ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК
КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ РЕАЛИЗАЦИИ АЛГОРИТМОВ ВЫЯВЛЕНИЯ ПРИЧИННО-СЛЕДСТВЕННЫХ ОТНОШЕНИЙ В ЭКСПЕРИМЕНТАЛЬНЫХ ДАННЫХ

Выпускная квалификационная работа

студента очной формы обучения

обучающегося по направлению подготовки 02.03.03

Математическое обеспечение и администрирование информационных систем

4 курса группы 07001302

Мирошниченко Андрея Сергеевича

Научный руководитель
к.т.н. доц. Михелев В.М.

БЕЛГОРОД 2017

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время объем данных, в частности информационных становится все больше и больше, что создает проблему обработки, анализа и представления этих данных. Анализ данных является важной задачей почти в любой сфере деятельности человека, в том числе в научных исследованиях. Таким образом основная задача анализа данных – это выбор самой важной и ключевой информации. Отбор важной информации определяется некоторым набором правил, условий или соответствий. При анализе человеком тысяч и даже миллионов строк данных есть такой человеческий фактор, когда человек может допустить ошибку в следствии усталости или невнимательности. Однако вычислительная техника, такая как компьютер лишен этого фактора, что позволяет избежать таких ошибок. В настоящее время повышается популярность и актуальность таких понятий как: машинное обучение, искусственный интеллект, нейронные сети. Эти технологии повышают эффективность обработки и анализа колоссальных объемов данных.

Актуальность данной выпускной квалификационной работы заключается в том, что представленный метод выявления причинно-следственных отношений в экспериментальных данных, позволяет провести анализ большого набора данных и выявляет самую важную информацию, затем идет процесс обработки полученных данных после чего результат представляется в виде графа, так как по данных некоторых исследователей человек легче всего воспринимает графическую информацию нежели обычный текст.

Главной целью данной дипломной работы является разработать математическое и программное обеспечение, позволяющее выявить причинно-следственные отношения в наборе экспериментальных данных большого объема.

Данная цель дипломной работы ставит следующие задачи:

- разработать программу для анализа экспериментальных данных;
- протестировать программу на известном наборе данных;
- исследовать экспериментальные данные;
- провести анализ полученных результатов.

При анализе данных часто требуется объяснить зависимость или независимость определенных свойств, признаков у группы исследуемых. Например, необходимо объяснить зависимость уровня подготовки спортсмена к соревнованиям от его личностных характеристик таких как: соперничество и агрессивность. В данном примере под фактом подразумевается уровень соперничества или агрессивности. С помощью которого описывается группа спортсменов. Следовательно, возникает основная задача описать каузальные отношения между значениями признаков или самими признаками. Важно отметить, что в данном примере прослеживается корреляция между признаками, как прямая зависимость, но это не значит, что изменение одной из характеристик с изменением другой будет давать точный ответ что является причиной, а что следствием. Точно также можно сказать о факторном анализе, в процессе выявления линейных комбинаций признаков, будут являться лишь факторами, по которым разделяются признаки. Проблемой интерпретации результатов статистического анализа является, что результаты можно использовать только для уровня достоверности уже выдвинутых гипотез о каузальной зависимости.

Отдельно следует отметить что имеет место случай, когда признак или совокупность признаков зависит от цело ряда признаков. Выделить такие зависимости с использованием только корреляции достаточно трудная задача.

Также кроме статистического анализа данных в машинном обучении часто применяется интеллектуальный метод анализа данных, в частности индуктивные методы, в частности ДСМ-метод. Данный метод является автоматическим методом порождения гипотез, формализующий схему правдоподобного и достоверного вывода, который называется ДСМ-

рассуждение. ДСМ-рассуждение является синтезом познавательных процедур: индукции, аналогии и абдукции. ДСМ-метод был создан как средство автоматизированного построения формализации знаний о предметной области средствами называемых квазиаксиоматических теорий.

