'Sch', ' ', 'I', '',
    'E',
    //заменим русские английскими
    $string = str_replace($rus, $eng, $string);
    return $string;
}

/** возвращает букву по ее позиции в алфавите
 * сейчас работает только для англ. алфавита и с позицией
не больше, чем кол-во символов алфавита
 * мне больше и не нужно
 * @param $iPos - позиция буквы
 * @return string - буква алфавита
*/
public static function findLetterByPos($iPos) {
    $iPos = (int)$iPos;
    // начальная позиция - начало алфавита
    if(!$iPos)
        $sLetter = '';
    // сдвинем ее на позицию нужной буквы
    for($i=1; $i <= $iPos; $i++)
        $sLetter++;
    return $sLetter;
}

/** собирает иерархическое дерево проблемных ситуаций в
цепочку
 * @param $iSituationId - id нижней ситуации
 * @return array - иерархия
*/
public static function buildHierarchy($iSituationId) {
    $iSituationId = (int)$iSituationId;
    if(!$iSituationId)
        return [];
    $aHierarchy = [];
    $oSituation = \Situation::find($iSituationId);
    if(!$oSituation)
        return [];
    // сначала запихнем сам этот раздел
    array_push($aHierarchy, ['id' => $oSituation->id, 'name' => $oSituation->name]);
    // пока есть предки будем собирать их в цепочку
    while($oSituation->parent()->get()->first()) {
        $oParent = $oSituation->parent()->get()->first();
        array_unshift($aHierarchy, ['id' => $oParent->id, 'name' => $oParent->name]);
    }
    $oSituation = $oParent;
}
return $aHierarchy;
}

App/classes/Elluminate/Engine/Search.php
<?php
/**
 * Created by PhpStorm.
 * User: elluminate
 * Date: 18.02.15
 * Time: 14:32
 */
namespace Illuminate\Engine;

/**
 * Class Search - реализует механизм поиска по приложению
 * @package Illuminate\Engine
 */
class Search

```

```

{
    /**
     * @var - текст, введенный пользователем для поиска
     */
    private $sSearchText;

    /**
     * - максимальное кол-во результатов по каждой группе
     */
    const GROUP_LIMIT = 10;

    /**
     * @var array - результаты поиска по коду ОКВЭД
     */
    private $aOkvedCodeResult;

    /**
     * @var - результаты поиска по названию проблемной
     * ситуации
     */
    private $aSituationNameResult;

    /**
     * @var - результаты поиска по названию решаемой задачи
     */
    private $aModelNameResult;

    /**
     * @var - флаг переполнения по коду ОКВЭД
     */
    private $bOkvedCodeOverlimit;

    /**
     * @var - флаг переполнения по названию проблемной
     * ситуации
     */
    private $bSituationNameOverlimit;

    /**
     * @var bool - флаг переполнения по названию решаемой
     * задачи
     */
    private $b modelNameOverLimit;

    /** геттер результатов поиска по коду ОКВЭД
     * @return mixed
     */
    public function getOkvedCodeResult()
    {
        return $this->aOkvedCodeResult;
    }

    /** геттер результатов по названию проблемной ситуации
     * @return mixed
     */
    public function getSituationNameResult()
    {
        return $this->aSituationNameResult;
    }

    /** геттер результатов по названию решаемой задачи
     * @return mixed
     */
    public function getModelNameResult()
    {
        return $this->aModelNameResult;
    }

    /** сеттер
     * @param mixed $sSearchText
     */
    public function setSearchText($sSearchText)
    {
        $this->sSearchText = $sSearchText;
    }

    /** геттер флага переполнения по коду ОКВЭД
     * @return mixed
     */
    public function getOkvedCodeOverlimit()
    {
        return $this->bOkvedCodeOverlimit;
    }

    /** геттер флага переполнения по названию проблемной
     * ситуации
     * @return mixed
     */
}

/*
public function getSituationNameOverlimit()
{
    return $this->bSituationNameOverlimit;
}

/** геттер флага переполнения по названию решаемой
 * задачи
 * @return mixed
 */
public function getModelNameOverLimit()
{
    return $this->bmodelNameOverLimit;
}

public function __construct()
{
    // инициализируем и обнулим результаты поиска
    $this->aOkvedCodeResult = $this->aSituationNameResult = $this->amodelNameResult = [];
    $this->bmodelNameOverLimit = $this->bOkvedCodeOverlimit = $this->bSituationNameOverlimit = false;
}

/**
 * реализует поиск по приложению
 */
public function startSearch()
{
    // ищем по названию решаемой задачи
    $this->amodelNameResult = $this->findBy modelName();
    // по названию проблемной ситуации
    $this->aSituationNameResult = $this->findBySituationName();
    // по коду ОКВЭД
    $this->aOkvedCodeResult = $this->findByOkvedCode();

    /**
     * ищет по названию решаемой задачи
     * @return array|\Illuminate\Database\Eloquent\Collection|static[]
     */
    private function findBy modelName()
    {
        // посчитаем сколько всего записей подошло под условия
        $iCount = \Model::where('name', 'LIKE', "%$this->sSearchText%")->count('id');
        // если больше, чем надо для вывода, установим флаг
        if ($iCount > self::GROUP_LIMIT) $this->bmodelNameOverLimit = true;
        return \Model::where('name', 'LIKE', "%$this->sSearchText%")->where('threshold', '>=', DB::raw('min_threshold'))->take(self::GROUP_LIMIT)->get(['id', 'name']);
    }

    /**
     * ищет по названию проблемной ситуации
     * @return array|\Illuminate\Database\Eloquent\Collection|static[]
     */
    private function findBySituationName()
    {
        // посчитаем сколько всего записей подошло под условия
        $iCount = \Situation::where('name', 'LIKE', "%$this->sSearchText%")->count('id');
        // если больше, чем надо для вывода, установим флаг
        if ($iCount > self::GROUP_LIMIT) $this->bSituationNameOverlimit = true;
        return \Situation::where('name', 'LIKE', "%$this->sSearchText%")->take(self::GROUP_LIMIT)->get();
    }

    /**
     * ищет по коду ОКВЭД
     * @return array|\Illuminate\Database\Eloquent\Collection|static[]
     */
    private function findByOkvedCode()
    {
        // посчитаем сколько всего записей подошло под условия
        $iCount = \Situation::where('okved_correspondence', 'LIKE', "%$this->sSearchText%")->count('id');
        // если больше, чем надо для вывода, установим флаг
        if ($iCount > self::GROUP_LIMIT) $this->bOkvedCodeOverlimit = true;
        return \Situation::where('okved_correspondence', 'LIKE', "%$this->sSearchText%")->take(self::GROUP_LIMIT)->get();
    }
}

```

```

<?php
/**
 *      Created       by      PhpStorm.
 *      User:        Date:    elluminate
 *      Date:        Time:    28.12.14
 *      Time:        1:07
 */
namespace Elluminate\Math;

/** аппарат логистической регрессии
 * Class @package LogisticRegression
 */
class LogisticRegression extends MathCore
{
    /**
     * максимальное кол-во регрессоров
     */
    const MAX_COVARIATES_NUM = 12;

    /**
     * максимальное кол-во экспериментов
     */
    const MAX_EXPERIMENTS_NUM = 10000;

    /**
     * минимальное кол-во регрессоров
     */
    const MIN_COVARIATES_NUM = 1;

    /**
     * степень покрытия регрессоров экспериментами
     */
    const EXPERIMENTS_COVERAGE = 10;

    /**
     * максимальное кол-во раз ухудшения найденных коэффициентов, после которого можно считать, что метод расходится
     */
    const MAX_WORSE_TIMES = 4;

    /**
     * максимальное кол-во итераций метода Ньютона-Рафсона
     */
    const MAX_NEWTON_RAPHSON_ITERATIONS = 1000;

    /**
     * точность метода Ньютона-Рафсона
     */
    const EPSILON = 0.001;

    /**
     * величина скачка метода Ньютона-Рафсона, когда считаем, что метод разошелся
     */
    const JUMP_VALUE = 1000;

    /**
     * минимальный порог отсечения по умолчанию
     */
    const DEFAULT_MIN_THRESHOLD = 75;

    /**
     * @var MathCore - математическое ядро
     */
    protected $oMath;

    /**
     * @var - кол-во регрессоров
     */
    private $iCovariatesNum;

    /**
     * @var - кол-во экспериментов
     */
    private $iExperimentsNum;

    /**
     * @var - массив регрессоров
     */
    private $aCovariates;
}

```

```

    /**
     * @var - вектор значений регрессии
     */
    private $aRegression;

    /**
     * @var - вектор коэффициентов модели
     */
    private $aCoefficients;

    /**
     * геттер для коэффициентов модели mixed
     */
    public function getCoefficients()
    {
        return $this->aCoefficients;
    }

    /**
     * геттер вектора значений функции mixed
     */
    public function getRegression()
    {
        return $this->aRegression;
    }

    /**
     * геттер для регрессоров mixed
     */
    public function getCovariates()
    {
        return $this->aCovariates;
    }

    public function __construct(MathCore $oMath)
    {
        // внедри зависимость - мат. ядро
        $this->oMath = $oMath;
    }

    /**
     * внедряет обучающую выборку, если она прошла проверку
     * @param $aTrainingSet - массив с обучающей выборкой
     * @throws \Illuminate\Exceptions\TrainSetFileNotFoundException
     */
    public function setTrainingSet($aTrainingSet)
    {
        // проверим выборку на соответствие нашим правилам
        $bCheckResult = $this->checkTrainingSet($aTrainingSet);
        // если проверка прошла успешно, то выделяем регрессоры и функцию
        if ($bCheckResult)
            $this->separateRegressionAndCovariates($aTrainingSet);
        unset($aTrainingSet);
    }

    /**
     * разделяет регрессоры и значения функции
     * @param $aTrainingSet - массив обучающей выборки
     * @return bool - результат разделения
     */
    private function separateRegressionAndCovariates($aTrainingSet)
    {
        $aRegression = [];
        $aCovariates = [];
        foreach ($aTrainingSet as $key => $value) {
            // регрессия - первый столбец
            $aRegression[] = $value[0];
            // т.к. коэффициентов на 1 больше (есть свободный член), то надо заполнить первый столбец // регрессоров искусственными данными, рекомендуется единица
            $value[0] = 1;
            // регрессоры - все остальное
            $aCovariates[] = $value;
        }
        unset($aTrainingSet);
        $this->aCovariates = $aCovariates;
        unset($aCovariates);
        $this->aRegression = $aRegression;
        unset($aRegression);
        return true;
    }

    /**
     * обучаем модель, ищем коэффициенты
     * @throws \Illuminate\Exceptions\SingularException
     */

```

```

/*
public function trainModel()
{
    $aCoefficients = $this->computeCoefficients();
    if ($aCoefficients && count($aCoefficients) == $aCoefficients)
        $this->aCoefficients = $aCoefficients;
}

/** ищет коэффициенты регрессии методом Ньютона-Рафсона
 * @return array|bool - вектор коэффициентов или false, если
 * получилось
 *      @throws \Illuminate\Exceptions\MathException
 *      @throws \Illuminate\Exceptions\SingularException
 */
public function computeCoefficients()
{
    // создаем начальное приближение
    $aCoeffVector = $this->oMath->constructVector($this->iCovariatesNum + 1);
    // эти коэффициенты на данный момент лучшие
    $aFinalCoeffVector = $aCoeffVector;
    // посчитаем значение функции при таких коэффициентах
    $aActualRegVector = $this->constructActualFuncVector($this->aCovariates, $aCoeffVector);
    // посчитаем СКО между нашими расчетами и функцией
    // из обучающей выборки
    $fMse = $this->oMath->meanSqError($aActualRegVector, $this->aRegression);
    // пока коэффициенты хуже не стали
    $iWorseTimes = 0;
    for ($i = 0; $i < self::MAX_NEWTON_RAPHSON_ITERATIONS; ++$i)
    {
        // посчитаем новый вектор коэффициентов
        $aNextCoeffVector = $this->constructNextCoeffVector($aCoeffVector, $this->aCovariates, $this->aRegression, $aActualRegVector);
        // там может не найтись обратной матрицы, тогда
        // обучение не удалось, так может случиться в редких случаях
        if (!$aNextCoeffVector) throw new \Illuminate\Exceptions\SingularException("Определитель
матрицы равен 0, поиск коэффициентов невозможен. Так
бывает при наличии зависимости между регрессорами.
Проверьте обучающую выборку и повторите ошибку");
        // если на последующем шаге коэффициенты не
        // изменились с заданной точностью, то считаем, что нашли
        // лучшие
        if (!$this->checkChanging($aCoeffVector, $aNextCoeffVector)) return $aFinalCoeffVector;
        // если произошел прыжок, то метод начал
        // расходиться, отдадим коэффициенты, которые получили до
        // расхождения, они получше должны быть
        if ($this->checkJumping($aCoeffVector, $aNextCoeffVector)) return $aFinalCoeffVector;
        // посчитаем функцию с коэффициентами, которые
        // лучше к этому шагу
        $aActualRegVector = $this->constructActualFuncVector($this->aCovariates, $aNextCoeffVector);
        // проверим СКО
        $fNextMse = $this->meanSqError($aActualRegVector, $this->aRegression);
        // если СКО стало больше
        if ($fNextMse > $fMse)
        {
            // значит мы ухудшаем наши коэффициенты
            ++$iWorseTimes;
            // если ухудшаем уже долго, то пора вернуть хоть
            // какие-то, пока не стало совсем плохо
            if ($iWorseTimes > self::MAX_WORSE_TIMES)
                return $aFinalCoeffVector;
            // $aCoeffVector = $aNextCoeffVector;
            // если ухудшаем не очень давно, то возьмем
            // средние коэффициенты между шагами, может метод еще
            // сойдется
            for ($k = 0; $k < count($aCoeffVector); ++$k)
                $aCoeffVector[$k] = ($aCoeffVector[$k] +
                $aNextCoeffVector[$k]) / 2;
            // если коэффициенты стали лучше
            $aCoeffVector = $aNextCoeffVector;
            $aFinalCoeffVector = $aCoeffVector;
            // обнулим счетчик ухудшений
            $iWorseTimes = 0;
        }
        // обновим СКО для следующей итерации
        $fMse = $fNextMse;
    }
    return $aFinalCoeffVector;
}

*/
    }

    /**
     * проверяет вектор коэффициентов на наличие изменений
     * @param $aOldCoeffVector - старый вектор
     * @param $aNewCoeffVector - новый вектор
     * @return bool - истина, если изменения есть; иначе - ложь
     */
    private function checkChanging($aOldCoeffVector, $aNewCoeffVector)
    {
        $iRows = count($aOldCoeffVector);
        for ($i = 0; $i < $iRows; ++$i)
            if (abs($aOldCoeffVector[$i] - $aNewCoeffVector[$i]) > self::EPSILON)
                return true;
        return false;
    }

    /**
     * проверяет не произошел ли прыжок из-за расхождения
     * метода
     * @param $aOldCoeffVector - старый вектор коэффициентов
     * @param $aNewCoeffVector - новый вектор коэффициентов
     * @return bool - истина, если произошел прыжок, иначе - ложь
     */
    private function checkJumping($aOldCoeffVector, $aNewCoeffVector)
    {
        $iRows = count($aOldCoeffVector);
        for ($i = 0; $i < $iRows; ++$i)
            if ($aOldCoeffVector[$i] === 0) return false;
            if (abs($aOldCoeffVector[$i] - $aNewCoeffVector[$i]) > abs($aOldCoeffVector[$i])) return self::JUMP_VALUE;
        return true;
    }

    /**
     * строит новый вектор коэффициентов методом
     * максимального правдоподобия
     * @param $aCoeffVector - старый вектор коэффициентов
     * @param $aCovMatrix - матрица регрессоров
     * @param $aRegVector - вектор функции
     * @param $aActualRegVector - вектор функции, которая
     * посчитана с актуальными лучшими коэффициентами
     * @return array|bool - вектор коэффициентов или ложь, если
     * матрица оказалась сингулярной
     */
    private function constructNextCoeffVector($aCoeffVector, $aCovMatrix, $aRegVector, $aActualRegVector)
    {
        // $aTranspCovMatrix = $this->oMath-
        // >matrixTranspose($aCovMatrix);
        // $a = $this->computeXtilde($aActualRegVector, $aCovMatrix);
        // $b = $this->oMath->matrixProduct($aTranspCovMatrix, $a);
        // $c = $this->oMath->matrixInverse($b);
        // вдруг там сингулярная матрица, тогда вернем ложь,
        // выше будет перехвачено и выброшено исключение
        if (!$c) return false;
        // $d = $this->oMath->matrixProduct($c, $aTranspCovMatrix);
        // $aSubtract = $this->oMath->vectorSubtraction($aRegVector,
        // $aActualRegVector);
        // $e = $this->oMath->matrixVectorProduct($d, $aSubtract);
        // $b + inverse($X'WX)X'(y-p)
        // $aResult = $this->oMath->vectorAddition($aCoeffVector, $e);
        // return $aResult;
    }

    /**
     * считает логистическую регрессию
     * @param $aCovMatrix - матрица регрессоров
     * @param $aCoeffVector - вектор коэффициентов
     * @return array - вектор логистической регрессии
     */
    private function constructActualFuncVector($aCovMatrix, $aCoeffVector)
    {
        $aResult = $this->oMath->constructVector($this->iExperimentsNum);
        for ($i = 0; $i < $this->iExperimentsNum; ++$i)
            $z = 0;
    }
}

```

```

        for ($j = 0; $j < $this->iCovariatesNum + 1; ++$j)
            $z += $aCovMatrix[$i][$j] * $aCoeffVector[$j];
            $p = 1 / (1 + exp(-$z));
            $aResult[$i] = $p;
    }
    return $aResult;
}

/** проверяет обучающую выборку на соответствие
 * требованиям
 * @param $aTrainingSet - массив обучающей выборки
 * @return bool - результат проверки, истина - все
 * нормально, ложь - что-то не так
 * @throws \Illuminate\Exceptions\TrainSetFileNotFoundException - исключение
 */
private function checkTrainingSet($aTrainingSet)
{
    foreach ($aTrainingSet as $row)
        foreach ($row as $value)
            //если не заполнено или не является числом
            if (!isset($value) || !is_numeric($value))
                throw new \Illuminate\Exceptions\TrainSetFileNotFoundException("Проверьте, что все поля обучающей выборки заполнены и являются числами.");
    // один столбец - значение функции, поэтому регрессоров
    // на 1 меньше
    $this->iCovariatesNum = count($aTrainingSet[0]) - 1;
    $this->iExperimentsNum = count($aTrainingSet);
    // проверяем кол-во регрессоров
    if ($this->iCovariatesNum > self::MAX_COVARIATES_NUM || $this->iCovariatesNum < 1)
        throw new \Illuminate\Exceptions\TrainSetFileNotFoundException("Количество параметров регрессии не должно быть больше " . self::MAX_COVARIATES_NUM);
    // проверяем кол-во экспериментов
    if ($this->iExperimentsNum < $this->iCovariatesNum * self::EXPERIMENTS_COVERAGE || $this->iExperimentsNum > self::MAX_EXPERIMENTS_NUM)
        throw new \Illuminate\Exceptions\TrainSetFileNotFoundException("Количество опытов должно быть не меньше " . $this->iCovariatesNum * self::EXPERIMENTS_COVERAGE . " и не больше " . self::MAX_EXPERIMENTS_NUM);
    return true;
}

/** считает WX для метода максимального правдоподобия
 * @param $aActualRegVector - вектор функции, посчитанный
 * актуальными коэффициентами
 * @param $aCovMatrix - матрица регрессоров
 * @return array - WX
 * @throws \Illuminate\Exceptions\MathException
 */
private function computeXtilde($aActualRegVector,
    $aCovMatrix)
{
    $pRows = count($aActualRegVector);
    $xRows = count($aCovMatrix);
    $xCols = count($aCovMatrix[0]);
    if ($pRows != $xRows)
        throw new \Illuminate\Exceptions\MathException("Несоответствие размерностей матриц при попытке построить X");
    $result = $this->oMath->constructMatrix($pRows, $xCols);
    for ($i = 0; $i < $pRows; ++$i)
        for ($j = 0; $j < $xCols; ++$j)
            $result[$i][$j] = $aActualRegVector[$i] * (1 - $aActualRegVector[$i]) * $aCovMatrix[$i][$j];
    return $result;
}

/** считает значения функции
 * @return array
 */
public function estimate()
{
    if(isset($this->aCoefficients) && count($this->aCoefficients))
        return $this->constructActualFuncVector($this->aCovariates, $this->aCoefficients);
    return [];
}

```

App/classes/Elluminate/Math/MathCore.php

```

<?php
/*
*      Created   User:      by      PhpStorm.
*      Date:     Time:    elluminate
*                                         06.01.15
*                                         13:50
*/
namespace Elluminate\Math;

/**
*      Class      @package      MathCore
*                                         Elluminate\Math
*/
class MathCore
{
    /**
*      создает вектор
*      @param $iSize - размер
*      @param int $uValue - значение для заполнения
*      @return array - вектор
*/
public function constructVector($iSize, $uValue = 0)
{
    $iSize = ($iSize) ? $iSize : 0;
    $aVector = array_fill(0, $iSize, $uValue);
    return $aVector;
}