Важное место в анализе данных имеет представление полученных результатов. Для более наглядного понимания результатов в данной дипломной работе представлен метод генерирующий граф, который отображает важные свойства признаков. На вход графа поступают гипотезы в виде бинарных массивов, которые генерируются в процессе работы обучения и анализа данных. В качестве вершин выступают свойства признаков, а дуги (ребра) графа отображают связь (зависимость) свойств признаков.

Данная дипломная работа состоит из 3 глав, посвященных обзору и анализу предметной области, проектной части и тестированию программы, которые раскрывают основной смысл дипломной работы. Дипломная работа состоит из 16 рисунков, 3 таблиц, 12 литературных источников и 2 приложений.

1. ОБЗОР И АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Исследуемые данные

При проведении исследования данных чаще всего встречаются статистические и экспериментальные данные, которые получают в результате проведения тестирования.

В данной выпускной квалификационной работе были использованы данные психологического тестирования предоставленные Институтом системного анализа РАН. Часто в исследованиях используют случайные данные, в следствии чего при использовании выходного продукта (программное обеспечение или устройство) на реально поставленной задаче может нести большую погрешность. Так как в исследовании участвовала группа испытуемых, которая дала согласие на обработку данных тестирования, это подтверждает реальность экспериментальных данных, что позволяет избежать погрешности между рандомизированными и реальными данными.

Психологическое тестирование – исследование определенных психологических свойств и качеств человека (личности) с использованием направленных тестов. Главные характеристики психологических тестов – это валидность, надежность, репрезентативность и достоверность. Валидность – это соответствие результатов теста той характеристике, для измерения которой он предназначен. Надёжность – свойство теста давать при повторном измерении близкие результаты. Надёжность как внутренняя согласованность – направленность всех элементов тестовой шкалы на измерение одного качества. Репрезентативность – соответствие между нормами (интервалами на тестовой шкале), полученными на выборке, и нормами, которые могут быть получены на популяции. Достоверность – свойство теста противодействовать

фальсификации – намеренному или бессознательному искажению результатов испытуемыми [1].

1.2. Анализ данных

1.2.1. Понятие анализа данных

Анализ данных — область математики и информатики, занимающаяся построением и исследованием наиболее общих математических методов и вычислительных алгоритмов извлечения знаний из экспериментальных (в широком смысле) данных; процесс исследования, фильтрации, преобразования и моделирования данных с целью извлечения полезной информации и принятия решений [2]. Одним из видов анализа является статистических анализ данных.

Так как в дипломной работе основной целью является разработка программного обеспечения для выявления причинно-следственных отношений, основной задачей является программа, позволяющая производить анализ данных по определенной схеме.

Ниже на рис. 1.1. представлена общая схема анализа данных. Данная схема демонстрирует основные этапы анализа. Представленное ниже описание этапов позволяет понять основные шаги и действия необходимые для анализа данных.

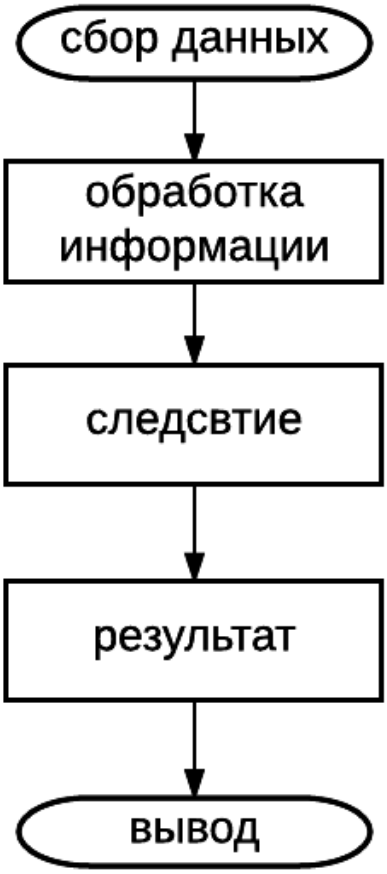

Рис. 1.1. Общая схема анализа данных

Любой анализ данных начинается с этапа сбора данных, т.е. необходимо провести физический эксперимент получения данных. Процесс сбора данных в определенной степени зависит от цели всего исследования или эксперимента. Поэтому важно уже на начальном этапе учитывать следующие шаги анализа такие как обработка и следствие, для того чтобы неким образом подготовить или классифицировать данные, для более удобной и продуктивной обработки.

Следующим этап после сбора данных идет обработка информации. Именно информации, т.к. любые данные несут в себе информацию. На данном этапе производится одна из главных частей всего анализа, которая позволяет представить входные данные в необходимую форму. Ниже представлены примеры подготовки (обработки информации) данных для следующего этапа.

- Преобразование качественной характеристики в числовую и наоборот.

- Классификация данных по определенным группам, классам.
- Подбор системы оценивания, измерения информации.

Этап обработки информации позволяет определить структуру анализа данных.

Следующий этап – это следствие, который в свою очередь является основной частью анализа. Под следствием подразумевается определение и использование методов анализа информации. Другими словами, следствие представляет собой некое решение поставленной задачи при анализе. Например, требуется провести анализ средней учебной успеваемости студентов за прошедший год. Рассмотрим подробнее этап следствия.

- После сбора данных и обработки полученной информации мы имеем: кол-во учащихся, оценочный критерий (балл за успеваемость).
- Определяем метод решения поставленной задачи, для приведенного примера решением является подсчет среднего арифметического.
- Производим подсчет среднего арифметического, после чего полученный ответ является следствием решения поставленной задачи.

После того как мы имеем решение на поставленную задачу необходимо провести следующий этап анализа – результат. На этапе результата выдвигаются гипотезы, правила, условия, которые дают определенную характеристику анализа данных, т.е. это может быть сравнение с эталонным результатом, сопоставления данных по определенным оценочным критериям и т.д.

Для примера, представленного выше результат может дать ответ на такие вопросы как: успеваемость относительно других вузов, уровень подготовки студентов и т.д.

Вывод – это заключительный этап анализа. Данных этап завершает анализ и дает окончательный ответ на поставленную задачу, т.к. часто возникает потребность повторной обработки информации, после этапа результата.

1.2.2. Основные методы статистического анализа данных

Статистика – это отрасль знаний, наука излагающая общие вопросы сбора, измерения и анализа массовых статистических данных (количественных или качественных) [3]. Часто понятие статистики ассоциируют с областью математики, где используются сложные формулы и вычисления. Однако, статистика встречается и в повседневной жизни человека, будь то планирование бюджета, расчет времени, оценка затраты физической силы или предсказывание погоды по текущим показателям. Постоянный отбор, классификация и упорядочивание информации, связывание её с другими данными и показателями, которые получаем в прочих исследованиях, позволяет сделать определенный вывод для принятия правильного решения.

Статистические данные – это совокупность объектов (наблюдений, случаев) и признаков (переменных), их характеризующих. Например, объекты исследования – страны мира и признаки, – географические и экономические показатели их характеризующие: континент; высота местности над уровнем моря; среднегодовая температура; место страны в списке по качеству жизни, доли ВВП на душу населения; расходы общества на здравоохранение, образование, армию; средняя продолжительность жизни; доля безработицы, безграмотных; индекс качества жизни и т.д. [4].

В роли переменных выступают величины, принимающие различные значения в результате измерения. Независимые переменные – это переменные, значения которых можно изменить в ходе эксперимента. Зависимые переменные – это переменные у которых значения остаются неизменными и их значения можно только измерить. Переменные могут измеряться в различных системах счета, шкалах. При анализе используют следующие виды шкал: номинальная, порядковая, интервальная, шкала отношений, абсолютная. Они отличаются между собой количеством математических операций (действий), которые можно к ним применить.

В номинальной шкале не определена ни одна арифметическая операция. При измерении по этой шкале возможно классифицировать объекты.