/**
*      создает матрицу
*      @param $iRowsNum - число строк
*      @param $iColsNum - число столбцов
*      @param int $uValue - значение для заполнения
*      @return array - матрица
*/
public function constructMatrix($iRowsNum, $iColsNum, $uValue = 0)
{
    $iRowsNum = ($iRowsNum) ? $iRowsNum : 0;
    $iColsNum = ($iColsNum) ? $iColsNum : 0;
    $aResult = array_fill(0, $iRowsNum, array_fill(0, $iColsNum, $uValue));
    return $aResult;
}

/**
*      считает произведение матриц
*      @param $aMatrix1 - матрица 1
*      @param $aMatrix2 - матрица 2
*      @return array - результат произведения матриц
*      @throws \Illuminate\Exceptions\MathException
*/
public function matrixProduct($aMatrix1, $aMatrix2)
{
    $iRows1 = count($aMatrix1);
    $iCols1 = count($aMatrix1[0]);
    $iRows2 = count($aMatrix2);
    $iCols2 = count($aMatrix2[0]);
    if ($iCols1 != $iRows2)
        throw new \Illuminate\Exceptions\MathException("Несоответствие размеров матриц при перемножении");
    $aResult = $this->constructMatrix($iRows1, $iCols2);
    for ($i = 0; $i < $iRows1; ++$i)
        for ($j = 0; $j < $iCols2; ++$j)
            for ($k = 0; $k < $iCols1; ++$k)
                $aResult[$i][$j] += $aMatrix1[$i][$k] * $aMatrix2[$k][$j];
    return $aResult;
}

/**
*      считает произведение матрицы на вектор
*      @param $aMatrix - матрица
*      @param $aVector - вектор
*      @return array - результат произведения
*      @throws \Illuminate\Exceptions\MathException
*/
public function matrixVectorProduct($aMatrix, $aVector)
{
    $iMRows = count($aMatrix);
    $iMCols = count($aMatrix[0]);
    $iVRows = count($aVector);
    if ($iMCols != $iVRows)
        throw new \Illuminate\Exceptions\MathException("Несоответствие размеров матриц при перемножении");
    $aResult = $this->constructVector($iMRows);

```

```

for ($i = 0; $i < $iMRows; ++$i)
    for ($j = 0; $j < $iMCols; ++$j)
        $aResult[$i] += $aMatrix[$i][$j] * $aVector[$j];
return $aResult;
}

/** транспонирует матрицу
 * @param $aMatrix - входная матрица
 * @return array - результат транспонирования
 * @throws \Illuminate\Exceptions\MathException
 */
public function matrixTranspose($aMatrix)
{
    if (!is_array($aMatrix) || !count($aMatrix)) throw new
    \Illuminate\Exceptions\MathException("Ошибка при
транспонировании
матрицы");
    $result = array();
    $last = sizeof($aMatrix) - 1;
    eval('$result = array_map(null, $aMatrix['
        . implode('', $aMatrix['range(0, $last)) . ']);');
    return $result;
}

/** ищет обратную матрицу
 * @param $aMatrix - входная матрица
 * @return array|bool - обратная матрица или false, если
найти не получается
 * @throws \Illuminate\Exceptions\MathException
 */
public function matrixInverse($aMatrix)
{
    if (!is_array($aMatrix) || !count($aMatrix)) throw new
    \Illuminate\Exceptions\MathException("Ошибка при поиске
обратной
матрицы");
    $iToggel = 0;
    $aPerm = [];
    $aLum = $this->matrixDecompose($aMatrix, $aPerm,
    &$iToggel);
    $iRows = count($aMatrix);
    if (!$aLum) return false;
    $aB = [];
    $aResult = [];
    for ($i = 0; $i < $iRows; ++$i) {
        for ($j = 0; $j < $iRows; ++$j) {
            if ($i == $j) $aB[$j] = 1;
            else $aB[$j] = 0;
        }
        $x = $this->equationSolver($aLum, $aB);
        for ($j = 0; $j < $iRows; ++$j)
            $aResult[$j][$i] = $x[$j];
    }
    return $aResult;
}

/** раскладывает матрицу
 * @param $aMatrix - входная матрица
 * @param $aPerm
 * @param $iToggel
 * @return bool - разложенная матрица или false, если
разложить не получилось
 * @throws \Illuminate\Exceptions\MathException
 */
private function matrixDecompose($aMatrix, &$aPerm,
&$iToggel)
{
    $iRows = count($aMatrix);
    $iCols = count($aMatrix[0]);
    if ($iCols != $iRows) throw new
    \Illuminate\Exceptions\MathException("Попытка
разложить
матрицу, которая не является квадратной");
    for ($i = 0; $i < $iRows; ++$i)
        $aPerm[$i] = $i;
    $iToggel = 1;
    $aResult = $aMatrix;
    for ($j = 0; $j < $iRows - 1; ++$j) {
        $max = abs($aResult[$j][$j]);
        $pRow = $j;
        for ($i = $j + 1; $i < $iRows; ++$i) {
            $aij = $aResult[$i][$j];
            if ($aij > $max) {
                $max = $aij;
                $pRow = $i;
            }
        }
        if ($pRow != $j) {
            $rowPtr = $aResult[$pRow];

```

```

            $aResult[$pRow] = $aResult[$j];
            $aResult[$j] = $rowPtr;
            $tmp = $aPerm[$pRow];
            $aPerm[$pRow] = $aPerm[$j];
            $aPerm[$j] = $tmp;
            $iToggel = -$iToggel;
        }
        $aij = abs($aResult[$j][$j]);
        if ($aij < 0.00000001) return false;
        for ($i = $j + 1; $i < $iRows; ++$i) {
            $aij = $aResult[$i][$j] / $aij;
            $aResult[$i][$j] = $aij;
            for ($k = $j + 1; $k < $iRows; ++$k)
                $aResult[$i][$k] -= $aij * $aResult[$j][$k];
        }
    }
    return $aResult;
}

/** решает уравнение Ax = B
 * @param $aLuMatrix - разложенная матрица
 * @param $aB - матрица B
 * @return mixed - найденная матрица B
 * @throws \Illuminate\Exceptions\MathException
 */
private function equationSolver($aLuMatrix, $aB)
{
    if (!is_array($aLuMatrix) || !count($aLuMatrix)) throw new
    \Illuminate\Exceptions\MathException("Ошибка при решении
уравнения
Ax=b");
    $iRows = count($aLuMatrix);
    // решает Ly = b прямой заменой
    for ($i = 1; $i < $iRows; ++$i) {
        $fSum = $aB[$i];
        for ($j = 0; $j < $i; $j++)
            $fSum -= $aLuMatrix[$i][$j] * $aB[$j];
        $aB[$i] = $fSum;
    }
    // решает Ux = y обратной заменой
    $aB[$iRows - 1] /= $aLuMatrix[$iRows - 1][$iRows - 1];
    for ($i = $iRows - 2; $i >= 0; -$i) {
        $fSum = $aB[$i];
        for ($j = $i + 1; $j < $iRows; ++$j)
            $fSum -= $aLuMatrix[$i][$j] * $aB[$j];
        $aB[$i] = $fSum / $aLuMatrix[$i][$i];
    }
    return $aB;
}

/** считает разность векторов
 * @param $aVector1 - вектор 1
 * @param $aVector2 - вектор 2
 * @return array - результат разности
 * @throws \Illuminate\Exceptions\MathException
 */
public function vectorSubtraction($aVector1, $aVector2)
{
    $iRows1 = count($aVector1);
    $iRows2 = count($aVector2);
    if ($iRows1 != $iRows2) throw new
    \Illuminate\Exceptions\MathException("Несоответствие
размерностей
векторов при вычитании");
    $aResult = [];
    for ($i = 0; $i < $iRows1; ++$i)
        $aResult[$i] = $aVector1[$i] - $aVector2[$i];
    return $aResult;
}

/** считает сумму векторов
 * @param $aVector1 - вектор 1
 * @param $aVector2 - вектор 2
 * @return array - результат суммирования
 * @throws \Illuminate\Exceptions\MathException
 */
public function vectorAddition($aVector1, $aVector2)
{
    $iRows1 = count($aVector1);
    $iRows2 = count($aVector2);
    if ($iRows1 != $iRows2) throw new
    \Illuminate\Exceptions\MathException("Несоответствие
размерностей
векторов при сложении");
    $aResult = [];
    for ($i = 0; $i < $iRows1; ++$i)
        $aResult[$i] = $aVector1[$i] + $aVector2[$i];
    return $aResult;
}

```

```


    считает разность числа и вектора
    @param $aVector - вектор
    @param $fNumber - число
    @return mixed - результирующий вектор
*/
public function vectorNumberSubtraction($aVector, $fNumber)
{
    if (is_array($aVector) && count($aVector) &&
        is_numeric($fNumber))
        foreach ($aVector as $value)
            $value = $fNumber - $value;
        return $aVector;
    } else throw new \Illuminate\Exceptions\MathException("Ошибка при поиске
разности числа и вектора");
}

    считает мат. ожидание матрицы
    @param $matrix - матрица
    @return float - мат. ожидание
    @throws \Illuminate\Exceptions\MathException
*/
public static function matrixMean($matrix)
{
    if (is_array($matrix) && count($matrix))
        return array_sum($matrix) / count($matrix);
    else throw new \Illuminate\Exceptions\MathException("Ошибка математического ожидания в матрице");
}

    считает СКО матрицы
    @param $matrix - матрица
    @return float - СКО
    @throws \Illuminate\Exceptions\MathException
*/
public static function matrixStdDiv($matrix)
{
    if (is_array($matrix) && count($matrix))
        return sqrt(array_sum(array_map("self::sdSquare", $matrix,
            array_fill(0, count($matrix), (array_sum($matrix) /
            count($matrix)))))) / (count($matrix) - 1));
    else throw new \Illuminate\Exceptions\MathException("Ошибка поиска СКО в матрице");
}

    считает разность матриц
    @param $matrix1 - матрица 1
    @param $matrix2 - матрица 2
    @return array - результат разности
    @throws \Illuminate\Exceptions\MathException
*/
public static function matrixDiff($matrix1, $matrix2)
{
    if (!is_array($matrix1) || !is_array($matrix2) ||
        count($matrix1) != count($matrix2)) throw new \Illuminate\Exceptions\MathException("Ошибка поиска разности матриц");
    $result = array();
    foreach ($matrix1 as $key => $value) {
        $result[$key] = abs($value - $matrix2[$key]);
    }
    return $result;
}

    ищет позицию минимума вектора
    @param $matrix - вектор
    @return mixed - минимальное значение
    @throws \Illuminate\Exceptions\MathException
*/
public static function findVectorMinimumPos($matrix)
{
    if (!is_array($matrix) || !count($matrix)) throw new \Illuminate\Exceptions\MathException("Ошибка поиска минимума в матрице");
    $result = array_keys($matrix, min($matrix));
    //может быть несколько одинаковых чисел, поэтому берем первое
    $last = sizeof($result) - 1;
    return $result[$last];
}

    считает СКО между двумя матрицами
    @param $matrix1 - матрица 1
    @param $matrix2 - матрица 2


```

```

    @return float|int - СКО
    @throws \Illuminate\Exceptions\MathException
*/
public static function meanSqError($matrix1, $matrix2)
{
    $pRows = count($matrix1);
    $yRows = count($matrix2);
    if ($pRows != $yRows) throw new \Illuminate\Exceptions\MathException("Несоответствие размерностей матриц при попытке найти СКО");
    if ($pRows == 0) return 0;
    $result = 0;
    for ($i = 0; $i < $pRows; ++$i)
        $result += ($matrix1[$i] - $matrix2[$i]) * ($matrix1[$i] -
        $matrix2[$i]);
    return $result / $pRows;
}

```

```

    считает квадрат разности, нужна для поиска СКО
    @param $x
    @param $mean
    @return number
*/
private static function sdSquare($x, $mean)
{
    return pow($x - $mean, 2);
}

    logisticRegression($aCovariates,
$aCoefficients)
{
    if(!is_array($aCovariates) || !is_array($aCoefficients) ||
!count($aCovariates) || !count($aCoefficients)) throw new \Illuminate\Exceptions\MathException("Ошибка при расчете значения логистической регрессии");
    if(count($aCovariates) != count($aCoefficients)-1) throw new \Illuminate\Exceptions\MathException("Количество коэффициентов не соответствует количеству регрессоров");
    $z = 0;
    array_unshift($aCovariates, 1);
    for ($j = 0; $j < count($aCovariates); ++$j)
        $z += $aCovariates[$j] * $aCoefficients[$j];
    $fResult = 1 / (1 + exp(-$z));
    return $fResult;
}

```

App/classes/Elluminate/Math/QualityAnalysis.php

```

<?php
/*
* Created by PhpStorm.
* User: elluminate
* Date: 06.01.15
* Time: 13:48
*/
namespace Illuminate\Math;

/**
* Class QualityAnalysis
* @package Illuminate\Math
*/
class QualityAnalysis
{
    /**
     * @var - модель, качество которой необходимо оценить
     */
    private $oModel;

    /**
     * @var MathCore - класс для работы с математическими функциями
     */
    private $oMath;

    /**
     * @var - вектор специфичности
     */
    private $aSpecificity;

    /**
     * @var - вектор чувствительности
     */
    private $aSensitivity;
}

```

```

* @var - вектор эластичных коэффициентов
*/
private $aElasticCoeff;

/** * @var - вектор стандартизованных коэффициентов
*/
private $aStdCoeff;

/** * @var - значение в пороге отсечения
*/
private $fThreshold;

/** * @var - порог отсечения
*/
private $fSill;

/** * @var - площадь под ROC-кривой
*/
private $fCurveArea;

/** геттер эластичных коэффициентов mixed
*/
public function getElasticCoeff()
{
    return $this->aElasticCoeff;
}

/** геттер стандартизованных коэффициентов mixed
*/
public function getStdCoeff()
{
    return $this->aStdCoeff;
}

/** геттер значения в пороге отсечения mixed
*/
public function getThreshold()
{
    return $this->fThreshold;
}

public function getSill()
{
    return $this->fSill;
}

/** геттер значения площади под ROC-кривой mixed
*/
public function getCurveArea()
{
    return $this->fCurveArea;
}

public function __construct(MathCore $oMath)
{
    $this->oMath = $oMath;
}

/** сеттер @param для модели $oModel
*/
public function setModel($oModel)
{
    $this->oModel = $oModel;
}

/** анализирует кач-во предоставленной модели
* @throws \Illuminate\Exceptions\MathException
*/
public function getQualityAnalysis()
{
    $this->calcSpecAndSens();
    $this->fThreshold = $this->findThreshold();
    $this->fCurveArea = $this->calcAreaUnderRocCurve();
    $this->findElasticAndStdCoeff();
    unset($this->aSensitivity);
    unset($this->aSpecificity);
    unset($this->oModel);
    unset($this->oMath);
}

```

```

/** считает чувствительность и специфичность bool
*/
private function calcSpecAndSens()
{
    // откуда начнем считать 0;
    // $iStartPoint с каким шагом 0.001;
    // $fStep где закончим 1;
    // $iEndPoint = 1;
    // посчитаем кол-во отрицательных и положительных исходов в выборке
    // $aOutcomes = $this->countOutcomes();
    // $iPositives = isset($aOutcomes['positives']) ? 0;
    // $iNegatives = isset($aOutcomes['negatives']) ? 0;
    // $aOutcomes['negatives'] : 0;
    $aOutcomes['positives'] : 0;
    $fRoller = $iStartPoint;
    // посчитаем функцию с нашими коэффициентами
    $aEstimatedFunction = $this->oModel->estimate();
    $j = 0;
    while ($fRoller <= $iEndPoint) {
        // обнулим счетчик ошибок
        $iFalsePositive = $iTruePositive = 0;
        if (is_array($aEstimatedFunction) && count($aEstimatedFunction))
            foreach ($aEstimatedFunction as $key => $value) {
                // если значение больше, чем текущий порог отсечения
                if ($value >= $fRoller) {
                    // если значение функции в обучающей выборке
                    // увеличиваем кол-во истинно-положительных
                    if (isset($this->oModel->getRegression())[$key]) && $this->oModel->getRegression()[$key] == 1)
                        $iTruePositive++;
                    else
                        // иначе увеличиваем кол-во ложно положительных
                        $iFalsePositive++;
                }
                if ($iTruePositive > 100)
                    $this->aSensitivity[$j] = $iTruePositive / $iPositives * 100;
                if ($iFalsePositive > 100)
                    $this->aSpecificity[$j] = $iFalsePositive / $iNegatives * 100;
                $j++;
                $fRoller += $fStep;
            }
        return true;
    }

    /** ищет значение в пороге отсечения mixed
    * @throws \Illuminate\Exceptions\MathException
    */
    private function findThreshold()
    {
        // найдем разность векторов чувствительности и 100 - специфичность
        $aDifference = $this->oMath->matrixDiff($this->oMath->vectorNumberSubtraction($this->aSpecificity, 100), $this->aSensitivity);
        // найдем минимум модуля разности
        $iSill = $this->oMath->findVectorMinimumPos($aDifference);
        $this->fSill = $iSill / 1000;
        unset($aDifference);
        // возьмем минимальное из двух значений, чтобы обеспечить качество, не ниже заявленного
        return min($this->oMath->vectorNumberSubtraction($this->aSpecificity, 100)[0], $this->aSensitivity[0]);
    }

    /** считает кол-во положительных и отрицательных исходов
    * @return array - массив с числом исходов
    */
    private function countOutcomes()
    {
        $iNegative = 0;
        $iPositive = 0;
        if (is_array($this->oModel->getRegression()) && count($this->oModel->getRegression()))
            foreach ($this->oModel->getRegression() as $value) {
                if ($value == 1)
                    $iPositive++;
            }
    }
}

```

```

        if ($value == 0)
    }
    return ['positives' => $iPositive, 'negatives' => $iNegative];
}

/** ищет эластичные и стандартизованные коэффициенты
 * @throws \Illuminate\Exceptions\MathException */
private function findElasticAndStdCoeff()
{
    // для свободного члена такие показатели не ищут,
    // уберем его
    $aCoefficients = array_slice($this->oModel->getCoefficients(), 1);
    $aTranspCov = $this->oMath->matrixTranspose($this->oModel->getCovariates());
    // уберем первый столбец, мы его искусственно заполняли
    // единицами, это свободный член
    $aTranspCov = array_slice($aTranspCov, 1);
    $i = 0;
    $aElasticCoeff = $aStdCoeff = [];
    if (!is_array($aTranspCov) || !count($aTranspCov))
        throw new \Illuminate\Exceptions\MathException("Ошибка
при попытке найти мат. ожидание и СКО для регрессоров");
    // найдем среднее для функции
    $aRegMean = $this->oMath->matrixMean($this->oModel->getRegression());
    // найдем СКО для функции
    $aRegStd = $this->oMath->matrixStdDiv($this->oModel->getRegression());
    foreach ($aTranspCov as $value) {
        // найдем среднее по столбцу для регрессоров
        $aCovMean = $this->oMath->matrixMean($value);
        // найдем СКО по столбцу для регрессоров
        $aCovStd = $this->oMath->matrixStdDiv($value);
        // посчитаем i-ый эластичный коэффициент
        $aElasticCoeff[$i] = $aCoefficients[$i] * $aCovMean /
        $aRegMean;
        // посчитаем i-ый стандартизованный коэффициент
        $aStdCoeff[$i] = $aCoefficients[$i] * $aCovStd / $aRegStd;
        $i++;
    }
    unset($aCoefficients);
    unset($aTranspCov);
    if (is_array($aElasticCoeff) && count($aElasticCoeff))
        $this->aElasticCoeff = $aElasticCoeff;
    if (is_array($aStdCoeff) && count($aStdCoeff))
        $this->aStdCoeff = $aStdCoeff;
}

/** считает площадь под ROC-кривой
 * @return float - площадь под ROC-кривой
 * @throws \Illuminate\Exceptions\MathException */
private function calcAreaUnderRocCurve()
{
    $iSensitivityCount = count($this->aSensitivity);
    $iSpecificityCount = count($this->aSpecificity);
    if (($iSensitivityCount != $iSpecificityCount) ||
    !$iSensitivityCount)
        throw new \Illuminate\Exceptions\MathException("Ошибка
при попытке посчитать площадь под ROC-кривой");
    $i = 0;
    $fSum = 0;
    // посчитаем интеграл
    while ($i <= $iSensitivityCount - 2) {
        $fSum += abs($this->aSpecificity[$i + 1] - $this-
>aSpecificity[$i]) * ((($this->aSensitivity[$i + 1] + $this-
>aSensitivity[$i]) / 2));
        $i++;
    }
    // представим в виде 0,00-100,00
    $fArea = round($fSum / 100, 2);
    return $fArea;
}

```

App/classes/Elluminate/Workers/ModelTrainer.php

```

<?php
/**
 * Created by PhpStorm.
 * User: elluminate
 * Date: 19.02.15
 * Time: 16:53
 */
namespace Elluminate\Workers;

```

```

class ModelTrainer extends ModelRepo
{
    protected $oModelRepo;

    public function __construct(\ClientModelRepository $oModelRepo)
    {
        $this->oModelRepo = $oModelRepo;
    }

    public function fire($job, $data)
    {
        $iEvaluationId = (int)$data['iEvaluationId'];
        // если нет такого вычисления, то просто удалим это
        // задание
        if (!$iEvaluationId || !$Evaluation::find($iEvaluationId, ['id']))
            $job->delete();
        $oEvaluation = \Evaluation::find($iEvaluationId);
        // получим реальный результат и введенные
        // пользователем параметры из БД
        $iRealResult = $oEvaluation->real_result;
        // восстановим массив параметров
        $aCovariates = json_decode($oEvaluation->covariates);
        $aRow = $aCovariates;
        unset($aCovariates);
        // получается строка {Значение
        // функции}{Регрессор1}...{РегрессорN} - в общем как в
        // обучающей выборке
        array_unshift($aRow, $iRealResult);
        $iModelId = $oEvaluation->model_id;
        $oModel = \Model::find($iModelId);
        // восстановим массив доп. выборки, или если там пусто,
        // будем писать с нуля
        $aOversampling = is_array(json_decode($oModel-
>oversampling)) ? json_decode($oModel->oversampling) : array();
        // допишем в доп. выборку строчку из нашего вычисления
        array_push($aOversampling, $aRow);
        unset($aRow);
        $oModel->oversampling = json_encode($aOversampling);
        $oModel->save();
        // переобучим модель
        $this->oModelRepo->retrainModel($iModelId);
        // теперь удаляем сущность вычисления, она больше не
        // пригодится
        $oEvaluation->delete();
        // удаляем задачу из очереди
        $job->delete();
    }
}

```

App/controllers/BaseController.php

```

<?php
class BaseController extends Controller
{
    /**
     * Setup the layout used by the controller.
     *
     * @return void
     */
    protected function setupLayout()
    {
        if (!is_null($this->layout))
        {
            $this->layout = View::make($this->layout);
        }
    }
}

```

App/controllers/UsersController.php

```

<?php
class UsersController extends Controller
{
    /**
     * Implements actions regarding user management
     */
    class UsersController extends Controller
    {
        /**
         * Displays the form for account creation
         *
         * @return Illuminate\Http\Response
         */
    }
}

```

```

public function create()
{
    return View::make(Config::get('confide::signup_form'));
}

/**
 * Stores new account
 */
public function store()
{
    $repo = App::make('UserRepository');
    $user = $repo->signup(Input::all());

    if ($user->id) {
        if (Config::get('confide::signup_email')) {
            Mail::queueOn(
                Config::get('confide::email_queue'),
                Config::get('confide::email_account_confirmation'),
                compact('user'),
                function ($message) use ($user) {
                    $message
                        ->to($user->email, $user->username)
                        ->subject(Lang::get('confide::confide.email.account_confirmation.subject'));
                });
        }
    }

    return Redirect::action('UsersController@login')
        ->with('notice', Lang::get('confide::confide.alerts.account_created'));
} else {
    $error = $user->errors()->all(':message');
}

return Redirect::action('UsersController@create')
    ->withInput(Input::except('password'))
    ->with('error', $error);
}

/**
 * Displays the login form
 */
public function login()
{
    if (Auth::user()) {
        return Redirect::to('/');
    } else {
        return View::make(Config::get('confide::login_form'));
    }
}

/**
 * Attempt to do login
 */
public function doLogin()
{
    $repo = App::make('UserRepository');
    $input = Input::all();

    if ($repo->login($input)) {
        return Redirect::intended('/');
    } else {
        $err_msg = Lang::get('confide::confide.alerts.too_many_attempts');
        if ($repo->existsButNotConfirmed($input)) {
            $err_msg = Lang::get('confide::confide.alerts.not_confirmed');
        } else {
            $err_msg = Lang::get('confide::confide.alerts.wrong_credentials');
        }
    }

    return Redirect::action('UsersController@login')
        ->withInput(Input::except('password'))
        ->with('error', $err_msg);
}

/**
 * Attempt to confirm account with code
 */
public function confirm($code)
{
    if (Confide::confirm($code)) {
        $notice_msg = Lang::get('confide::confide.alerts.confirmation');
        return Redirect::action('UsersController@login')
            ->with('notice', $notice_msg);
    } else {
        $error_msg = Lang::get('confide::confide.alerts.wrong_confirmation');
        return Redirect::action('UsersController@login')
            ->with('error', $error_msg);
    }
}

/**
 * Displays the forgot password form
 */
public function forgotPassword()
{
    return View::make(Config::get('confide::forgot_password_form'));
}

/**
 * Attempt to send change password link to the given email
 */
public function doForgotPassword()
{
    if (Confide::forgotPassword(Input::get('email'))) {
        $notice_msg = Lang::get('confide::confide.alerts.password_forgot');
        return Redirect::action('UsersController@login')
            ->with('notice', $notice_msg);
    } else {
        $error_msg = Lang::get('confide::confide.alerts.wrong_password_forgot');
        return Redirect::action('UsersController@doForgotPassword')
            ->withInput()
            ->with('error', $error_msg);
    }
}

/**
 * Shows the change password form with the given token
 */
public function resetPassword($token)
{
    return View::make(Config::get('confide::reset_password_form'))
        ->with('token', $token);
}

/**
 * Attempt change password of the user
 */
public function doResetPassword()
{
    $repo = App::make('UserRepository');
    $input = Input::all();
    $token = Input::get('token');
    $password = Input::get('password');
    $password_confirmation = Input::get('password_confirmation');

    if ($repo->resetPassword($input)) {
        $notice_msg = Lang::get('confide::confide.alerts.password_reset');
    } else {
        $error_msg = Lang::get('confide::confide.alerts.reset_password_error');
    }
}

```

```

        return Redirect::action('UsersController@login')
            ->with('notice', $notice_msg);
    }
    $error_msg = Lang::get('confide::confide.alerts.wrong_password_reset');
    return Redirect::action('UsersController@resetPassword',
array('token'=>$input['token']))
        ->withInput()
        ->with('error', $error_msg);
}

/**
 * Log the user out of the application.
 */
@return Illuminate\Http\Response
public function logout()
{
    Confide::logout();
    return Redirect::to('/');
}

```

App/controllers/admin/AdminModelController.php

```

<?php

/** Админский контроллер моделей (решаемых задач)
 * Class AdminModelController
 */
class AdminModelController extends \BaseController

{
    /**
     * @var AdminModelRepository - админский репозиторий
     * моделей
     */
    protected $oRepo;

    public function __construct(AdminModelRepository $oRepo)
    {
        $this->oRepo = $oRepo;
    }

    /**
     * @param $iSituationId - id родительской ситуации
     * @return \Illuminate\View\View
     */
    public function index($iSituationId)
    {
        // получим список моделей
        $oModels = $this->oRepo->getModelsList($iSituationId);

        $aHierarchy = \Illuminate\Engine\E::buildHierarchy($iSituationId);
        // отдадим вид
        return View::make('admin.models.index', [
            'models' => $oModels,
            'situation_id' => $iSituationId,
            'hierarchy' => $aHierarchy
        ]);
    }

    /**
     * показывает форму добавления новой модели
     * @param $iSituationId - id ситуации, в которую надо
     * добавить
     * @return \Illuminate\View\View
     */
    public function create($iSituationId)
    {
        $iSituationId = (int)$iSituationId;
        if (!$iSituationId || !Situation::find($iSituationId, ['id'])) throw
new \Symfony\Component\HttpKernel\Exception\NotFoundHttpException;
        $aHierarchy = \Illuminate\Engine\E::buildHierarchy($iSituationId);
        $oDurations = Duration::lists('name', 'id');
        return View::make('admin.models.create', [
            'situation_id' => $iSituationId,
            'hierarchy' => $aHierarchy,
            'durations' => $oDurations
        ]);
    }

    /**
     * сохраняет новую модель
     */

```

```

        * @return $this
        */
    public function store()
    {
        $iSituationId = (int)Input::get('situation_id');
        if (!$iSituationId || !Situation::find($iSituationId, ['id'])) throw
new \Symfony\Component\HttpKernel\Exception\NotFoundHttpException;
        $sName = Input::get('name');
        $iDuration = (int)Input::get('duration');
        $iMinThreshold = (int)Input::get('min_threshold');
        $sComment = Input::get('comment');
        $fTrainFile = Input::file('train_file');
        $oValidation = Model::validate([
            'name' => $sName,
            'duration' => $iDuration,
            'min_threshold' => $iMinThreshold,
            'comment' => $sComment,
            'train_file' => $fTrainFile,
            'situation_id' => $iSituationId
        ]);
        // если валидация провалилась, редиректим обратно с
        // ошибками и заполненными полями
        if ($oValidation->fails())
            return Redirect::route('models.create', ['iSituationId' =>
$iSituationId])->withErrors($oValidation)->withInput();
        $bResult = $this->oRepo->storeModel($iSituationId, $sName,
        $iDuration, $iMinThreshold, $sComment, $fTrainFile);
        if ($bResult)
            return Redirect::route('models.list', ['iSituationId' =>
$iSituationId])->withForm_result('addDone');
        else
            return Redirect::route('models.list', ['iSituationId' =>
$iSituationId])->withForm_result('addFailed');
    }

    /**
     * показывает информацию о модели
     * @param $iModelId - id модели
     * @return \Illuminate\View\View
     */
    public function show($iModelId)
    {
        $oModel = $this->oRepo->getModelDetail($iModelId);
        $aHierarchy = \Illuminate\Engine\E::buildHierarchy($oModel->situation_id);
        return View::make('admin.models.detail', [
            'model' => $oModel,
            'hierarchy' => $aHierarchy
        ]);
    }

    /**
     * удаляет модель
     * @param $iModelId - id модели
     * @return mixed
     */
    public function destroy($iModelId)
    {
        // если нас хотят обмануть или такой модели просто нет,
        // отдаем 404 ошибку
        if (!$iModelId || !Model::find($iModelId)) throw
new \Symfony\Component\HttpKernel\Exception\NotFoundHttpException;
        // получим родителя удаляемой модели, чтобы знать куда
        // вернуть пользователя при редиректе
        $iSituationId = isset(Model::find($iModelId)->situation()->get()->first()->id) ?
(int)Model::find($iModelId)->situation()->get()->first()->id : 0;
        $bResult = $this->oRepo->destroyModel($iModelId);
        // магия - передает в вид переменную form_result - какой
        // шаблон отрисовать в качестве результата удаления
        if ($bResult)
            return Redirect::route('models.list', ['iSituationId' =>
$iSituationId])->withForm_result('delDone');
        else
            return Redirect::route('models.list', ['iSituationId' =>
$iSituationId])->withForm_result('delFailed');
    }

    public function edit($iModelId)
    {
        $oModel = $this->oRepo->getModel($iModelId, ['id',
'situation_id', 'name', 'comment', 'durations_id', 'min_threshold']);
        // собираем дерево родителей для хлебных крошек
        $aHierarchy = \Illuminate\Engine\E::buildHierarchy($oModel->situation_id);
        $oDurations = Duration::lists('name', 'id');
        // рисуем
    }
}
```

```

        return View::make('admin.models.edit', [
            'hierarchy' => $aHierarchy,
            'model' => $oModel,
            'durations' => $oDurations
        ]);
    }

    public function update() {
        $iModelId = Input::get('model_id');
        if (!$iModelId || !Model::find($iModelId)) throw new \Symfony\Component\HttpKernel\Exception\NotFoundHttpException;
        $sName = Input::get('name');
        $iDuration = (int)Input::get('duration');
        $iMinThreshold = (int)Input::get('min_threshold');
        $sComment = Input::get('comment');
        $fTrainFile = Input::file('train_file');
        $oValidation = Model::validate([
            'id' => $iModelId,
            'name' => $sName,
            'duration' => $iDuration,
            'min_threshold' => $iMinThreshold,
            'comment' => $sComment,
            'train_file' => $fTrainFile
        ]);
        // если валидация провалилась, редиректим обратно с ошибками и заполненными полями
        if ($oValidation->fails())
            return Redirect::route('models.edit', ['iModelId' => $iModelId])->withErrors($oValidation)->withInput();
        $bResult = $this->oRepo->updateModel($iModelId, $sName, $sComment, $fTrainFile, $iDuration, $iMinThreshold);
        if ($bResult)
            return Redirect::route('models.detail', ['iModelId' => $iModelId])->withForm_result('editDone');
        else
            return Redirect::route('models.detail', ['iModelId' => $iModelId])->withForm_result('editFailed');
    }

    /** отдает на скачивание файл шаблона обучающей выборки
     * @return \Symfony\Component\HttpFoundation\BinaryFileResponse */
    public function downloadTemplate() {
        return Response::download(public_path().'/files/template.xlt');
    }

    /** неактивные по порогу отсечения модели
     * @return \Illuminate\View\View */
    public function inactiveModels() {
        $oModels = $this->oRepo->getInactiveModels();
        return View::make('admin.models.inactive', [
            'models' => $oModels
        ]);
    }

    /** отдает на скачивание файл основной обучающей выборки
     * @param $iModelId - id модели
     * @throws \Illuminate\Exceptions\DumpSelectionException */
    public function dump($iModelId) {
        $this->oRepo->dumpSelectionToFile($iModelId);
    }
}

App/controllers/admin/AdminSituationController.php
<?php

/** контроллер проблемных ситуаций
 * Class AdminSituationController */
class AdminSituationController extends \BaseController
{
    /**
     * @var SituationRepository - репозиторий проблемных ситуаций, там хранится бизнес-логика
     */
    protected $oRepo;

    public function __construct(AdminSituationRepository $oRepo)

```

```

    {
        // внедряем зависимость - репозиторий
        $this->oRepo = $oRepo;
    }

    /** показывает список проблемных ситуаций
     * @param int $iParentSituationId - ситуация-родитель, наследников которой надо показать
     * если 0, значит это верхний уровень
     * @return \Illuminate\View\View */
    public function index($iParentSituationId = 0) {
        // собираем список разделов
        $oSituations = $this->oRepo->getSituationsList($iParentSituationId, true);
        // собираем дерево родителей для хлебных крошек
        $aHierarchy = \Elluminate\Engine\E::buildHierarchy($iParentSituationId);
        // отдаем данные в вид и рисуем его
        return View::make('admin.situations.index', [
            'situations' => $oSituations,
            'parent_situation' => $iParentSituationId,
            'hierarchy' => $aHierarchy
        ]);
    }

    /** показывает форму для добавления ситуации
     * @param int $iParentSituationId - id ситуации, внутрь которой будем добавлять
     * если 0, то добавляем на верхний уровень
     * @return \Illuminate\View\View */
    public function create($iParentSituationId = 0) {
        $iParentSituationId = (int)$iParentSituationId;
        // собираем дерево родителей для хлебных крошек
        $aHierarchy = \Elluminate\Engine\E::buildHierarchy($iParentSituationId);
        // отдаем данные в вид и рисуем его
        return View::make('admin.situations.create', [
            'parent_situation_id' => $iParentSituationId,
            'hierarchy' => $aHierarchy
        ]);
    }

    /** сохраняет добавленную проблемную ситуацию
     * @return $this */
    public function store() {
        // получаем данные из post-массива
        $sName = Input::get('name');
        $sOkvedCorrespondence = Input::get('okved_correspondence');
        $iParentId = Input::get('parent_id', 0);
        // проводим валидацию входных данных
        $oValidation = Situation::validate([
            'name' => $sName,
            'okved_correspondence' => $sOkvedCorrespondence,
            'parent_id' => $iParentId
        ]);
        // если валидация провалилась, редиректим обратно с ошибками и заполненными полями
        if ($oValidation->fails())
            return Redirect::route('situations.create', ['iParentSituationId' => $iParentId])->withErrors($oValidation)->withInput();
        $bResult = $this->oRepo->storeSituation($sName, $sOkvedCorrespondence, $iParentId);
        // магия - передает в вид переменную form_result - какой шаблон отрисовать в качестве результата добавления
        if ($bResult)
            return Redirect::route('situations.list', ['iParentSituationId' => $iParentId])->withForm_result('addDone');
        else
            return Redirect::route('situations.list', ['iParentSituationId' => $iParentId])->withForm_result('addFailed');
    }

    /** показывает форму редактирования проблемной ситуации
     * @param $iSituationId - id ситуации, которую будем редактировать */

```

```

/*
 */
public function edit($iSituationId)
{
    $iSituationId = (int)$iSituationId;
    // если нас хотят обмануть или такой ситуации просто
    // нет, отдаём 404 ошибку
    if (!$iSituationId || !Situation::find($iSituationId, ['id'])) throw new \Symfony\Component\HttpKernel\Exception\NotFoundHttpException();
    // собираем дерево родителей для хлебных крошек
    $aHierarchy = \Illuminate\Engine\E::buildHierarchy($iSituationId);
    // находим нужную сущность (физически - строка
    // таблицы)
    $oSituation = Situation::find($iSituationId);
    // рисуем вид
    return View::make('admin.situations.edit', [
        'hierarchy' => $aHierarchy,
        'situation' => $oSituation
    ]);
}

/** сохраняет изменения, внесенные в проблемную
 * ситуацию
 */
public function update()
{
    $iSituationId = (int)Input::get('situation_id');
    // получаем данные из post-массива
    $sName = Input::get('name');
    $sOkvedCorrespondence = Input::get('okved_correspondence');
    // проводим валидацию входных данных
    $oValidation = Situation::validate([
        'name' => $sName,
        'id' => $iSituationId
    ]);
    // получим родителя редактируемого раздела, чтобы знать
    // куда вернуть пользователя при редиректе
    $iParentId = isset(Situation::find($iSituationId)->parent()->get()->first()->id) ? (int)Situation::find($iSituationId)->parent()->get()->first()->id : 0;
    // если валидация провалилась, редиректим обратно с
    // ошибками и заполненными полями
    if ($oValidation->fails())
        return Redirect::route('situations.edit', ['iSituationId' => $iSituationId])->withErrors($oValidation)->withInput();
    $bResult = $this->oRepo->updateSituation($iSituationId, $sName, $sOkvedCorrespondence);
    // магия - передает в вид переменную form_result - какой
    // шаблон отрисовать в качестве результата добавления
    if ($bResult)
        return Redirect::route('situations.list', ['iParentSituationId' => $iParentId])->withForm_result('editDone');
    else
        return Redirect::route('situations.list', ['iParentSituationId' => $iParentId])->withForm_result('editFailed');
}

/** удаляет проблемную ситуацию
 * @param $iSituationId - id ситуации, которую надо удалить
 */
public function destroy($iSituationId)
{
    // если нас хотят обмануть или такой ситуации просто
    // нет, отдаём 404 ошибку
    if (!$iSituationId || !Situation::find($iSituationId)) throw new \Symfony\Component\HttpKernel\Exception\NotFoundHttpException();
    // получим родителя удаляемого раздела, чтобы знать
    // куда вернуть пользователя при редиректе
    $iParentId = isset(Situation::find($iSituationId)->parent()->get()->first()->id) ? (int)Situation::find($iSituationId)->parent()->get()->first()->id : 0;
    $bResult = $this->oRepo->destroySection($iSituationId);
    // магия - передает в вид переменную form_result - какой
    // шаблон отрисовать в качестве результата удаления
    if ($bResult)
        return Redirect::route('situations.list', ['iParentSituationId' => $iParentId])->withForm_result('delDone');
    else
        return Redirect::route('situations.list', ['iParentSituationId' => $iParentId]);
}

```

```

=> $iParentId])->withForm_result('delFailed');
}

App/controllers/client/ClientModelController.php
<?php

/**
 * Created by PhpStorm.
 * User: elluminate
 * Date: 18.02.15
 * Time: 14:23
 */
class ClientModelController extends \BaseController
{
    /**
     * @var ClientModelRepository - репозиторий
     */
    protected $oRepo;

    public function __construct(ClientModelRepository $oRepo)
    {
        // запрещаем пользоваться, пока не даст нам ответ
        $this->beforeFilter(function()
        {
            if (!Entrust::hasRole('administrator'))
            {
                $bExpired =
                    EvaluationRepository::checkExpiredEvaluations();
                if ($bExpired) return Redirect::route('evaluations.list');
            }
        });
        $this->oRepo = $oRepo;
    }

    /**
     * показывает список решаемых задач
     * @param $iSituationId - id родительской проблемной
     * ситуации
     */
    public function index($iSituationId)
    {
        $oModels = $this->oRepo->getModelsList($iSituationId, true);
        $aHierarchy = \Illuminate\Engine\E::buildHierarchy($iSituationId);
        return View::make('client.models.index', [
            'models' => $oModels,
            'hierarchy' => $aHierarchy
        ]);
    }

    /**
     * показывает форму для использования модели
     * @param $iModelId - id модели
     */
    public function showModelForm($iModelId)
    {
        $oModel = $this->oRepo->getModel($iModelId, ['id', 'name', 'comment', 'situation_id', 'min_threshold', 'threshold', 'durations_id']);
        if ($oModel->threshold < $oModel->min_threshold) throw new \Symfony\Component\HttpKernel\Exception\NotFoundHttpException();
        $iSituationId = $oModel->situation_id;
        $iDurationsId = (int)$oModel->durations_id;
        $oDuration = Duration::find($iDurationsId, ['id', 'name']);
        $sDurationName = $oDuration->name;
        unset($oDuration);
        $oForm = $this->oRepo->getApplyingForm($iModelId);
        $aHierarchy = \Illuminate\Engine\E::buildHierarchy($iSituationId);
        return View::make('client.models.detail', [
            'model' => $oModel,
            'form' => $oForm,
            'duration' => $sDurationName,
            'hierarchy' => $aHierarchy
        ]);
    }