Порядковая шкала позволяет определить кроме класса, упорядочивание наблюдений, сравнивая их между собой, но данная шкала не определяет расстояние между классами. Часто порядковые переменные называют группирующими переменными или категориальными переменными.

Интервальная шкала позволяет произвести расчет расстояния между наблюдениями, другими словами, как и порядковая шкала, но с учетом расстояния. С переменными интервальной шкалы можно производить арифметические операции.

Шкала отношений схожа с интервальной шкалой, также шкала имеет фиксированную точку отсчета, но произвольный масштаб. С переменными шкалы отношений также можно производить арифметические операции.

Абсолютная шкала похожа на числовую прямую, которая имеет и абсолютный нуль, и масштаб.

Ниже представлены основные методы статистического анализа.

- Корреляционный.
- Регрессионный.
- Канонический.
- Методы сравнения средних.
- Частотный анализ.
- Кросстабулиция (сопряжение).
- Анализ соответствий.
- Кластерный.
- Дискриминантный.
- Факторный.
- Деревья классификации.
- Анализ главных компонент и классификация.
- Многомерное шкалирование.

- Моделирование структурными уравнениями (причинное моделирование).
- Временные ряды.
- Нейронные сети.
- Планирование экспериментов.
- Карты контроля качества.

Рассмотрим подробнее каждый метод.

Корреляционный анализ. Между несколькими переменными или случайными величинами может существовать функциональная зависимость, которая проявляется так, что одна из величин (переменных) определяется как функция от другой. Но между величинами может быть и другая зависимость, которая проявляется в том, что одна из величин может влиять на изменение другой изменением своего закона распространения. Данный вид зависимости называют: стохастической. Стохастическая зависимость появляется тогда, когда существуют общие случайные факторы, которые влияют на обе переменные. Для определения меры зависимости между этими переменными используют коэффициент корреляции, пределы которого меняются от -1 до +1. При отрицательном коэффициенте корреляции, увеличение значения одной переменной, повлияет на убывание значения другой переменной. В случае, когда переменные независимы коэффициент корреляции равен 0. Существуют корреляции: Пирсона, Спирмена, Кендала, Гамма.

Регрессионный анализ. В данном виде анализа строится зависимость одной случайной величины (переменной) от одной или нескольких других случайных величин. Учитывая вышесказанное первая, величина является зависимой, а другие – независимые. Независимые величины называются регрессорами (факторами), а зависимая величина – результативным признаком. Если число предикатов больше одного, то регрессию называют множественной, иначе регрессия будет являться простой. Общий случай регрессия выглядит так:

$$y=f(x_1, x_2, \ldots, x_n), \quad (1.1)$$

где y-зависимая величина (переменная), x_i $(i=1, \ldots,n)$ – предикторы и n число предикторов.

Канонический анализ используется для анализа зависимостей между двумя списками признаков (независимых величин), которые характеризуют объекты. Данный вид анализа является обобщением множественной корреляции.

Методы сравнения средних. В исследованиях бывают случаи, когда средний результат некоторого признака одной серии испытаний (экспериментов) отличается от среднего другой серии. При этом сравнение средних результатов – это один из способов обнаружения связей между переменными признаками, которые характеризуют исследуемую совокупность наблюдений. Самым общим методом сравнения средних – это дисперсионный анализ.

Частотный анализ. Частотные таблицы, являются простейшим методом анализа категориальных величин. Этот вид статистического анализа используют как одну из процедур разведочного анализа, для определения как распределены различные множества наблюдений в выборке, также для определения распределения величин от максимального до минимального порога. Часто данный метод иллюстрируется в виде гистограмм.

Кросстабуляция или сопряжение – это процесс слияния двух или более частотных таблиц так, что каждая ячейка в сформированной таблице является единственной комбинацией значений или уровней объединенных переменных. Сопряжение позволяет совместить частоты появления наблюдения на разных уровнях. При исследовании этих частот, появляется возможность обнаружить зависимости между объединенными переменными и просмотреть структуру этой зависимости. Как правило объединяются категориальные или количественные величины (переменные) с относительно небольшим числом значений.