    /**
     * запускает работу модели с введенными пользователем
     * данными
     */
    public function compute()
    {
        $aInput = Input::all();
        $oValidation = $this->oRepo->validateUserInput($aInput);
        if ($oValidation->fails())
            return Redirect::route('tasks.detail', ['iModelId' => $aInput['model_id']])->withErrors($oValidation)->withInput();
    }
}

```

```

    $fResult = $this->oRepo->computeResult($alnput);
    $oModel = $this->oRepo->getModel($alnput['model_id'], ['id',
    'reg_name', 'comment', 'situation_id', 'durations_id']);
    $sRegName = isset($oModel->reg_name) ? $oModel-
>reg_name : '';
    $sComment = isset($oModel->comment) ? $oModel-
>comment : '';
    $aHierarchy = \Elluminate\Engine\E::buildHierarchy($oModel-
>situation_id);
    return View::make('client.models.result', [
        'result' => $fResult,
        'reg_name' => $sRegName,
        'comment' => $sComment,
        'hierarchy' => $aHierarchy,
        'model_id' => $oModel->id
    ]);
}
}

App/controllers/clientt/ClientSituationController
<?php

```

```

/** контроллер проблемных ситуаций для
зарегистрированного пользователя
*/
class ClientSituationController extends \BaseController
{

    /**
     * @var ClientSituationRepository - репозиторий
     */
    protected $oRepo;

    public function __construct(ClientSituationRepository $oRepo)
    {
        $this->oRepo = $oRepo;
    }

    /** показывает список проблемных ситуаций
     * @param int $iParentSituationId - id родительской
проблемной
     * @return \Illuminate\View\View
     */
    public function index($iParentSituationId = 0)
    {
        // собираем список разделов с моделями
        $oSituations = $this->oRepo-
>getSituationsList($iParentSituationId, true, true, true);
        // собираем дерево родителей для хлебных крошек
        $aHierarchy = \Elluminate\Engine\E::buildHierarchy($iParentSituationId);
        // отдаем данные в вид и рисуем его
        return View::make('client.situations.index', [
            'situations' => $oSituations,
            'hierarchy' => $aHierarchy
        ]);
    }
}

App/controllers/client/EvaluationController.php
<?php
/** Created by PhpStorm.
 * User: elluminate
 * Date: 23.02.15
 * Time: 16:18
 */
class EvaluationController extends \BaseController {

    /**
     * @var EvaluationRepository - репозиторий проведенных
вычислений
     */
    protected $oRepo;

    public function __construct(EvaluationRepository $oRepo) {
        $this->oRepo = $oRepo;
    }

    /** показывает список вычислений, которые нуждаются в
подтверждении
     * @return \Illuminate\View\View
     */
    public function index() {
        // проверим есть ли у пользователя роль зарег. юзера
        $iUserId = \Auth::user()->id;
        if(!$iUserId || !Entrust::hasRole('user')) throw new

```

```

\Symfony\Component\HttpKernel\Exception\NotFoundHttpException
n;
    // возьмем все вычисления, которые соответствуют этому
пользователю и не умеют сведений о реальном результате
    $oEvaluations = Evaluation::with('Model')->where('user_id',
    '=', $iUserId)->whereNull('real_result')->get();
    return View::make('client.evaluations.index', [
        'evaluations' => $oEvaluations
    ]);
}

/** показывает форму обратной связи
 * @param $iEvaluationId - id вычисления, для которого
выводится форма
 * @return \Illuminate\View\View
 */
public function show($iEvaluationId) {
    $oForm = $this->oRepo-
>getConfirmationForm($iEvaluationId);
    $iModelId = Evaluation::find($iEvaluationId, ['id', 'model_id'])-
>model_id;
    $iEstimatedResult = $this->oRepo-
>getEstimatedResult($iEvaluationId);
    return View::make('client.evaluations.detail', [
        'form' => $oForm,
        'iEvaluationId' => $iEvaluationId,
        'estimated_result' => $iEstimatedResult,
        'model_id' => $iModelId
    ]);
}

/** осуществляет обратную связь
 */
public function confirm() {
    $iRealResult = (int)Input::get('real_result');
    $iEvaluationId = (int)Input::get('evaluation_id');
    $oValidation = Evaluation::validate([
        'evaluation_id' => $iEvaluationId,
        'real_result' => $iRealResult
    ]);
    // если валидация провалилась, редиректим обратно с
ошибками и заполненными полями
    if ($oValidation->fails())
        return Redirect::route('evaluations.detail', ['iEvaluationId' =>
        $iEvaluationId])->withErrors($oValidation)->withInput();
    $this->oRepo->confirm($iEvaluationId, $iRealResult);
    return View::make('client.evaluations.success');
}
}

App/controllers/client/SearchController.php
<?php
/**
 * Created by PhpStorm.
 * User: elluminate
 * Date: 18.02.15
 * Time: 14:30
 */
class SearchController extends BaseController {

    protected $oSearch;

    public function __construct(\Elluminate\Engine\Search $oSearch)
    {
        $this->oSearch = $oSearch;
    }

    public function index()
    {
        // получим поисковую фразу
        $sSearchText = Input::get('search_text', '');
        $this->oSearch->setSearchText($sSearchText);
        $this->oSearch->startSearch();
        // возьмем все нужные результаты поиска
        $aOkvedCodeResult = $this->oSearch-
>getOkvedCodeResult();
        $aSituationNameResult = $this->oSearch-
>getSituationNameResult();
        $aModelNameResult = $this->oSearch-
>getModelNameResult();
        $bCodeOverlimit = $this->oSearch-
>getOkvedCodeOverlimit();
        $bSitNameOverlimit = $this->oSearch-
>getSituationNameOverlimit();
        $bModelNameOverlimit = $this->oSearch-
>getModelNameOverlimit();
        // и отдадим в вид
        return View::make('client.search.index', [
            'okved_code' => $aOkvedCodeResult,

```

```

        'code_overlimit'      =>      $bCodeOverlimit,
        'situation_name'     =>      $aSituationNameResult,
        'situation_overlimit' =>      $bSitNameOverlimit,
        'model_name'          =>      $a modelNameResult,
        'model_overlimit'    =>      $bModelNameOverlimit
    ]);
}

App/database/migrations/2015_01_19_115031_confide_setup_user_s_table.php
<?php

use Illuminate\Database\Migrations\Migration;
class ConfideSetupUsersTable extends Migration
{
    /**
     * Run the migrations.
     */
    public function up()
    {
        // Creates the users table
        Schema::create('users', function ($table) {
            $table->increments('id');
            $table->string('username')->unique();
            $table->string('email')->unique();
            $table->string('password');
            $table->string('confirmation_code');
            $table->string('remember_token')->nullable();
            $table->boolean('confirmed')->default(false);
            $table->timestamps();
        });

        // Creates password reminders table
        Schema::create('password_reminders', function ($table) {
            $table->string('email');
            $table->string('token');
            $table->timestamp('created_at');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down()
    {
        Schema::drop('password_reminders');
        Schema::drop('users');
    }
}

App/database/migrations/2015_01_19_115612 Entrust_setup_table_s.php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;

class EntrustSetupTables extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        // Creates the roles table
        Schema::create('roles', function ($table) {
            $table->increments('id')->unsigned();
            $table->string('name')->unique();
            $table->timestamps();
        });

        // Creates the assigned_roles (Many-to-Many relation) table
        Schema::create('assigned_roles', function ($table) {
            $table->increments('id')->unsigned();
            $table->integer('user_id')->unsigned();
            $table->integer('role_id')->unsigned();
            $table->foreign('user_id')->references('id')->on('users')
                ->onDelete('cascade')->onUpdate('cascade');
            $table->foreign('role_id')->references('id')->on('roles');
        });

        // Creates the permissions table
        Schema::create('permissions', function ($table) {
            $table->increments('id')->unsigned();
            $table->string('name')->unique();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down()
    {
        Schema::drop('assigned_roles');
        Schema::drop('permissions');
        Schema::drop('roles');
        Schema::drop('users');
    }
}

App/database/migrations/2015_01_30_114651_create_durations_table.php
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateDurationsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        // создаем таблицу с возможными вариантами корректности решения
        Schema::create('durations', function($table)
        {
            $table->increments('id')->unsigned();
            $table->string('name', 50);
            // будем хранить в часах, а потом добавлять с помощью Carbon
            $table->smallInteger('duration')->unsigned();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down()
    {
        // удаляем таблицу при откате миграции
        Schema::drop('durations');
    }
}

App/database/migrations/2015_01_30_130502_create_models_table.php

```

```

<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateModelsTable extends Migration {
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('models', function($table) {
            $table->increments('id');
            $table->integer('situation_id');
            $table->string('name');
            $table->text('comment')->nullable();
            $table->text('cov_names');
            $table->text('cov_comments');
            $table->string('reg_name');
            $table->string('reg_comment');
            $table->text('coefficients');
            $table->smallInteger('durations_id');
            $table->smallInteger('min_threshold')->unsigned();
            $table->text('core_selection');
            $table->text('oversampling')-
        });
        $table->integer('threshold');
        $table->text('std_coeff');
        $table->text('elastic_coeff');
        $table->float('curve_area');
        $table->float('sill')->unsigned();
        $table->foreign('situation_id')-
        references('id')->on('situations')
            ->onUpdate('cascade')-
        onDelete('cascade');
        $table->foreign('durations_id')-
        references('id')->on('durations')
            ->onUpdate('cascade')-
        onDelete('cascade');
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::drop('models');
    }
}

App/database/migrations/2015_01_30_154606_create_situations_table.php
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateSituationsTable extends Migration {
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('situations', function($table) {
            $table->increments('id');
            $table->string('name', 512);
            $table->string('okved_correspondence', 16)->default('');
            $table->unsigned()->nullable();
            $table->foreign('parent_id')-
            references('id')->on('situations')->onDelete('cascade')-
            onUpdate('cascade');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::table('situations', function (Blueprint $table) {
            $table->dropForeign('situations_parent_id_foreign');
        });
        Schema::drop('situations');
    }
}

App/database/migrations/2015_02_15_162118_drop_situations_for_key.php
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class DropSituationsForkey extends Migration {
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('situations', function (Blueprint $table) {
            $table->dropForeign('situations_parent_id_foreign');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::table('situations', function (Blueprint $table) {
            $table->foreign('parent_id')-
            references('id')->on('situations')->onDelete('cascade')-
            onUpdate('cascade');
        });
    }
}

App/database/migrations/2015_02_19_163722_create_evaluations_table.php
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateEvaluationsTable extends Migration {
}

```

```


    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('evaluations', function($table)
        {
            $table->increments('id')-
            >unsigned();
            $table-
            >smallInteger('estimated_result')->unsigned();
            $table->smallInteger('real_result')-
            >unsigned()->nullable();
            $table->text('covariates');
            $table-
            >dateTime('expired_moment');
            $table->integer('user_id')-
            >unsigned();
            $table->integer('model_id')-
            >unsigned();
            $table->foreign('user_id')-
            >references('id')->on('users')
                ->onUpdate('cascade')-
            >onDelete('cascade');
            $table->foreign('model_id')-
            >references('id')->on('models')
                ->onUpdate('cascade')-
            >onDelete('cascade');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::drop('evaluations');
    }
}


```

App/database/migrations/2015_02_20_151016_create_failed_jobs_table.php
<?php

```


use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;
class CreateFailedJobsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('failed_jobs', function(Blueprint $table)
        {
            $table->increments('id');
            $table->text('connection');
            $table->text('queue');
            $table->text('payload');
            $table->timestamp('failed_at');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::drop('failed_jobs');
    }
}


```

App/database/seeds/AdminSeeder.php

```


<?php
/*
 * Created by User: Date: Time:
 */
class AdminSeeder extends Seeder
{
    public function run()
    {
        $user = new User();
        $user->username = 'admin';
        $user->email = 'elluminatte@icloud.com';
        $user->password = 'admin';
        $user->password_confirmation = 'admin';
        $user->confirmation_code = md5(uniqid(mt_rand(), true));
        if(!$user->save())
            Log::info('Unable to create user '.$user->username, (array)$user->errors());
        else
            Log::info('Created user "'.$user->username.'" <'.$user->email.'>');
    }

    $admin = new Role();
    $admin->name = 'administrator';
    $admin->save();
    $userRole = Role::where('name', '=', 'user')->first();
    $user = User::where('username','=' , 'admin')->first();
    $user->attachRole($admin);
    $user->attachRole($userRole);
}

App/database/seeds/DurationSeeder.php
<?php
/*
 * Created by User: Date: Time:
 */
class DurationSeeder extends Seeder
{
    public function run()
    {
        DB::table('durations')->delete();

        $duration = new Duration();
        $duration->name = 'час';
        $duration->duration = 1;
        $duration->save();

        $duration = new Duration();
        $duration->name = 'день';
        $duration->duration = 24;
        $duration->save();

        $duration = new Duration();
        $duration->name = 'неделя';
        $duration->duration = 24*7;
        $duration->save();

        $duration = new Duration();
        $duration->name = 'месяц';
        $duration->duration = 24*30;
        $duration->save();

        $duration = new Duration();
        $duration->name = 'квартал';
        $duration->duration = 24*30*4;
        $duration->save();

        $duration = new Duration();
        $duration->name = 'год';
        $duration->duration = 365*24;
        $duration->save();
    }
}


```

app/models/repos/admin/AdminModelRepository.php

```


<?php
/*
 * Created by User: Date: Time:
 */


```

```

/*
* Date: 25.02.15
* Time: 12:13
*/
class AdminModelRepository extends ModelRepository {
    /**
     * получает модель и информацию о ней
     * @param $iModelId - id модели
     * @return \Illuminate\Database\Eloquent\Model|\Illuminate\Support\Collection
    */
    public function getModelDetail($iModelId) {
        $iModelId = (int)$iModelId;
        if(!$iModelId || !Model::find($iModelId, ['id'])) throw new \Symfony\Component\HttpKernel\Exception\NotFoundHttpException();
        // возьмем все поля, которые интересуют администратора
        $aFields = ['id', 'name', 'comment', 'cov_names', 'cov_comments', 'reg_name', 'reg_comment', 'coefficients', 'min_threshold', 'threshold', 'std_coeff', 'elastic_coeff', 'curve_area', 'durations_id', 'situation_id'];
        // это типа join
        $oModel = Model::with('duration')->find($iModelId, $aFields);
        // преобразуем в массивы
        $oModel->cov_names = json_decode($oModel->cov_names);
        $oModel->cov_comments = json_decode($oModel->cov_comments);
        $oModel->coefficients = json_decode($oModel->coefficients);
        $oModel->std_coeff = json_decode($oModel->std_coeff);
        $oModel->elastic_coeff = json_decode($oModel->elastic_coeff);
        $oModel->duration = Duration::find($oModel->durations_id, ['id', 'name']);
        return $oModel;
    }

    /**
     * получает неактивные по порогу отсечения модели
     * @return \Illuminate\Database\Eloquent\Collection|static[]
    */
    public function getInactiveModels() {
        return Model::where('threshold', '<', DB::raw('min_threshold'))->get(['id', 'name', 'situation_id']);
    }

    /**
     * удаляет модель
     * @param $iModelId - id модели
     * @return bool|null - результат удаления
     * @throws Exception
    */
    public function destroyModel($iModelId) {
        $iModelId = (int)$iModelId;
        // проверяем не хотят ли нас обмануть - существует ли такая модель
        if (!$iModelId || !Model::find($iModelId, ['id'])) throw new \Symfony\Component\HttpKernel\Exception\NotFoundHttpException();
        // удаляем сущность
        return Model::find($iModelId)->delete();
    }

    /**
     * сохраняет новую модель
     * @param $iSituationId - id родительской ситуации
     * @param $sName - название
     * @param $iDuration - время корректности решения
     * @param $iMinThreshold - минимальный порог отсечения
     * @param $sComment - комментарий
     * @param $fTrainFile - файл с обучающей выборкой
     * @return bool
     * @throws \Illuminate\Exceptions\TrainSetFileNotFoundException
    */
    public function storeModel($iSituationId, $sName, $iDuration, $iMinThreshold, $sComment, $fTrainFile) {
        $iSituationId = (int)$iSituationId;
        if(!$iSituationId || !Situation::find($iSituationId, ['id'])) throw new \Symfony\Component\HttpKernel\Exception\NotFoundHttpException();
        $result = $this->constructModel(null, $iSituationId, $sName, $sComment, $fTrainFile, $iDuration, $iMinThreshold);
        return $result;
    }

    /**
     * объединяет общий функционал обновления и добавления моделей
     * @param null $iModelId - id модели, если null, то сущность будет новой
     * @param $sName - название
     * @param $sComment - комментарий
    */
    private function constructModel($iModelId = null, $iSituationId = null, $sName, $fTrainFile, $iDuration, $iMinThreshold) {
        /**
         * @param $fTrainFile - файл обучающей выборки
         * @param $iDuration - время корректности решения
         * @param $iMinThreshold - минимальный порог отсечения
         * @return bool
         * @throws \Illuminate\Exceptions\TrainSetFileNotFoundException
        */
        if(is_null($iModelId)) $oModel = new Model();
        else $oModel = Model::find($iModelId);
        $oModel->name = $sName;
        $oModel->durations_id = $iDuration;
        $oModel->min_threshold = $iMinThreshold;
        $oModel->comment = $sComment;
        if(!is_null($iSituationId))
            $oModel->situation_id = $iSituationId;
        if(is_null($iModelId) || (!is_null($iModelId) && !is_null($fTrainFile))) {
            // получим из файла все, что нам нужно
            $aFileContent = $this->extractDataFromExcel($fTrainFile);
            // преобразуем содержимое в нужную форму
            $this->prepareFileContent($aFileContent);
            // заберем оттуда саму выборку
            $aTrainingSet = array_slice($aFileContent, 2);
            // возьмем название регрессии
            $aRegName = $aFileContent[0][0];
            // единицы измерения регрессии
            $aRegComment = $aFileContent[1][0];
            // названия regressorов
            $aCovNames = array_slice($aFileContent[0], 1);
            // единицы измерения regressorов
            $aCovComments = array_slice($aFileContent[1], 1);
            unset($aFileContent);
            // зададим выборку для модели
            $this->oModel->setTrainingSet($aTrainingSet);
            // обучим модель
            $this->oModel->trainModel();
            // зададим модель для анализа качества
            $this->oQuality->setModel($this->oModel);
            // проведем анализ качества
            $this->oQuality->getQualityAnalysis();
            $oModel->cov_names = json_encode($aCovNames, JSON_UNESCAPED_UNICODE);
            $oModel->cov_comments = json_encode($aCovComments, JSON_UNESCAPED_UNICODE);
            $oModel->reg_name = $aRegName;
            $oModel->reg_comment = $aRegComment;
            $oModel->coefficients = json_encode($this->oModel->getCoefficients(), JSON_NUMERIC_CHECK);
            $oModel->durations_id = $iDuration;
            $oModel->min_threshold = $iMinThreshold;
            $oModel->core_selection = json_encode($aTrainingSet, JSON_NUMERIC_CHECK);
            $oModel->threshold = $this->oQuality->getThreshold();
            $oModel->std_coeff = json_encode($this->oQuality->getStdCoeff(), JSON_NUMERIC_CHECK);
            $oModel->elastic_coeff = json_encode($this->oQuality->getElasticCoeff(), JSON_NUMERIC_CHECK);
            $oModel->curve_area = $this->oQuality->getCurveArea();
            $oModel->sill = $this->oQuality->getSill();
        }
        return $oModel->save();
    }

    /**
     * обновляет модель
     * @param $iModelId - id модели
     * @param $sName - название
     * @param $sComment - комментарий
     * @param $fTrainFile - файл обучающей выборки
     * @param $iDuration - время корректности решения
     * @param $iMinThreshold - минимальный порог отсечения
     * @return bool
     */
    public function updateModel($iModelId, $sName, $sComment, $fTrainFile, $iDuration, $iMinThreshold) {
        $iModelId = (int)$iModelId;
        if(!$iModelId || !Model::find($iModelId)) throw new \Symfony\Component\HttpKernel\Exception\NotFoundHttpException();
        return $this->constructModel($iModelId, null, $sName, $sComment, $fTrainFile, $iDuration, $iMinThreshold);
    }

    /**
     * приводит содержимое файла обучающей выборки в нужный нам вид
     * @param $aFileContent - содержимое файла обучающей выборки
    */
}

```

```

* @throws \Elluminate\Exceptions\TrainSetFileException
*/
private function prepareFileContent(&$aDialogContent) {
    // проверим, что в массиве что-то есть
    if(!is_array($aDialogContent) || !count($aDialogContent)) throw new
\Elluminate\Exceptions\TrainSetFileException("Ошибка при
считывании обучающей выборки");
    // найдем последнюю колонку
    $iLastCol = $this->findLastCol($aDialogContent);
    // уберем лишние колонки и вычеркнем неполные строки
    $this->removeExcessColsAndIncomplRows($aDialogContent,
    $iLastCol);
}

/** удаляет лишние столбцы и строчки из файла обучающей
выборки
 * @param $aDialogContent
 * @param $iOffset
 */
private function removeExcessColsAndIncomplRows(&$aDialogContent, $iOffset) {
    // пройдем по строкам
    foreach($aDialogContent as $rowKey => &$aRow) {
        // отрежем лишние столбцы
        array_splice($aRow, $iOffset);
        // пройдем по столбцам
        foreach($aRow as $key => $value) {
            // первые 2 трогать не будем, там имена и
комментарии, их мы уже проверили
            if($rowKey < 2) continue;
            // если в строке есть хотя бы 1 пустой столбец
            if(is_null($value))
                // то вычеркиваем всю строку
                unset($aDialogContent[$rowKey]);
        }
    }
    // заново проиндексируем массив, чтобы убрать пробелы
от      вычикивания      строк
$aDialogContent = array_values($aDialogContent);
}

/** ищет номер последней значащей колонки в файле
 * @param $aDialogContent - содержимое файла
 * @return mixed - номер последней значащей колонки
 */
private function findLastCol($aDialogContent) {
    $aEmpty = [];
    // пройдем по первым двум строкам - если нет имени
рессорса или названия, то мы его брать не будем - это
бесмысленно
    for($iRow = 0; $iRow <= min(count($aDialogContent), 1);
++$iRow)
        foreach($aDialogContent[$iRow] as $key => $value) {
            // если значения нет
            if(is_null($value))
                // запишем номер столбца, где пустая строкка
                array_push($aEmpty, $key);
            // дальше смотреть не надо, нам нужен минимум,
пойдем      на      следующую      строку
            break;
        }
    }
    // найдем самый левый столбец, где еще есть нужные нам
данные
    return min($aEmpty);
}

/** получает данные из файла Excel
 * @param $sFileName - путь к файлу
 * @param int $iOffset - сколько строчек сверху пропускать
 * @param int $iRowLimit - сколько строк прочесть
 * @param int $iColLimit - сколько столбцов прочесть
 * @param int $iSheetNumber - номер листа
 * @return array - данные из файла
 * @throws PHPExcel_Exception
 * @throws \Elluminate\Exceptions\TrainSetFileException
 */
protected function extractDataFromExcel($sFileName, $iOffset
= 12, $iRowLimit = 0, $iColLimit = 0, $iSheetNumber = 0) {
    $iOffset = (int)$iOffset;
    $iRowLimit = (int)$iRowLimit;
    $iColLimit = (int)$iColLimit;
    $iSheetNumber = (int)$iSheetNumber;
    $aTrainingSet = array();
    try {
        $oInputFileType
        = PHPExcel_IOFactory::identify($sFileName);

```

```

        $oReader
        = PHPExcel_IOFactory::createReader($oInputFileType);
        // видит непустую ячейку только там, где есть реальное
значение, а не стиль и т.д.
        $oReader->setReadDataOnly(true);
        $oPHPExcel = $oReader->load($sFileName);
    }
    catch (\Exception $e) {
        throw new \Elluminate\Exceptions\TrainSetFileException("Ошибка при
попытке считать данные из файла обучающей выборки");
    }
    $oSheet = $oPHPExcel->getSheet($iSheetNumber);
    // чтобы не считать ячейки, где есть, например, стиль,
должно помочь вместе с setReadDataOnly
    // если заданы ограничения, то читаем до них, иначе -
читаем всё, что есть
    $iHighestRow = $iRowLimit ? $iRowLimit : $oSheet-
>getHighestDataRow();
    $iHighestColumn = $iColLimit ? $iColLimit : $oSheet-
>getHighestDataColumn();
    // колонок не должно быть больше, чем максимальное
кол-во регрессоров + значение функции
    $iHighestColumn = min($iHighestColumn,
\Elluminate\Engine\E::findLetterByPos(\Elluminate\Math\LogisticRe
gression::MAX_COVARIATES_NUM));
    for ($iRow = $iOffset; $iRow <= $iHighestRow; $iRow++) {
        $aRowData = $oSheet->rangeToArray('A' . $iRow . ':' .
$iRow);
        $iHighestColumn
        =
        $aRowData[0];
    }
    return $aTrainingSet;
}

/** пишет основную обучающую выборку в файл
 * @param $iModelId - id модели
 * @return string - путь к файлу
 * @throws \Elluminate\Exceptions\DumpSelectionException
 */
public function dumpSelectionToFile($iModelId) {
    // получим всё, что нужно для модели
    $oModel = $this->getModel($iModelId, ['id', 'name',
'cov_names', 'cov_comments', 'reg_name', 'core_selection']);
    // преобразуем имя в валидное для имени файла
    $sName = \Elluminate\Engine\E::transliterate($oModel-
>name);
    try {
        // загрузим шаблон
        $objPHPExcel = PHPExcel_IOFactory::load(public_path() .
'/files/dump.xls');
        // будем писать в первый лист
        $objPHPExcel->setActiveSheetIndex(0);
        // запишем имя функции
        $objPHPExcel->getActiveSheet()->SetCellValue('A12',
$oModel->reg_name);
        // сформируем массив X1..Xn
        $aCovNums = [];
        $iCovCount = count(json_decode($oModel->cov_names));
        for($i = 1; $i <= $iCovCount; ++$i) {
            array_push($aCovNums, 'X' . $i);
        }
        // запишем строку X1..Xn
        $objPHPExcel->getActiveSheet()->fromArray($aCovNums,
'B11');
        // запишем строку имен регрессоров
        $objPHPExcel->getActiveSheet()-
>fromArray(json_decode($oModel->cov_comments), ' ', 'B12');
        // запишем строку единиц измерения регрессоров
        $objPHPExcel->getActiveSheet()-
>fromArray(json_decode($oModel->cov_names), ' ', 'B13');
        // начнём писать основную обучающую выборку
        $objPHPExcel->getActiveSheet()-
>fromArray(json_decode($oModel->core_selection), null, 'A14',
true);
        $objWriter = new PHPExcel_Writer_Excel5($objPHPExcel);
        // создадим новый файл по имени функции + текущее
дата/время
        $dumpName = $sName . '-' . date("Y-m-d-H-i-s") . '.xls';
        // отдадим всё, что записали в стандартный поток
вывода
        header('Content-Type: application/vnd.ms-excel');
        header('Content-Disposition: attachment;filename="'.$dumpName.'"]');
        header('Cache-Control: max-age=0');
        $objWriter->save('php://output');
        return true;
    }
    catch(\Exception $e) {

```

```

        throw new
        \Illuminate\Exceptions\DumpSelectionException("Ошибка при
попытке выгрузить основную обучающую выборку");
    }

}

App/models/repos/admin/AdminSituationRepository.php
<?php
/**
 * Created by PhpStorm.
 * User: elluminate
 * Date: 25.02.15
 * Time: 12:23
 */
class AdminSituationRepository extends SituationRepository {
    /**
     * добавляет новую ситуацию
     * @param $sName - название
     * @param $sOkvedCorrespondence - соответствие ОКВЭД
     * @param $iParentId - id ситуации-родителя
     * @return bool
     */
    public function storeSituation($sName,
$sOkvedCorrespondence,
$iParentId)
    {
        // кое-как обработали данные, внутри PDO, с
        // экранированием заморачиваться не надо
        $sName = (string)$sName;
        $sOkvedCorrespondence = (string)$sOkvedCorrespondence;
        $iParentId = (int)$iParentId;
        // проверяем не хотят ли нас обмануть - существует ли
        // такой раздел
        if ($iParentId !== 0 && !Situation::find($iParentId, ['id'])) throw
new
\Symfony\Component\HttpKernel\Exception\NotFoundHttpException;
        // создаем сущность
        $oSituation = new Situation();
        // задаем ей атрибуты
        $oSituation->name = $sName;
        $oSituation->okved_correspondence =
        $sOkvedCorrespondence;
        // если родителя нет, то null подставится в СУБД
        if ($iParentId !== 0)
            $oSituation->parent_id = $iParentId;
        // сохраним сущность
        return $oSituation->save();
    }

    /**
     * удаляет проблемную ситуацию
     * @param $iSituationId - id ситуации, которую будем удалять
     * @return bool|null
     * @throws Exception
     */
    public function destroySection($iSituationId)
    {
        $iSituationId = (int)$iSituationId;
        // проверяем не хотят ли нас обмануть - существует ли
        // такой раздел
        if (!$iSituationId || !Situation::find($iSituationId, ['id'])) throw
new
\Symfony\Component\HttpKernel\Exception\NotFoundHttpException;
        // удаляем сущность, о вложенных позаботиться СУБД с
        // помощью внешних ключей
        return Situation::find($iSituationId)->delete();
    }

    /**
     * обновляет данные о проблемной ситуации
     * @param $iSituationId - id ситуации, которую обновляем
     * @param $sName - название
     * @param $sOkvedCorrespondence - соответствие ОКВЭД
     * @return bool
     */
    public function updateSituation($iSituationId, $sName,
$sOkvedCorrespondence)
    {
        $iSituationId = (int)$iSituationId;
        // проверяем не хотят ли нас обмануть - существует ли
        // такой раздел
        if (!$iSituationId || !Situation::find($iSituationId, ['id'])) throw
new
\Symfony\Component\HttpKernel\Exception\NotFoundHttpException;
        // кое-как обработали данные, внутри PDO, с
        // экранированием заморачиваться не надо
    }
}

```

```

        $sName = (string)$sName;
        $sOkvedCorrespondence = (string)$sOkvedCorrespondence;
        // нашли сущность
        $oSituation = Situation::find($iSituationId);
        // записали атрибуты
        $oSituation->name = $sName;
        $oSituation->okved_correspondence =
        $sOkvedCorrespondence;
        // сохранили
        return $oSituation->save();
    }

}

App/models/repos/client/ClientModelRepository.php
<?php
/**
 * Created by PhpStorm.
 * User: elluminate
 * Date: 25.02.15
 * Time: 12:13
 */
class ClientModelRepository extends ModelRepository {
    /**
     * собирает форму для ввода параметров
     * @param $iModelId - id модели
     * @param array $aValues - значения, которыми нужно
     * заполнить поля
     * @return array - массив с полями формы
     */
    public function getApplyingForm($iModelId, $aValues = [])
    {
        $iModelId = (int)$iModelId;
        if (!$iModelId || !Model::find($iModelId, ['id'])) throw
new
\Symfony\Component\HttpKernel\Exception\NotFoundHttpException;
        // какие поля надо взять
        $aFields = ['id', 'cov_names', 'cov_comments'];
        $oModel = $this->getModel($iModelId, $aFields);
        // восстанавливаем массив из БД
        $aNames = json_decode($oModel->cov_names);
        $aComments = json_decode($oModel->cov_comments);
        $aForm = [];
        // надо ли заполнять поля значениями
        $bNeedValues = false;
        if(is_array($aValues))
            $bNeedValues = count($aValues) == true;
        foreach($aNames as $key => $value) {
            $sName = $value;
            $sComment = isset($aComments[$key]) ? $aComments[$key] : '';
            if($bNeedValues)
                $sValue = isset($aValues[$key]) ? $aValues[$key] : '';
            // вставляем очередное поле в массив формы
            if(isset($sValue))
                array_push($aForm, ['tech_name' => $sName,
                    'comment' => $sComment, 'value' => $sValue]);
            else
                array_push($aForm, ['tech_name' => $sName,
                    'comment' => $sComment]);
        }
        unset($aNames);
        unset($aComments);
        return $aForm;
    }

    /**
     * валидирует пользовательский ввод в форму
     * @param $aInput - массив входных данных
     * @return \Illuminate\Validation\Validator - экземпляр
     * валидатора
     */
    public function validateUserInput($aInput)
    {
        // проверим есть ли вообще такая модель
        $iModelId = (int)$aInput['model_id'];
        if (!$iModelId || !Model::find($iModelId, ['id'])) throw
new
\Symfony\Component\HttpKernel\Exception\NotFoundHttpException;
        // соберем форму
        $oForm = $this->getApplyingForm($iModelId);
        // получим правила валидации и названия полей для
        // вывода
        $aRulesAndNames = $this->getUserValidRulesAndNames($oForm);
        $aValidRules = $aRulesAndNames['rules'];
        $aFieldNames = $aRulesAndNames['names'];
    }
}

```

```

        return \Validator::make($alnput, $aValidRules, array(),
$aFieldNames);
    }

/** формирует правила валидации и названия полей для вывода
 * @param $oForm - массив формы
 * @return array - правила и названия
 */
private function getUserValidRulesAndNames($oForm) {
    $aValidRules = [];
    $aNames = [];
    foreach($oForm as $aField) {
        // все поля должны быть числовыми и обязательными для заполнения
        $aValidRules[$aField['tech_name']] = 'Required|Numeric';
        $aNames[$aField['tech_name']] = $aField['name'];
    }
    return ['names' => $aNames, 'rules' => $aValidRules];
}

/** считает значение логистической регрессии при введенных пользователем параметрах
 * @param $alnput - входные данные
 * @return float - значение логистической регрессии
 * @throws \Illuminate\Exceptions\EvaluationException
 * @throws \Illuminate\Exceptions\MathException
 */
public function computeResult($alnput) {
    $iModelId = $alnput['model_id'];
    $iModelId = (int)$iModelId;
    // проверим есть ли нужная нам модель
    if(!$iModelId || !Model::find($iModelId, ['id'])) throw new \Symfony\Component\HttpKernel\Exception\NotFoundHttpException;
    // уберем поля, которые не являются значениями регрессоров
    unset($alnput['model_id']);
    unset($alnput['_token']);
    $aCovValues = [];
    foreach($alnput as $value) {
        array_push($aCovValues, $value);
    }
    $aFields = ['id', 'coefficients'];
    $oModel = $this->getModel($iModelId, $aFields);
    // восстановим массив коэффициентов из БД
    $aCoefficients = json_decode($oModel->coefficients);
    // посчитаем логит. регрессию
    $fResult = round(\Illuminate\Math\MathCore::logisticRegression($aCovValues, $aCoefficients), 2);
    // получим id пользователя, который всё это считает
    $oEvaluationRepo = new EvaluationRepository($this);
    // добавим факт использования модели в БД
    $oEvaluationRepo->addEvaluation($iModelId, $aCovValues, $fResult);
    return $fResult;
}

/** переобучает модель с учетом дополнительной выборки
 * @param $iModelId - id модели, которую надо переобучить
 */
public function retrainModel($iModelId) {
    $iModelId = (int)$iModelId;
    // проверим есть ли такая модель вообще
    if (!$iModelId || !Model::find($iModelId, ['id'])) throw new \Symfony\Component\HttpKernel\Exception\NotFoundHttpException;
    $oModel = Model::find($iModelId);
    // объединим основную и дополнительную обучающую выборку
    $aCoreSelection = json_decode($oModel->core_selection);
    $aOversampling = json_decode($oModel->oversampling);
    if(!is_array($aCoreSelection) || !is_array($aOversampling) || (count($aCoreSelection[0]) != count($aOversampling[0]))) throw new \Illuminate\Exceptions\EvaluationException("Несоответствие размерностей основной и обучающей выборки");
    $aTrainingSet = array_merge($aCoreSelection, $aOversampling);
    // установим ее как единую выборку для обучения
    $this->oModel->setTrainingSet($aTrainingSet);
    // обучим модель
    $this->oModel->trainModel();
    // установим модель для анализа кач-ва
    $this->oQuality->setModel($this->oModel);
    // проведем анализ качества
    $this->oQuality->getQualityAnalysis();
    // обновим для сущности БД те показатели, которые могли

```

```

        $oModel->coefficients = json_encode($this->oModel->getCoefficients());
        $oModel->threshold = $this->oQuality->getThreshold();
        $oModel->std_coeff = json_encode($this->oQuality->getStdCoeff());
        $oModel->elastic_coeff = json_encode($this->oQuality->getElasticCoeff());
        $oModel->curve_area = $this->oQuality->getCurveArea();
        $oModel->sill = $this->oQuality->getSill();
        // если порог стал меньше минимального значения, то отправим администрации письмо
        if($oModel->threshold < $oModel->min_threshold)
            Mail::send('emails.evaluations.model_broke',
['model_name' => $oModel->name, 'model_id' => $oModel->id],
function($message) {
    $message->to(Config::get('app.admin_email'),
'Администратор')->subject('Порог отсечения модели стал ниже минимального значения'));
});

// сохраним сущность в БД
return $oModel->save();
}

App/models/repos/client/EvaluationRepository.php
<?php

/**
 * Created by PhpStorm.
 * User: elluminate
 * Date: 23.02.15
 * Time: 15:44
 */
class EvaluationRepository {

    /**
     * @var ClientModelRepository - клиентский репозиторий моделей (решаемых задач)
     */
    protected $oModelRepo;

    public function __construct(ClientModelRepository $oModelRepo) {
        $this->oModelRepo = $oModelRepo;
    }

    /** добавляет факт совершения вычислений пользователем
     * @param $iModelId - id модели
     * @param $aCovariates - введенные пользователем параметров
     * @param $fEstimatedResult - просчитанный моделью результат
     * @throws \Illuminate\Exceptions\EvaluationException
     */
    public function addEvaluation($iModelId, $aCovariates, $fEstimatedResult) {
        // получим пользователя и проверим есть ли такой
        $iUserId = (int)Auth::user()->id;
        if (!$iUserId || !User::find($iUserId, ['id'])) throw new \Illuminate\Exceptions\EvaluationException("Не представлен идентификатор пользователя");
        // то же и с моделью
        $iModelId = (int)$iModelId;
        if (!$iModelId || !Model::find($iModelId, ['id'])) throw new \Illuminate\Exceptions\EvaluationException("Не представлен идентификатор модели");
        if (!is_array($aCovariates) || !count($aCovariates)) throw new \Illuminate\Exceptions\EvaluationException("Некорректный массив регрессоров");
        if ($fEstimatedResult > 1 || $fEstimatedResult < 0) throw new \Illuminate\Exceptions\EvaluationException("Результат вычисления логистической регрессии выходит за диапазон допустимых значений 0-1");
        $oModel = Model::with('duration')->find($iModelId, ['id', 'durations_id']);
        $iDuration = (int)$oModel->duration->duration;
        $fSill = $oModel->sill;
        unset($oModel);
        $oEvaluation = new Evaluation();
        // граница между Да и Нет находится не в 0,5, а в
        // абсциссе порога отсечения
        $oEvaluation->estimated_result = $fEstimatedResult <= $fSill ? 0 : 1;
        $oEvaluation->covariates = json_encode($aCovariates,
JSON_NUMERIC_CHECK);
    }
}

```

```

    // добавим к текущему времени интервал корректности
    // решения
    $oEvaluation->expired_moment = \Carbon\Carbon::now()->addHours($iDuration);
    $oEvaluation->user_id = $iUserId;
    $oEvaluation->model_id = $iModelId;
    // сохраним новую сущность
    $oEvaluation->save();
}

/**
 * получает уведомления об ожидающих и просроченных задачах
 */
public static function getNotifications()
{
    // просроченные задачи
    self::checkExpiredEvaluations();
    // ожидающие задачи
    self::checkWaitingEvaluations();
}

/**
 * проверяет пользователя на просроченные задачи
 */
public static function checkExpiredEvaluations()
{
    // сбросим предыдущий результат проверки
    Session::forget('expired_evaluations');
    $iUserId = \Auth::user()->id;
    // найдем соответствующие текущему пользователю вычисления с незаполненным реальным результатом и прошедшей датой и временем
    if (!$iUserId || !User::find($iUserId, ['id'])) return;
    $oEvaluations = Evaluation::where('user_id', '=', $iUserId)->whereNull('real_result')->where('expired_moment', '<', 'NOW()')->get();
    // если такие сущности нашлись, то запишем в сессию флагок
    if (!$oEvaluations->isEmpty())
    {
        Session::flash('expired_evaluations', 1);
        return true;
    }
    return false;
}

/**
 * проверяет пользователя на ожидающие вычисления
 */
public static function checkWaitingEvaluations()
{
    // если уже есть просроченные, то про ожидающие можно не говорить
    if (Session::get('expired_evaluations') == 1) return;
    // сбросим результат прошлой проверки
    Session::forget('waiting_evaluations');
    $iUserId = \Auth::user()->id;
    if (!$iUserId || !User::find($iUserId, ['id'])) return;
    // найдем для этого пользователя сущности с незаполненным реальным результатом
    $oEvaluations = Evaluation::where('user_id', '=', $iUserId)->whereNull('real_result')->get();
    // если такие есть, то запишем флагок в сессию
    if (!$oEvaluations->isEmpty())
    {
        Session::flash('waiting_evaluations', 1);
        return true;
    }
    return false;
}

/** собирает точно такую же форму, как заполнил ее пользователь
 * @param $iEvaluationId - id вычисления
 * @return array - массив полей формы
 */
public function getConfirmationForm($iEvaluationId)
{
    $iEvaluationId = (int)$iEvaluationId;
    if (!$iEvaluationId || !Evaluation::where('id', '=', $iEvaluationId)->where('user_id', '=', \Auth::user()->id)) throw new \Symfony\Component\HttpKernel\Exception\NotFoundHttpException();
    $oEvaluation = Evaluation::find($iEvaluationId);
    $iModelId = (int)$oEvaluation->model_id;
    // восстановим массив значений, которые пользователь вводил,
    // из БД
    $aValues = json_decode($oEvaluation->covariates);
    // соберем форму как и при решении задачи, только
}

```

```

подставим туда прошлые значения
$oForm = $this->oModelRepo->getApplyingForm($iModelId, $aValues);
return $oForm;
}

/** получает просчитанный моделью результат в случае конкретного вычисления
 * @param $iEvaluationId - id вычисления
 * @return mixed - результат
 */
public function getEstimatedResult($iEvaluationId)
{
    $iEvaluationId = (int)$iEvaluationId;
    // получим сущность, где пользователь равен текущему и равен требуемому
    if (!$iEvaluationId || !Evaluation::where('id', '=', $iEvaluationId)->where('user_id', '=', \Auth::user()->id)) throw new \Symfony\Component\HttpKernel\Exception\NotFoundHttpException();
    $oEvaluation = Evaluation::find($iEvaluationId);
    // если нашлось, то вернем результат
    if (!empty($oEvaluation)) return $oEvaluation->estimated_result;
    else throw new \Symfony\Component\HttpKernel\Exception\NotFoundHttpException();
}

/** завершает обратную связь вычисления
 * @param $iEvaluationId - id реальный результат
 * @return bool|null
 */
public function confirm($iEvaluationId, $iRealResult)
{
    $iEvaluationId = (int)$iEvaluationId;
    if (!$iEvaluationId || !Evaluation::where('id', '=', $iEvaluationId)->where('user_id', '=', \Auth::user()->id)) throw new \Symfony\Component\HttpKernel\Exception\NotFoundHttpException();
    $oEvaluation = Evaluation::find($iEvaluationId);
    $iRealResult = (int)$iRealResult;
    // -1 это когда у пользователя нет данных, в таком случае просто удалим это вычисление, пользы от него немного
    // иначе
    if ($iRealResult != -1)
    {
        // запишем реальный результат в БД
        $oEvaluation->real_result = $iRealResult;
        $bResult = $oEvaluation->save();
        unset($oEvaluation);
        // поставим в очередь задание на переобучение модели
        Queue::push('Elluminate\Workers\ModelTrainer', [
            'iEvaluationId' => $iEvaluationId
        ]);
        else
            $bResult = $oEvaluation->delete();
        // надо обновить уведомления о ждущих и просроченных задачах, может таких уже и нет
        self::getNotifications();
        return $bResult;
    }
}

App/models/repos/ModelRepository.php
<?php
/*
 * Created by PhpStorm.
 * User: elluminate
 * Date: 31.01.15
 * Time: 16:36
 */
class ModelRepository
{
    /**
     * @var математическая модель \Elluminate\Math\LogisticRegression
     */
    protected $oModel;

    /**
     * @var анализатор качества модели \Elluminate\Math\QualityAnalysis
     */
    protected $oQuality;

    public function __construct(\Elluminate\Math\LogisticRegression $oModel, \Elluminate\Math\QualityAnalysis $oQuality)
    {
        $this->oModel = $oModel;
        $this->oQuality = $oQuality;
    }
}

```

```

    /**
     * получает список моделей по id проблемной ситуации
     * @param $iSituationId - id проблемной ситуации
     * @return mixed - список моделей
     */
    public function getModelsList($iSituationId, $bActiveModels = false) {
        $iSituationId = (int)$iSituationId;
        // проверим есть ли такая ситуация
        if(!$iSituationId || !Situation::find($iSituationId, ['id'])) throw new \Symfony\Component\HttpKernel\Exception\NotFoundHttpException();
        // отдадим список связанных моделей
        if($bActiveModels)
            $oModels = Situation::find($iSituationId)->activeModels();
        >get(['id', 'name']);
        else
            $oModels = Situation::find($iSituationId)->models();
        >get(['id', 'name', 'threshold', 'min_threshold']);
        return $oModels;
    }

    /**
     * получает модель по id
     * @param $iModelId - id модели
     * @param array $aFields - список необходимых для выборки полей
     * @return \Illuminate\Support\Collection|mixed|null|static - модель
     */
    public function getModel($iModelId, $aFields = ['*']) {
        $iModelId = (int)$iModelId;
        // проверим есть ли такая модель
        if(!$iModelId || !Model::find($iModelId, ['id'])) throw new \Symfony\Component\HttpKernel\Exception\NotFoundHttpException();
        // найдем ее и вернем
        $oModel = Model::find($iModelId, $aFields);
        if(isset($oModel[0]))
            return $oModel[0];
        return $oModel;
    }
}
App/models/repos/SituationRepository.php
<?php

```

```

/*
*      Created      by      PhpStorm.
*      User:          elluminate
*      Date:         30.01.15
*      Time:        15:34
*/
class SituationRepository
{
    /**
     * получает список проблемных ситуаций
     * @param $iParentSituationId - id ситуации, наследников которой надо получить
     * @return \Illuminate\Database\Eloquent\Collection|static[]
     */
    public function getSituationsList($iParentSituationId, $bNeedModels = false, $bNeedChildren = false, $bActiveModels = false) {
        $iParentSituationId = (int)$iParentSituationId;
        // если это не верхний уровень и нет такого раздела, значит нас хотят обмануть
        // или просто такой раздел не существует, отдадим 404 ошибку
        if ($iParentSituationId !== 0 && !Situation::find($iParentSituationId, ['id'])) throw new \Symfony\Component\HttpKernel\Exception\NotFoundHttpException();
        // если верхний уровень, отдадим разделы, у которых родителя нет
        if (!($iParentSituationId))
            $oSituations = Situation::whereNull('parent_id')->get();
        // иначе найдем наследников
        else
            $oSituations = Situation::find($iParentSituationId)->children()->get();
        if($bNeedModels)
            if
                $oSituations->load('activeModels');
            else
                $oSituations->load('modelsId');
        }
        if($bNeedChildren)
            $oSituations->load('children');
    }
}

```

```

        return $oSituations;
    }
}
App/models/repos/UserRepository.php
<?php

/**
*          Class           UserRepository
*
* This service abstracts some interactions that occurs between Confide and the Database.
*/
class UserRepository
{
    /**
     * Signup a new account with the given parameters
     * @param array $input Array containing 'username', 'email' and 'password'.
     * @return User object that may or may not be saved successfully. Check the id to make sure.
     */
    public function signup($input) {
        $user = new User();
        $user->username = array_get($input, 'username');
        $user->email = array_get($input, 'email');
        $user->password = array_get($input, 'password');

        // The password confirmation will be removed from model before saving. This field will be used in Ardent's auto validation.
        $user->password_confirmation = array_get($input, 'password_confirmation');

        // Generate a random confirmation code
        $user->confirmation_code = md5(uniqid(mt_rand(), true));

        // Save if valid. Password field will be hashed before save
        $this->save($user);

        return $user;
    }

    /**
     * Attempts to login with the given credentials.
     * @param array $input Array containing the credentials (email/username and password)
     * @return boolean Success?
     */
    public function login($input) {
        if (!isset($input['password'])) {
            $input['password'] = null;
        }

        return Confide::logAttempt($input, Config::get('confide::signup_confirm'));
    }

    /**
     * Checks if the credentials has been throttled by too much failed login attempts
     * @param array $credentials Array containing the credentials (email/username and password)
     * @return boolean Is throttled
     */
    public function isThrottled($input) {
        return Confide::isThrottled($input);
    }

    /**
     * Checks if the given credentials correponds to a user that exists but is not confirmed
     * @param array $credentials Array containing the credentials (email/username and password)
     */
    public function isConfirmed($input) {
        return Confide::isConfirmed($input);
    }
}

```

```

/*
 * @return boolean Exists and is not confirmed?
 */
public function existsButNotConfirmed($input)
{
    $user = Confide::getUserByEmailOrUsername($input);

    if ($user) {
        $correctPassword = Hash::check(
            isset($input['password']) ? $input['password'] : false,
            $user->password
        );
    }

    return (! $user->confirmed && $correctPassword);
}

/**
 * Resets a password of a user. The $input['token'] will tell which user.
 *
 * @param array $input Array containing 'token', 'password' and 'password_confirmation' keys.
 */
public function resetPassword($input)
{
    $result = false;
    $user = Confide::userByResetPasswordToken($input['token']);

    if ($user) {
        $user->password = $input['password'];
        $user->password_confirmation = $input['password_confirmation'];
        $result = $this->save($user);
    }

    // If result is positive, destroy token
    if ($result) {
        Confide::destroyForgotPasswordToken($input['token']);
    }
}

return $result;
}

/**
 * Simply saves the given instance
 *
 * @param User $instance
 */
public function save(User $instance)
{
    return $instance->save();
}

App/models/Duration.php
<?php
/** Created by PhpStorm. elluminate 30.01.15 12:51 */
class Duration extends Eloquent {

    /**
     * @var bool - не надо использовать поля "создана в" и "изменено в", которые включены по умолчанию
     */
    public $timestamps = false;

    public function models()
    {
        return $this->hasMany('Model');
    }
}

App/models/Evaluation.php
<?php
/** Created by PhpStorm. elluminate 30.01.15 12:51 */

```

```

class Evaluation extends Eloquent {

    /**
     * @var bool - не надо использовать поля "создана в" и "изменено в", которые включены по умолчанию
     */
    public $timestamps = false;

    protected static $aFieldNames = [
        'evaluation_id' => 'Идентификатор',
        'real_result' => 'Реальный результат'
    ];

    protected static $aValidRules = [
        'evaluation_id' => [
            'real_result' => [
                'Integer',
                'in:-1,0,1'
            ]
        ]
    ];

    public function model()
    {
        return $this->belongsTo('Model')->select('id', 'name');
    }

    public static function validate($input)
    {
        return Validator::make($input, self::$aValidRules, array(),
            self::$aFieldNames);
    }
}

App/models/Model.php
<?php
/** Created by PhpStorm. elluminate 27.01.15 15:11 */
class Model extends Eloquent {

    /**
     * @var bool - не надо использовать поля "создана в" и "изменено в", которые включены по умолчанию
     */
    public $timestamps = false;

    protected static $aFieldNames = [
        'id' => 'Идентификатор',
        'name' => 'Название',
        'duration' => 'Время корректности решения',
        'min_threshold' => 'Минимальный порог отсечения',
        'comment' => 'Информация о задаче',
        'train_file' => 'Файл обучающей выборки',
        'situation_id' => 'Идентификатор проблемной ситуации'
    ];

    protected static $aValidRules = [
        'id' => [
            'Required|Min:5',
            'Integer'
        ],
        'name' => [
            'Required|Integer'
        ],
        'duration' => [
            'Required|Between:0,100',
            'Integer'
        ],
        'min_threshold' => [
            'Required|Between:0,100',
            'Integer'
        ],
        'situation_id' => [
            'Required_without:id|integer'
        ],
        'train_file' => [
            'Required_without:id|mimes:xls,xlsx'
        ]
    ];

    public function duration()
    {
        return $this->belongsTo('Duration', 'durations_id');
    }

    public function situation()
    {
        return $this->belongsTo('Situation');
    }

    public static function validate($input)
    {
        return Validator::make($input, self::$aValidRules, array(),
            self::$aFieldNames);
    }
}

App/models/Role.php
<?php
/** Created by PhpStorm. elluminate 19.01.15 12:56 */
use Zizaco\Entrust\EntrustRole;

```

```

class Role extends EntrustRole
{
}

App/models/Situation.php
<?php
/**
 * Created by PhpStorm.
 * User: elluminate
 * Date: 20.01.15
 * Time: 15:45
 */
class Situation extends Eloquent {

    /**
     * @var bool - не надо использовать поля "создана в" и
     * "изменено в", которые включены по умолчанию
     */
    public $timestamps = false;

    protected static $aFieldNames = [
        'id' => 'Идентификатор',
        'name' => 'Название',
        'okved_correspondence' => 'Соответствие ОКВЭД',
        'parent_id' => 'Номер родительской ситуации'
    ];

    protected static $aValidRules = [
        'id' => 'Integer',
        'name' => 'Required|Min:5',
        'parent_id' => 'Integer',
        'okved_correspondence' => 'Min:5|Max:16'
    ];
}

/** получает родителя записи (реализована рекурсивная
 связь)
 * @return \Illuminate\Database\Eloquent\Relations\BelongsTo
 */
public function parent()
{
    return $this->belongsTo('Situation', 'parent_id', 'id');
}

/** получает наследников записи (реализована рекурсивная
 связь)
 * @return \Illuminate\Database\Eloquent\Relations\HasMany
 */
public function children()
{
    return $this->hasMany('Situation', 'parent_id', 'id');
}

public function models()
{
    return $this->hasMany('Model');
}

public function activeModels()
{
    return $this->hasMany('Model')->where('threshold', '>=',
DB::raw('min_threshold'));
}

public function modelsId()
{
    return $this->modelsReqFields(['id', 'situation_id']);
}

// иногда не нужно тащить всю модель за собой, там же
огромные массивы
public function modelsReqFields($reqFields = [])
{
    if(!is_array($reqFields) || !count($reqFields))
        return $this->models();
    else
        return $this->hasMany('Model')->select($reqFields);
}

public static function validate($input)
{
    return Validator::make($input, self::$aValidRules, array(),
self::$aFieldNames);
}
}

App/models/User.php
<?php

use Zizaco\Confide\ConfideUser;
use Zizaco\Confide\ConfideUserInterface;
use Zizaco\Entrust\HasRole;

class User extends Eloquent implements ConfideUserInterface
{

```

```

use use
ConfideUser;
HasRole;

}
App/views/admin/forms/addDome.blade.php
<div class="row">
<div class="col-lg-5">
<div class="alert alert-dismissible alert-success">
<button type="button" class="close" data-dismiss="alert"><x>
Запись успешно добавлена!
</div>
</div>
</div>
App/views/admin/models/create.blade.php
@extends('layouts.master')
@section('styles')
{{ HTML::style('assets/css/bootstrap-slider.css') }}
@stop
@section('title')
Администрирование. Добавление решаемой задачи
@stop
@section('content')
{{ Breadcrumbs::render('admin.models', $hierarchy, 'create') }}
<div><a href="{{ URL::route('models.template') }}">Скачать шаблон файла обучающей выборки</a></div>
{{ Form::open(['route' => 'models.store', 'class' => 'form-horizontal', 'files' => true]) }}
<fieldset>
<legend>Добавление задачи</legend>
<div class="form-group">
{{ Form::label('name', 'Название', ['class' => 'col-lg-2 control-label']) }}
<div class="col-lg-5">
{{ Form::text('name', null, ['id' => 'name', 'class' => 'form-control', 'placeholder' => 'Название']) }}
</div>
</div>
<div class="form-group">
{{ Form::label('duration', 'Время корректности решения', ['class' => 'col-lg-2 control-label']) }}
<div class="col-lg-5">
{{ Form::select('duration', $durations, null, ['class' => 'form-control']) }}
</div>
</div>
<div class="form-group">
{{ Form::label('min_threshold', 'Минимальный порог отсечения', ['class' => 'col-lg-2 control-label']) }}
<div class="col-lg-5">
<input name="min_threshold" class="js_range_slider" data-slider-id="threshold_slider" type="text" data-slider-min="50" data-slider-max="100" data-slider-step="1" data-slider-value="75"/>
</div>
</div>
<div class="form-group">
{{ Form::label('comment', 'Информация о задаче', ['class' => 'col-lg-2 control-label']) }}
<div class="col-lg-5">
{{ Form::textarea('comment', null, ['class' => 'form-control', 'rows' => 4]) }}
</div>
</div>
<div class="form-group">
{{ Form::label('file', 'Файл обучающей выборки', ['class' => 'col-lg-2 control-label']) }}
<div class="col-lg-5">
{{ Form::file('train_file') }}
</div>
</div>
<div class="form-group">
{{ Form::hidden('situation_id', $situation_id) }}
<div class="col-lg-10 col-lg-offset-2">
{{ Form::submit('Добавить', ['class' => 'btn btn-primary']) }}
{{ Form::reset('Отмена', ['class' => 'btn btn-default']) }}
</div>
</div>
</div>
</div>
</div>
</div>
{{ Form::close() }}
@if (@if ($errors->all() as $error))
<div class="col-lg-5">
<div class="alert alert-dismissible alert-danger">
<button type="button" class="close" data-dismiss="alert"><x>

```

```

</td></td>
@if(!empty($model->elastic_coeff))
@foreach($model->elastic_coeff as $elastic_coeff)
<td> {{ $elastic_coeff }} </td>
@endforeach
@endif
</tr>
<tr>
<th><i class="fa fa-signal"></i> Стандартизованные
коэффициенты</th>
<td></td>
@if(!empty($model->std_coeff))
@foreach($model->std_coeff as $std_coeff)
<td> {{ $std_coeff }} </td>
@endforeach
@endif
</tr>
</tbody>
</table>
</div>
</div>
@stop
App/views/admin/models/edit.blade.php
@extends('layouts.master')
@section('title')
Администрирование. Редактирование задачи {{ $model->name }}
@stop
@section('styles')
{{ HTML::style('assets/css/bootstrap-slider.css') }}
@stop
@section('content')
{{ Breadcrumbs::render('admin.models', $hierarchy, 'edit', $model->id) }}
{{ Form::model($model, ['route' => ['models.update'], 'class' => 'form-horizontal', 'files' => true]) }}
<fieldset>
<legend>Редактирование параметров задачи</legend>
<div class="form-group">
{{ Form::label('name', 'Название', ['class' => 'col-lg-2 control-label']) }}
<div class="col-lg-5">
{{ Form::text('name', null, ['id' => 'name', 'class' => 'form-control', 'placeholder' => 'Название']) }}
</div>
</div>
<div class="form-group">
{{ Form::label('duration', 'Время корректности решения', ['class' => 'col-lg-2 control-label']) }}
<div class="col-lg-5">
{{ Form::select('duration', $durations, $model->durations_id, ['class' => 'form-control']) }}
</div>
</div>
<div class="form-group">
{{ Form::label('min_threshold', 'Минимальный порог отсечения', ['class' => 'col-lg-2 control-label']) }}
<div class="col-lg-5">
<input name='min_threshold' class="js_range_slider" data-slider-id='threshold_slider' type="text" data-slider-min="50" data-slider-max="100" data-slider-step="1" data-slider-value="{{ $model->min_threshold }}"/>
</div>
</div>
<div class="form-group">
{{ Form::label('comment', 'Информация о задаче', ['class' => 'col-lg-2 control-label']) }}
<div class="col-lg-5">
{{ Form::textarea('comment', null, ['class' => 'form-control', 'rows' => 4]) }}
</div>
</div>
<div class="form-group">
{{ Form::label('file', 'Файл обучающей выборки', ['class' => 'col-lg-2 control-label']) }}
<div class="col-lg-5">
{{ Form::file('train_file') }}
<span class="help-block"><p class="text-warning">Внимание! При загрузке файла модель будет переобучена. При этом данные дополнительной выборки будут утеряны.</p>
<p class="text-warning">Кроме того будут удалены пользовательские вычисления для этой модели.</p>
<p class="text-warning">Если Вы не хотите сейчас переобучать модель, просто оставьте это поле незаполненным.</p>
</span>
</div>

```

```
</div>
{{ Form::hidden('model_id', $model->id) }}
<div class="form-group">
<div class="col-lg-10 col-lg-offset-2">
{{ Form::submit('Изменить', ['class' => 'btn btn-primary'])}}
{{ Form::reset('Отмена', ['class' => 'btn btn-default'])}}
</div>
</div>
</fieldset>
{{ Form::close() }}
@if ( !$errors->isEmpty() )
@foreach($errors->all() as $error)
<div class="col-lg-5">
<div class="alert alert-dismissible alert-danger">
<button type="button" class="close" data-dismiss="alert">x</button>
{{ $error }}
</div>
</div>
@endforeach
@stop

@section('scripts')
{{ HTML::script('assets/js/bootstrap-slider.js') }}
{{ HTML::script('assets/js/slider.js') }}
{{--{{ HTML::script('assets/js/checkFile.js') }}--}}
@stop
App/views/admin/models/inactive.blade.php
@extends('layouts.master')
@section('title')
Администрирование. Список неактивных задач
@stop
@section('content')
{{ Breadcrumbs::render('models.inactive') }}
<div class="page-header">
<h3><i class="fa fa-angle-double-down"></i> Список неактивных задач</h3>
</div>
<div style="margin-bottom: 20px">
<table class="table table-striped table-hover">
<thead>
<tr>
<th>Название</th>
</tr>
</thead>
<tbody>
@if(!$models->isEmpty())
@foreach($models as $model)
<tr>
<td class="td-align_left"><a href="{{ URL::route('models.detail', ['iModelId' => $model->id]) }}>{{ $model->name }}</a></td>
</tr>
@endif
</tbody>
</table>

```

```

        <th>Редактирование
        <th>Удалить
    </tr>
</thead>
<tbody>
@if(!$models->isEmpty())
@foreach($models as $model)
<tr>
    <td class="td-align_left">@if($model->threshold < $model->min_threshold)<i class="fa fa-lock"></i>@endif <a href="{{ URL::route('models.detail', ['iModelId' => $model->id]) }}>{{ $model->name }}</a></td>
    <td><a title="Редактировать параметры" href="{{ URL::route('models.edit', ['iModelId' => $model->id]) }}" class="btn btn-primary"><i class="fa fa-pencil"></i></a></td>
    <td><a title="Удалить задачу" data-href="{{ URL::route('models.destroy', ['iModelId' => $model->id]) }}" class="btn btn-danger" type="button" data-toggle="modal" data-target="#delModal"><i class="fa fa-times"></i></a></td>
</tr>
@endforeach
@endif
</tbody>
</table>
<div class="modal fade" id="delModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span>×</span></button>
                <h4 id="myModalLabel">Подтверждение удаления</h4>
            </div>
            <div class="modal-body">
                Действительно хотите удалить эту запись?
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-default" data-dismiss="modal">Нет</button>
                <a href="#" class="btn btn-danger danger">Да</a>
            </div>
        </div>
    </div>
</div>
@stop

@section('scripts')
    {{ HTML::script('assets/js/list.js') }}
@stop
App\views\admin\situations\create.blade.php
@extends('layouts.master')
@section('title')
    Администрирование. Добавление проблемной ситуации
@stop
@section('content')
    {{ Breadcrumbs::render('admin.situations', $hierarchy, 'create') }}
    {{ Form::open(['route' => 'situations.store', 'class' => 'form-horizontal']) }}
    <fieldset>
        <legend>Добавление ситуации</legend>
        <div class="form-group">
            {{ Form::label('name', 'Название', ['class' => 'col-lg-2 control-label']) }}
            <div class="col-lg-5">
                {{ Form::text('name', null, ['id' => 'name', 'class' => 'form-control', 'placeholder' => 'Название']) }}
            </div>
        </div>
        <div class="form-group">
            {{ Form::label('okved_correspondence', 'Соответствие ОКВЭД', ['class' => 'col-lg-2 control-label']) }}
            <div class="col-lg-5">
                {{ Form::text('okved_correspondence', null, ['id' => 'okved_correspondence', 'class' => 'form-control', 'placeholder' => 'Соответствие ОКВЭД']) }}
            </div>
        </div>
        {{ Form::hidden('parent_id', $parent_situation_id) }}
        <div class="form-group">
            <div class="col-lg-10 col-lg-offset-2">
                {{ Form::submit('Добавить', ['class' => 'btn btn-primary']) }}
                {{ Form::reset('Отмена', ['class' => 'btn btn-default']) }}
            </div>
        </div>
    {{ Form::open(['route' => 'situations.store', 'class' => 'form-horizontal']) }}

```

```

        </div>
    </div>
<fieldset>
{{ Form::close() }}}
@if ( !$errors->isEmpty() )
@foreach($errors->all() as $error)
<div class="alert alert-dismissible alert-danger">
    <button type="button" class="close" data-dismiss="alert">×</button>
    {{ $error }}.
</div>
</div>
@endforeach
@endif
@stop
App/views/admin/situations/edit.blade.php
@extends('layouts.master')
@section('title')
    Администрирование. Редактирование проблемной ситуации
{{ $situation->name }}
@stop
@section('content')
    {{ Breadcrumbs::render('admin.situations', $hierarchy, 'edit') }}
    {{ Form::model($situation, ['route' => ['situations.update'], 'class' => 'form-horizontal']) }}
    <fieldset>
        <legend>Изменение реквизитов ситуации</legend>
        <div class="form-group">
            {{ Form::label('name', 'Название', ['class' => 'col-lg-2 control-label']) }}
            <div class="col-lg-5">
                {{ Form::text('name', null, ['id' => 'name', 'class' => 'form-control', 'placeholder' => 'Название']) }}
            </div>
        </div>
        <div class="form-group">
            {{ Form::label('okved_correspondence', 'Соответствие ОКВЭД', ['class' => 'col-lg-2 control-label']) }}
            <div class="col-lg-5">
                {{ Form::text('okved_correspondence', null, ['id' => 'okved_correspondence', 'class' => 'form-control', 'placeholder' => 'Соответствие ОКВЭД']) }}
            </div>
        </div>
        {{ Form::hidden('situation_id', $situation->id) }}
        <div class="form-group">
            <div class="col-lg-10 col-lg-offset-2">
                {{ Form::submit('Изменить', ['class' => 'btn btn-primary']) }}
                {{ Form::reset('Отмена', ['class' => 'btn btn-default']) }}
            </div>
        </div>
    </fieldset>
    {{ Form::close() }}
    @foreach($errors->all() as $error)
    <div class="alert alert-dismissible alert-danger">
        <button type="button" class="close" data-dismiss="alert">×</button>
        {{ $error }}.
    </div>
    </div>
@endforeach
@endif
@stop
App/views/admin/situations/index.blade.php
@extends('layouts.master')
@section('title')
    Администрирование. Список проблемных ситуаций
@stop
@section('content')
    {{--немного магии, взяли из сессии имя шаблона результата операции (удаление, изменение, добавление) и отрисовали его вот тут--}}
    @if ( Session::get('form_result') )
        @include('admin.forms.'.Session::get('form_result'))
    @endif
    <div class="page-header">
        <h3><i class="fa fa-angle-double-down"></i> Каталог проблемных ситуаций</h3>
    </div>
    {{ Breadcrumbs::render('admin.situations', $hierarchy, 'list') }}
    <a title="Добавить ситуацию" href="{{ URL::route('situations.create', ['iParentSituationId' => $parent_situation]) }}" class="btn btn-primary"><i class="fa fa-plus-circle"></i> Добавить ситуацию</a>

```

```

URL::route('situations.create', ['iParentSituationId' => $parent_situation]) }}" class="btn btn-primary"><i class="fa fa-plus-circle"></i> Добавить ситуацию</a>
<table class="table table-striped table-hover">
    <thead>
        <tr>
            <th>Название</th>
            <th>Соответствие ОКВЭД</th>
            <th>Редактировать реквизиты</th>
            <th>Удалить</th>
            <th>Решаемые задачи</th>
        </tr>
    </thead>
    <tbody>
        @if(!$situations->isEmpty())
            @foreach($situations as $situation)
                <tr>
                    <td class="td-align-left"><a href="{{ URL::route('situations.list', ['iParentSituationId' => $situation->id]) }}"><i class="fa fa-level-down"></i> {{ $situation->name }}</a></td>
                    <td>{{ $situation->okved_correspondence }}</td>
                    <td><a title="Редактировать реквизиты записи" href="{{ URL::route('situations.edit', ['iSituationId' => $situation->id]) }}" class="btn btn-primary"><i class="fa fa-pencil"></i></a></td>
                    <td><a title="Удалить запись" data-href="{{ URL::route('situations.destroy', ['iSituationId' => $situation->id]) }}" class="btn btn-danger" type="button" data-toggle="modal" data-target="#delModal"><i class="fa fa-times"></i></a></td>
                    <td><a title="Решаемые задачи" href="{{ URL::route('models.list', ['iSituationId' => $situation->id]) }}" class="btn btn-warning" data-href="#">{{ if( isset($situation->modelsId) && count($situation->modelsId) ) <span class="badge">{{ count($situation->modelsId) }}</span> }>@else <span class="badge">0</span> @endif </a>
                </td>
            </tr>
        @endforeach
    @endif
    </tbody>
</table>
<div class="modal fade" id="delModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="true">&times;</span></button>
                <h4 id="myModalLabel">Подтверждение удаления</h4>
            </div>
            <div class="modal-body">
                Действительно хотите удалить эту запись?
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-default" data-dismiss="modal">Нет</button>
                <a href="#" class="btn btn-danger danger">Да</a>
            </div>
        </div>
    </div>
@stop
@section('scripts')
    {{ HTML::script('assets/js/list.js') }}
@stop
App/views/client/evaluations/detail.blade.php
@extends('layouts.master')
@section('title')
    Обратная связь
@stop
@section('content')
    {{--<div class="col-lg-row">--}}
    {{--<blockquote>--}}
        {{--<p>{{ $model->name }}</p>--}}
        {{--<small>{{ $model->comment }}</small>--}}
    {{--</blockquote>--}}

```

```
{--</div>--}}}
{{ Breadcrumbs::render('evaluations', 'detail', $model_id) }}
<div class="col-lg-4">
<ul class="list-group">
<li class="list-group-item">
<div>Результат, полученный моделью <span class="badge">{{ $estimated_result }} {{ $estimated_result ? ' (Да)' : ' (Нет)' }}</span></div>
</li>
</ul>
</div>
<div class="col-lg-10">
{{ Form::open(array('route' => 'evaluations.confirm', 'class' => 'form-horizontal')) }}
<fieldset>
<legend>Выберите фактический результат</legend>
<div class="form-group">
{{ Form::label('real_result', 'Реальный результат', ['class' => 'control-label']) }}
<div 'col-lg-2' class="col-lg-5">
{{ Form::select('real_result', [-1 => 'Нет сведений', '0' => '0 - Нет', '1' => '1 - Да'], null, ['class' => 'form-control']) }}
<span class="help-block">1 - Да; 0 - Нет</span>
</div>
</div>
</div>
<div class="form-group">
{{ Form::hidden('evaluation_id', $iEvaluationId) }}
<div class="col-lg-10 col-lg-offset-2">
{{ Form::submit('Ok', ['class' => 'btn btn-primary']) }}
</div>
</div>
</div>
<div class="text-primary" id="eval_formCollapse">Показать параметров</p>
значения
</div>
@if ( !$errors->isEmpty() )
@foreach($errors->all() as $error)
<div class="col-lg-5">
<div class="alert alert-dismissible alert-danger">
<button type="button" class="close" data-dismiss="alert"></button>
{{ $error }}
</div>
</div>
@endif
@forelse($errors->all() as $error)
<div class="col-lg-5">
<div class="alert alert-dismissible alert-danger">
<button type="button" class="close" data-dismiss="alert"></button>
{{ $error }}
</div>
</div>
@else
<div class="collapse" id="eval_form" style="display: none;">
<div class="col-lg-10 row">
{{ Form::open(array('class' => 'form-horizontal')) }}
<div class="form-group">
@if(!empty($form))
@forelse($form as $field)
<div class="col-lg-2 control-label">
{{ Form::label($field['tech_name'], $field['name'], ['class' => 'control-label']) }}
<div class="col-lg-5">
{{ Form::number($field['tech_name'], null, ['id' => $field['tech_name'], 'class' => 'form-control', 'placeholder' => $field['value'], 'disabled']) }}
<span class="help-block" style="margin-left: 10px;">{{ $field['comment'] }}
</div>
</div>
@else
<div class="col-lg-10">
{{ Form::close() }}
</div>
@endif
</div>
</div>
@else
<div class="col-lg-10">
{{ Form::close() }}
</div>
@endif
</div>
</div>
</div>
@stop
@section('scripts')
{{ HTML::script('assets/js/evaluation.js') }}
@stop
App/views/client/evaluations/index.blade.php
@extends('layouts.master')
@section('title')
Задачи, ожидающие обратной связи
@stop
@section('content')
{{ Breadcrumbs::render('evaluations', 'list') }}
<div class="col-lg-10">
<div class="page-header">
<h3><i class="fa fa-angle-double-down"></i> Решения, которые ждут Вашей обратной связи</h3>

```

```
</div>
<table class="table table-striped table-hover">
    <thead>
        <tr>
            <th>Задача</th>
            <th>Окончание времени корректности решения</th>
            <th>Перейти к обратной связи</th>
        </tr>
    </thead>
    <tbody>
        @if(!$evaluations->isEmpty())
        @foreach($evaluations as $evaluation)
        <tr>
            <td>{{ $evaluation->model->name }}</td>
            <td>{{ $evaluation->expired_moment }}</td>
            <td><a title="Перейти к обратной связи" href="{{ URL::route('evaluations.detail', ['iEvaluationId' => $evaluation->id] ) }}" class="btn btn-primary"><i class="fa fa-send-o"></i></a></td>
        </tr>
        @endforeach
        @endif
    </tbody>
</table>
</div>
@stop
App/views/client/evaluations/success.blade.php
@extends('layouts.master')
@section('title')
    Результат обратной связи
@stop
@section('content')
    <div class="col-lg-10">
        <div class="alert alert-dismissible alert-success">
            <button type="button" class="close" data-dismiss="alert">x</button>
            <strong>Всё прошло успешно!</strong> Спасибо за Вашу обратную связь, теперь Вы можете <a class="alert-link" href="{{ URL::route('problems.list') }}>решать задачи без ограничений</a>.
        </div>
    </div>
</div>
@stop
App/views/client/models/detail.blade.php
@extends('layouts.master')
@section('title')
    Решаемая задача {{ $model->name }}
@stop
@section('content')
    {{ Breadcrumbs::render('client.models', $hierarchy, 'detail', $model->id) }}
    <blockquote>
        <p>{{ $model->name }}</p>
        <small>{{ $model->comment }}</small>
        <small>Время корректности решения: {{ $duration }}</small>
    </blockquote>
    {{ Form::open(array('route' => 'tasks.compute', 'class' => 'form-horizontal')) }}
    <fieldset>
        <legend>Введите значения параметров для решения задачи</legend>
        @if(!empty($form))
        @foreach($form as $field)
            <div class="form-group">
                {{ Form::label($field['tech_name'], $field['name'], ['class' => 'col-lg-2 control-label']) }}
                <div class="col-lg-5">
                    {{ Form::number($field['tech_name'], null, ['id' => $field['tech_name'], 'class' => 'form-control', 'placeholder' => $field['comment']] ) }}
                <span class="help-block">{{ $field['comment'] }}</span>
            </div>
        </div>
        @endforeach
        @endif
        <div class="form-group">
            {{ Form::hidden('model_id', $model->id) }}
            <div class="form-group">
                <div class="col-lg-10 col-lg-offset-1">
                    {{ Form::submit('Решить', ['class' => 'btn btn-primary']) }}
                    {{ Form::reset('Отмена', ['class' => 'btn btn-default']) }}
                </div>
            </div>
        </div>
    </fieldset>
```

```

{{ Form::close() }}
@if( !$errors->isEmpty() )
@foreach($errors->all() as $error)
<div class="alert alert-dismissible alert-danger">
    <button type="button" class="close" data-dismiss="alert">x</button>
    {{ $error }}.
</div>
@endforeach
@endif
@stop
App/views/client/models/index.blade.php
@extends('layouts.master')
@section('title')
    Список решаемых задач
@stop
@section('content')
    {{ Breadcrumbs::render('client.models', $hierarchy, 'list') }}
    <div class="page-header">
        <h3><i class="fa fa-angle-double-down"></i> Список решаемых задач</h3>
    </div>
    <table class="table table-striped table-hover">
        <thead>
            <tr>
                <th>Название</th>
            </tr>
        </thead>
        <tbody>
            @if(!$models->isEmpty())
            @foreach($models as $model)
                <tr>
                    <td class="td-align_left"><a href="{{ URL::route('tasks.detail', ['iModelId' => $model->id]) }}">{{ $model->name }}</a></td>
                </tr>
            @endforeach
            @endif
        </tbody>
    </table>
@stop
App/views/client/models/result.blade.php
@extends('layouts.master')
@section('title')
    Результаты решения
@stop
@section('content')
    {{ Breadcrumbs::render('client.models', $hierarchy, 'detail', $model_id) }}
    <blockquote>
        <p>{{ $reg_name }}</p>
        <small>{{ $comment }}</small>
    </blockquote>
    <div class="col-lg-5">
        <span class="label label-danger">0 - Нет</span>
        <span class="label label-success">1 - Да</span>
    <div class="progress">
        <div class="progress-bar" style="width: {{ round($result*100) }}%>{{ $result }}</div>
    </div>
    </div>
@stop
App/views/client/notifications/expired_evaluations.blade.php
<div class="alert alert-dismissible alert-danger">
    <button type="button" class="close" data-dismiss="alert">x</button>
    <p><i class="fa fa-hand-o-right"></i> У Вас имеются <a class="alert-link" href="{{ URL::route('evaluations.list') }}>задачи, срок подтверждения которых истек.</a></p>
    <p><i class="fa fa-exclamation-triangle"></i> Возможность решения задач классификации временно заблокирована для Вас до момента, пока Вы не осуществите обратную связь</p>
</div>
App/views/client/notifications/expired_redirect.blade.php
@extends('layouts.master')
@section('content')
    <div class="alert alert-dismissible alert-danger">
        <button type="button" class="close" data-dismiss="alert">x</button>
        <p><i class="fa fa-hand-o-right"></i> У Вас имеются <a class="alert-link" href="{{ URL::route('evaluations.list') }}>задачи, срок подтверждения которых истек.</a></p>
        <p><i class="fa fa-exclamation-triangle"></i> Возможность решения задач классификации временно заблокирована для

```

Вас до момента, пока Вы не осуществите обратную связь</p>
</div>
@stop
App/views/client/notifications/waiting_evaluations.blade.php
<div class="alert alert-dismissible alert-warning">
 <button type="button" class="close" data-dismiss="alert">x</button>
 <p><i class="fa fa-hand-o-right"></i> У Вас имеются задачи, которые ожидают Вашей обратной связи</p>
</div>
App/views/client/search/index.blade.php
@extends('layouts.master')
@section('title')
 Результаты поиска
@stop
@section('content')
 <h4 class="page-header">Совпадения по коду ОКВЭД</h4>
 @if(\$okved_code->isEmpty())
 <div class="row">
 <div class="col-lg-5">
 <div class="alert alert-dismissible alert-danger">
 <button type="button" class="close" data-dismiss="alert">x</button>
 <p>Совпадений по коду ОКВЭД не найдено.</p>
 </div>
 </div>
 </div>
 @else
 <ul class="list-group">
 @foreach(\$okved_code as \$value)
 <li class="list-group-item">
 \$value->parent_id]) }}>{{ \$value->name }}
 {{ \$value->okved_correspondence }}

 @endforeach

 @endif
 @if(\$code_overlimit)
 <div class="row">
 <div class="col-lg-7">
 <div class="alert alert-dismissible alert-info">
 <button type="button" class="close" data-dismiss="alert">x</button>
 <p>Здесь показаны не все совпадения по коду ОКВЭД, их слишком много. Пожалуйста, уточните условия поиска.</p>
 </div>
 </div>
 </div>
 @endif
 <h4 class="page-header">Совпадения по названию проблемной ситуации</h4>
 @if(\$situation_name->isEmpty())
 <div class="row">
 <div class="col-lg-5">
 <div class="alert alert-dismissible alert-danger">
 <button type="button" class="close" data-dismiss="alert">x</button>
 <p>Совпадений по названию проблемной ситуации найдено.</p>
 </div>
 </div>
 </div>
 @else
 <ul class="list-group">
 @foreach(\$situation_name as \$value)
 <li class="list-group-item">
 \$value->parent_id]) }}>{{ \$value->name }}
 {{ \$value->okved_correspondence }}

 @endforeach

 @endif
 @if(\$situation_overlimit)
 <div class="row">
 <div class="col-lg-7">
 <div class="alert alert-dismissible alert-info">
 <button type="button" class="close" data-dismiss="alert">x</button>

```

    Здесь показаны не все совпадения по названию
проблемной ситуации, их слишком много.
Пожалуйста, уточните условия поиска.

    </div>
</div>
@endif
<h4 class="page-header">Совпадения по названию
решаемой проблемы</h4>
@if($model_name->isEmpty())
<div class="row">
<div class="col-lg-5">
<div class="alert alert-dismissible alert-danger">
<button type="button" class="close" data-
dismiss="alert">x</button>
Совпадений по названию решаемой проблемы
не
</div>
</div>
</div>
@else
<ul class="list-group">
@foreach($model_name as $value)
<li class="list-group-item">
<a href="{{ URL::route('tasks.detail', ['iTaskId' =>
$value->id]) }}>{{ $value->name }}</a>
</li>
@endforeach
</ul>
@endif
@if($model_overlimit)
<div class="row">
<div class="col-lg-7">
<div class="alert alert-dismissible alert-info">
<button type="button" class="close" data-
dismiss="alert">x</button>
Здесь показаны не все совпадения, их слишком
много.
Пожалуйста, уточните условия
поиска.
</div>
</div>
@endif
@stop
App/views/client/situations/index.blade.php
@extends('layouts.master')
@section('title')
    Список проблемных ситуаций
@stop
@section('content')
    <div class="page-header">
        <h3><i class="fa fa-angle-double-down"></i> Каталог
проблемных ситуаций</h3>
    </div>
    {{ Breadcrumbs::render('client.situations', $hierarchy) }}
    <table class="table table-striped table-hover">
        <thead>
            <tr>
                <th>Название</th>
                <th>Соответствие</th>
                <th>Решаемые задачи</th>
            </tr>
        </thead>
        <tbody>
            @if(!$situations->isEmpty())
            @foreach($situations as $situation)
                <tr>
                    @if( isset($situation->children) &&
count($situation->children))
                        <td class="td-align_left"><a href="{{
URL::route('problems.list', ['iParentSituationId' =>
$situation->id]) }}><i class="fa fa-level-down"></i> {{ $situation->name
}}</a></td>
                    @else
                        <td class="td-align_left">{{ $situation->name
}}</td>
                    @endif
                    <td>{{ $situation->okved_correspondence
}}</td>
                    <td>
                        @if( isset($situation->activeModels) &&
count($situation->activeModels))
                            <a title="Решаемые задачи" href="{{
URL::route('tasks.list', ['iSituationId' =>
$situation->id]) }}>


```

```

class="fa fa-times-circle"></i></span>
                @endif
            </td>
        </tr>
    @endforeach
    @endif
</tbody>
</table>
@stop
App/views/emails/evaluations/model_broke.blade.php
Здравствуйте!
Порог отсечения модели <a href="{{ URL::route('models.detail',
['iModelId' => $model_id] ) }}>{{ $model_name }}</a> стал
ниже заданного Вами минимального значения.
В настоящее время эта модель недоступна для
использования. Чтобы это исправить необходимо достичь
желаемого уровня порога отсечения.

App/views/errors/exception.blade.php
@extends('layouts.master')
@section('content')
<div class="row">
<div class="alert alert-dismissible alert-danger">
<button type="button" class="close" data-
dismiss="alert">x</button>
<h4>Ой! Возникла исключительная ситуация.</h4>
<p>{{ $message }}</p>
</div>
</div>
@stop
App/views/errors/fatal.blade.php
<!DOCTYPE html
lang="ru">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<title>
    @yield('title')
</title>
{{ HTML::style('assets/bootstrap/css/bootstrap.css') }}
{{ HTML::style('assets/css/font-awesome.min.css') }}
{{ HTML::style('assets/css/main.css') }}

@yield('styles')

</head>
<body>
<div class="container">
<div class="col-lg-10">
<div class="row">
<div class="alert alert-dismissible alert-danger">
<h4>Упс!</h4>
<p>Произошла ошибка при работе приложения</p>
<p>Вы можете зайти на <a class="alert-link" href="{{
URL::to('/') }}>главную страницу</a> или вернуться <a
class="alert-link" href="javascript:history.go(-1)">назад</a> и
попробовать еще раз</p>
<p>Если система не заработала, обратитесь к
администратору <a class="alert-link" href="mailto:{{
\Config::get('app.admin_email') }}">{{ \Config::get('app.admin_email') }}</a>
или зайдите
позже.</p>
</div>
</div>
</div>
</div>
{{ HTML::script('assets/js/jquery-1.11.1.min.js') }}
{{ HTML::script('assets/bootstrap/js/bootstrap.min.js') }}
</body>
</html>
App/views/errors/missing.blade.php
@extends('layouts.master')
@section('content')
<div class="row">
<div class="alert alert-dismissible alert-warning">
<button type="button" class="close" data-
dismiss="alert">x</button>
<h4>Упс!</h4>
<p>Кажется, Вы ищите что-то, чего нет в нашем
приложении</p>
<p>Вы можете зайти на <a class="alert-link" href="{{
URL::to('/') }}>главную страницу</a> или вернуться <a
class="alert-link" href="javascript:history.go(-1)">назад</a> и
попробовать еще раз</p>
</div>

```

```

        </div>
    @stop
App/views/errors/uncaught.blade.php
@extends('layouts.master')
@section('content')
<div class="row">
    <div class="alert alert-dismissible alert-danger">
        <button type="button" class="close" data-dismiss="alert"><x></button>
        <h4>Ошибка!</h4>
        <p>Возникла критическая ошибка в работе системы.  
Вы можете продолжить работу или сообщить администратору  
при повторении</p>
    </div>
</div>
@stop
App/views/layouts/master.blade.php
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>
        @yield('title')
    </title>
    {{ HTML::style('assets/bootstrap/css/bootstrap.css') }}
    {{ HTML::style('assets/css/font-awesome.min.css') }}
    {{ HTML::style('assets/css/main.css') }}
    @yield('styles')
</head>
<body>
<header>

@include('partials.navbar')

</header>
<main>
<div>
    <div class="container">
        <div class="row">
            <div class="col-lg-12">
                <div class="col-lg-5">
                    @if( Session::get('waiting_evaluations') )
                        @include('client.notifications.waiting_evaluations')
                    @endif
                    @if( Session::get('expired_evaluations') )
                        @include('client.notifications.expired_evaluations')
                    @endif
                </div>
                <div class="container">
                    <div class="row">
                        <div class="col-lg-12">
                            @yield('content')
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
</main>
<footer>
{{--@include('partials.footer')--}}
</footer>
{{ HTML::script('assets/js/jquery-1.11.1.min.js') }}
{{ HTML::script('assets/bootstrap/js/bootstrap.min.js') }}
{{ HTML::script('assets/js/search.js') }}
@yield('scripts')
</body>
</html>
App/views/partials/navbar.blade.php
<div>
    <div class="container">
        <div class="navbar navbar-inverse">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-responsive-collapse">
                    <span> </span>
                    <span> </span>
                    <span> </span>
                </button>
                <a class="navbar-brand" href="{{ URL::to('/') }}><i class="fa fa-adjust"></i> dichotomy</a>
            </div>
        </div>
    </div>

```

```

<div class="navbar-collapse collapse navbar-responsive-collapse">
    <ul class="nav navbar-nav">
        <li><a href="{{ URL::to('/') }}>Главная</a></li>
    </ul>
    @if(isset($adminNavBar))
        <ul class="nav navbar-nav">
            @include(Config::get('laravel-menu::views.bootstrap-items'), array('items' => $adminNavBar->roots()))
        </ul>
    @endif
    @if(isset($userNavBar))
        <ul class="nav navbar-nav">
            @include(Config::get('laravel-menu::views.bootstrap-items'), array('items' => $userNavBar->roots()))
        </ul>
    @endif
    <ul class="nav navbar-nav navbar-right">
        @if(isset($authNavBar))
            @include(Config::get('laravel-menu::views.bootstrap-items'), array('items' => $authNavBar->roots()))
        @endif
        @if( \Auth::user() && \Entrust::hasRole('user'))
            {{ Form::open(['route' => 'search', 'class' => 'navbar-form']) }}
            {{ Form::text('search_text', null, ['id' => 'search_text', 'class' => 'form-control col-lg-6', 'placeholder' => 'Поиск', 'rel' => 'tooltip', 'title' => 'Введите сюда код ОКВЭД, название проблемной ситуации или название задачи для решения и нажмите Enter', 'data-toggle' => 'tooltip', 'data-placement' => 'bottom']) }}
            {{ Form::close() }}
        @endif
    </ul>
</div>
</div>
</div>
App/views/hello.blade.php
@extends('layouts.master')
@section('title')
    Главная
@endsection
@section('content')
    Автоматизированная информационная система для бинарной классификации объектов на основе логистической регрессии
@endsection
@stop
App/breadcrumbs.php
<?php
/*
 *          Created      by      PhpStorm.
 *          User:        elluminate
 *          Date:        23.01.15
 *          Time:        13:08
 */
// тут будем хранить механизмы генерации хлебных крошек для разных модулей приложения
// крошки для проблемных ситуаций админу
Breadcrumbs::register('admin.situations', function($oBreadcrumbs, $aHierarchy, $sMode) {
    $oBreadcrumbs->push('Главная', url('/'));
    // первый уровень-заглушка
    $oBreadcrumbs->push('Каталог проблемных ситуаций', route('situations.list'));
    if(is_array($aHierarchy)) {
        foreach($aHierarchy as $section) {
            $oBreadcrumbs->push($section['name'], URL::route('situations.list', array('iParentSituationId' => $section['id'])));
        }
    }
    switch($sMode)
    case 'list':
        break;
    // если показали форму редактирования, то надо добавить заглушку
    case 'edit':
        $oBreadcrumbs->push('Редактирование реквизитов');
        break;
    // аналогично редактированию
    case 'create':
        $oBreadcrumbs->push('Добавление ситуации');
    }
});
// крошки для задач классификации админу
Breadcrumbs::register('admin.models', function($oBreadcrumbs,

```

```

$aHierarchy, $sMode, $iModelId = null) {
    $oBreadcrumbs->push('Главная', url('/'));
    // первый уровень-заглушка
    $oBreadcrumbs->push('Каталог проблемных ситуаций',
route('situations.list'));
    if(is_array($aHierarchy) && count($aHierarchy))
        foreach($aHierarchy as $section) {
            $oBreadcrumbs->push($section['name'],
URL::route('situations.list', array('iParentSituationId' =>
$section['id'])));
        }
    switch($sMode) {
        case 'list':
            break;
        case 'detail':
            if(!is_null($iModelId) && \Model::find($iModelId, ['id']))
                $oBreadcrumbs->push(\Model::find($iModelId, ['name'])-
>name);
            break;
        case 'create':
            $oBreadcrumbs->push('Добавление задачи');
            break;
        case 'edit':
            if(!is_null($iModelId) && \Model::find($iModelId, ['id']))
                $oBreadcrumbs->push(\Model::find($iModelId, ['name'])-
>name, URL::route('models.detail', ['iModelId' => $iModelId]));
            $oBreadcrumbs->push('Редактирование параметров');
            break;
    });
}

// крошки для проблемных ситуаций пользователю
Breadcrumbs::register('client.situations', function($oBreadcrumbs,
$aHierarchy) {
    $oBreadcrumbs->push('Главная', url('/'));
    // первый уровень-заглушка
    $oBreadcrumbs->push('Каталог проблемных ситуаций',
route('problems.list'));
    if(is_array($aHierarchy) && count($aHierarchy))
        foreach($aHierarchy as $section) {
            $oBreadcrumbs->push($section['name'],
URL::route('problems.list', array('iParentSituationId' =>
$section['id'])));
        }
});

// крошки для задач классификации пользователю
Breadcrumbs::register('client.models', function($oBreadcrumbs,
$aHierarchy, $sMode, $iModelId = null) {
    $oBreadcrumbs->push('Главная', url('/'));
    // первый уровень-заглушка
    $oBreadcrumbs->push('Каталог проблемных ситуаций',
route('problems.list'));
    if(is_array($aHierarchy) && count($aHierarchy))
        foreach($aHierarchy as $section) {
            $oBreadcrumbs->push($section['name'],
URL::route('problems.list', array('iParentSituationId' =>
$section['id'])));
        }
    switch($sMode) {
        case 'list':
            break;
        case 'detail':
            if(!is_null($iModelId) && \Model::find($iModelId, ['id']))
                $oBreadcrumbs->push(\Model::find($iModelId, ['name'])-
>name);
            break;
    });
}

Breadcrumbs::register('evaluations', function($oBreadcrumbs,
$sMode, $iModelId = null) {
    $oBreadcrumbs->push('Главная', url('/'));
    $oBreadcrumbs->push('Обратная связь',
route('evaluations.list'));
    switch($sMode) {
        case 'list':
            break;
        case 'detail':
            if(!is_null($iModelId) && \Model::find($iModelId, ['id']))
                $oBreadcrumbs->push(\Model::find($iModelId, ['name'])-
>name);
            break;
    });
}

Breadcrumbs::register('models.inactive', function($oBreadcrumbs)
{

```

```

    $oBreadcrumbs->push('Главная',
    $oBreadcrumbs->push('Неактивные задачи',
route('models.inactive')));
});
```

App/filters.php
<?php

```

/*
----- Application & Route Filters -----
| Below you will find the "before" and "after" events for the application
| which may be used to do any work before or after a request into your
| application. Here you may also register your custom route filters.
| */

```

App::before(function(\$request)
{ //});

App::after(function(\$request, \$response)
{ //});

```

/*
----- Authentication Filters -----
| The following filters are used to verify that the user of the current session is logged into this application. The "basic" filter easily integrates HTTP Basic authentication for quick, simple checking.
| */

```

Route::filter('auth', function()
{
 if (Auth::guest())
 {
 if (Request::ajax())
 return Response::make('Unauthorized', 401);
 else
 return Redirect::guest('login');
 }
});

Route::filter('auth.basic', function()
{
 return Auth::basic();
});

```

/*
----- Guest Filter -----
| The "guest" filter is the counterpart of the authentication filters as it simply checks that the current user is not logged in. A redirect response will be issued if they are, which you may freely change.
| */

```

Route::filter('guest', function()
{
 if (Auth::check())
 return Redirect::to('/');
});

```

/*
----- CSRF Protection Filter -----
| */

```

```
| The CSRF filter is responsible for protecting your application
| against cross-site request forgery attacks. If this special token in a user
| session does not match the one given in this request, we'll bail.
| */

```

```
Route::filter('csrf', function()
{
    if (Session::token() !== Input::get('_token'))
    {
        throw new Illuminate\Session\TokenMismatchException;
    }
});

App::missing(function($exception)
{
    return Response::view('errors.missing', array(), 404);
});
```

```
// закрываем все маршруты после admin, если нет роли
администратора, переадресовываем на авторизацию
\Entrust::routeNeedsRole('admin*', 'administrator',
Redirect::to('users/login'));
```

```
// закрываем все маршруты после admin, если нет роли
администратора, переадресовываем на авторизацию
\Entrust::routeNeedsRole('client*', 'user', Redirect::to('users/login'));
```

```
App/routes.php
<?php

/*
----- Application Routes -----

```

```
Here is where you can register all of the routes for an application.
It's a breeze. Simply tell Laravel the URLs it should respond to
and give it the Closure to execute when that URI is requested.
*/
```

```
Route::get('/', function()
{
    return View::make('hello');
});

Route::group(array('prefix' => 'admin'), function()
{
    // Маршруты проблемных ситуаций
    Route::get('situations/list/{iParentSituationId?}', array('as' => 'situations.list', 'uses' => 'AdminSituationController@index'));
    Route::get('situations/create/{iParentSituationId?}', array('as' => 'situations.create', 'uses' => 'AdminSituationController@create'));
    Route::post('situations/store', array('before' => 'csrf', 'as' => 'situations.store', 'uses' => 'AdminSituationController@store'));
    Route::get('situations/edit/{iSituationId}', array('as' => 'situations.edit', 'uses' => 'AdminSituationController@edit'));
    Route::post('situations/update', array('before' => 'csrf', 'as' => 'situations.update', 'uses' => 'AdminSituationController@update'));
    Route::get('situations/destroy/{iSituationId}', array('as' => 'situations.destroy', 'uses' => 'AdminSituationController@destroy'));

    // Маршруты моделей
    Route::get('models/list/{iSituationId}', array('as' => 'models.list', 'uses' => 'AdminModelController@index'));
    Route::get('models/detail/{iModelId}', array('as' => 'models.detail', 'uses' => 'AdminModelController@show'));
    Route::get('models/create/{iSituationId}', array('as' => 'models.create', 'uses' => 'AdminModelController@create'));
    Route::post('models/store', array('before' => 'csrf', 'as' => 'models.store', 'uses' => 'AdminModelController@store'));
    Route::get('models/edit/{iModelId}', array('as' => 'models.edit', 'uses' => 'AdminModelController@edit'));
    Route::post('models/update', array('before' => 'csrf', 'as' => 'models.update', 'uses' => 'AdminModelController@update'));
    Route::get('models/destroy/{iModelId}', array('as' => 'models.destroy', 'uses' => 'AdminModelController@destroy'));
    Route::get('models/template/', array('as' => 'models.template', 'uses' => ''));
```

```
'AdminModelController@downloadTemplate'));
Route::get('models/inactive/', array('as' => 'models.inactive', 'uses' => 'AdminModelController@inactiveModels'));
Route::get('models/dump/{iModelId}', array('as' => 'models.dump', 'uses' => 'AdminModelController@dump'));
```

```
Route::group(array('prefix' => 'client'), function()
{
    Route::get('problems/list/{iParentSituationId?}', array('as' => 'problems.list', 'uses' => 'ClientSituationController@index'));
    Route::get('tasks/list/{iSituationId}', array('as' => 'tasks.list', 'uses' => 'ClientModelController@index'));
    Route::get('tasks/detail/{iModelId}', array('before' => 'expiredTasks', 'as' => 'tasks.detail', 'uses' => 'ClientModelController@showModelForm'));
    Route::post('tasks/compute', array('before' => 'csrf', 'as' => 'tasks.compute', 'uses' => 'ClientModelController@compute'));
    Route::post('/search/', array('before' => 'csrf', 'as' => 'search', 'uses' => 'SearchController@index'));
    Route::get('evaluations/list', array('as' => 'evaluations.list', 'uses' => 'EvaluationController@index'));
    Route::get('evaluations/detail/{iEvaluationId}', array('as' => 'evaluations.detail', 'uses' => 'EvaluationController@show'));
    Route::post('evaluations/confirm', array('before' => 'csrf', 'as' => 'evaluations.confirm', 'uses' => 'EvaluationController@confirm'));
    Route::get('test', array('as' => 'test.ajax', 'uses' => 'HomeController@test'));
```

```
/*
----- Confide routes -----
Route::get('users/create', 'UsersController@create');
Route::post('users', 'UsersController@store');
Route::get('users/login', 'UsersController@login');
Route::post('users/login', 'UsersController@login');
Route::get('users/confirm/{code}', 'UsersController@confirm');
Route::get('users/forgot_password', 'UsersController@forgotPassword');
Route::post('users/forgot_password', 'UsersController@doForgotPassword');
Route::get('users/reset_password/{token}', 'UsersController@resetPassword');
Route::post('users/reset_password', 'UsersController@doResetPassword');
Route::get('users/logout', 'UsersController@logout');
```

```
// пришлось вынести меню сюда, иначе не работают
именованные маршруты
// меню авторизации
Menu::make('authNavBar', function($menu)
{
    // если пользователь залогинился, то покажем кто он
    и кнопку на выход
    if(Auth::user())
    {
        $menu->add('<i class="fa fa-user"></i> Вы
    вошли как '.Auth::user()->username);
        $menu->add('<i class="fa fa-sign-out"></i>
    Выйти', 'users/logout');
    }
    // иначе предложим войти или зарегистрироваться
    else
    {
        $menu->add('<i class="fa fa-sign-in"></i>
    Войти', 'users/login');
        $menu->add('<i class="fa fa-user-plus"></i>
    Зарегистрироваться', 'users/create');
    });
});
```

```
// административное меню
Menu::make('adminNavBar', function($menu)
{
    // проверим, есть ли у пользователя роль
    администратора
    if(\Entrust::hasRole('administrator'))
    {
        // назначим ему id, чтобы потом добавлять
        // внутрь его, текстовые id почему-то не поддерживаются,
        // придется так
        $menu->add('<i class="fa fa-cogs"></i>
    Управление', 'id' => 1);
        $menu->find(1)->add('<i class="fa fa-folder-open"></i> Каталог проблемных ситуаций<a>', ['route' => 'situations.list', 'id' => 2]);
        $menu->find(1)->add('<i class="fa fa-lock"></i> Неактивные задачи<a>', ['route' => 'models.inactive', 'id' => 3]);
        if(\Request::is('/admin*'))
            $menu->find(1)->active();
        if(\Route::is('situations.*'))
            $menu->find(2)->active();
    }
});
```

```

        if(\Route::is('models.inactive'))
            $menu->find(3)->active();
    });

// меню зарегистрированного пользователя
Menu::make('userNavBar', function($menu) {
    // проверим, есть ли у пользователя роли юзера
    if(\Entrust::hasRole('user'))
        $menu->add('<i class="fa fa-check"></i>
Решение задач классификации', ['id' => 1]);
    $bExpired = false;
    if(\!Entrust::hasRole('administrator'))
        $bExpired =
EvaluationRepository::checkExpiredEvaluations();
    }
    if(!$bExpired)
        $menu->find(1)->add('<i class="fa fa-magic"></i> Поиск и решение задачи', ['route' => 'problems.list', 'id' => 2]);
        $menu->find(1)->add('<i class="fa fa-send-o"></i> Подтверждение решения (обратная связь)', ['route' => 'evaluations.list', 'id' => 3]);
        if(Request::is('/client*'))
            $menu->find(1)->active();
        EvaluationRepository::getNotifications();
    }
});
public/assets/css/main.css
.table td {
    text-align: center;
}
.table th {
    text-align: center;
}
#threshold_slider .slider-selection {
    background: #BABABA;
}
.table .td-align_left {
    text-align: left;
}
.page-header margin: 0 0 21px;
.menu-raw display: block !important;
}

```

```

Public/assets/js/evaluation.js
/**
 * Created by elluminate on 24.02.15.
 */
$( document ).ready(function() {
    $("#eval_formCollapse").click(function() {
        var className = $("#eval_form").attr('class');
        if(className == 'collapse') {
            $("#eval_form").removeClass();
            $("#eval_formCollapse").text('Скрыть значения параметров');
        } else {
            $("#eval_form").addClass('collapse');
            $("#eval_formCollapse").text('Показать значения параметров');
        }
    });
});
Public/assets/js/list.js
/**
 * Created by elluminate on 23.01.15.
 */
$( document ).ready(function(e) {
    $('#delModal').on('show.bs.modal', function(e) {
        $(this).find('.danger').attr('href', e.relatedTarget.data('href'));
    });
});
App/assets/js/search.js
/**
 * Created by elluminate on 18.02.15.
 */
$(function () {
    $('[data-toggle="tooltip"]').tooltip();
});
App/assets/js/slider.js
/**
 * Created by elluminate on 31.01.15.
 */
$( document ).ready(function() {
    $(".js_range_slider").slider({
        tooltip: 'always'
    });
});

```