Анализ соответствий. Данный метод содержит более мощные описательные и разведочные методы, по сравнению с частотным анализом. Анализ соответствий, похож на таблицы сопряженности, но при данном анализе частоты в таблицах сопряженности нормируются так, чтобы суммы элементов всей таблицы была равна. Главной целью этого метода является представления содержимого таблицы, как расстояние между отдельными столбцами и строками пространстве меньшей размерности.

Кластерный анализ – это вид классификационного анализа и его главное назначение – разделение множества исследуемых признаков и объектов на однородные группы (кластеры). Данный метод является многомерным статистическим, таким образом здесь учитывается большой объем объектов исследования и признаков, которые характеризируют эти объекты. Главным отличительным признаком кластерного анализа является разделение объектов не по одному критерию (признаку), а по группе критериев. Также данный метод не зависит от вида рассматриваемых объектов. Основная задача кластерного анализа – это разбиение признаков объектов на множество кластеров, таким образом, чтобы каждый объект принадлежал только одному классу (группе) разделения.

Дискриминантный анализ. Данный метод состоит из статистических методов классификации многомерных наблюдений тогда, когда тот, кто исследует имеет обучающую выборку. Дискриминантный анализ является многомерным, поскольку использует множество признаков объекта. Основная идея метода – это при исследовании различных признаков объекта распределить его к классу, что означает отнести его к одной группе классов из целого множества самым оптимальным образом. В данном методе учитывается, что исходные данные с признаками объектов содержат переменную, которая демонстрирует принадлежность определенному классу. Таким образом, в дискриминантном анализе присутствует такая проверка, чтобы отсутствовало противоречие классификации.

Факторный анализ является одним из самых востребованных многомерных статистических методов. Если два вышеописанным метода распределяют исследования, разделяя их на классы однородности, то данный метод распределяет величины (признаки или переменные), которые описывают наблюдения. Таким образом основная цель данного метода – это сокращение числа признаков на основе распределения (классификации) переменных и определения структуры зависимости между переменными. Сокращение производится путем добавления скрытых общих факторов, которые объясняют зависимости между исследуемыми признаками объекта.

Деревья классификации. Данный метод позволяет экстраполировать принадлежность объектов к той или иной группе в зависимости от соответствующих показателей признаков, которые характеризуют объекты. Признаки являются независимыми переменными, а величина показывающая принадлежность объектов к классам, называется зависимой. Между дискриминантным анализом и деревьями классификации есть разница, которая заключается в том, что деревья могут выполнять одномерное ветвление по величинам различных типов классифицирующими, порядковыми, интервальными. На правило распределения количественных величин (переменных) не накладывается никаких ограничений. Структура метода построена так, что существует возможность по изменяемым параметрам создавать деревья нефиксированной глубины и ширины, стараясь при этом избежать больших ошибок классификации. Однако, при сложной структуре дерева затруднительно производить классификацию нового объекта, т.к. большая совокупность решающих правил.

Классификация и анализ главных компонент. Часто при исследовании возникает потребность анализа данных большой размерности. Данный метод позволяет решить поставленную задачу (потребность анализа больших данных), а также этот метод позволяет достичь основные цели:

- редукция данных (значительное уменьшение числа переменных), для получения главных и не коррелирующих переменных;

- распределение переменных и наблюдений, с использованием создаваемого факторного пространства.

Многомерное шкалирование. Этот метод рассматривается как альтернатива факторного анализа, где достигается сокращение значительного числа переменных, когда выделяются латентные факторы, в свою очередь объясняющие связи между наблюдаемыми величинами. Поиск и интерпретация латентных переменных, которые дают возможность объяснить сходства между исследуемыми объектами является основной целью многомерного шкалирования. В данном методе исходные данные представляют собой произвольный тип матрицы сходства объектов. Главная идея метода состоит в том, что существует метрическое пространство существенных базовых характеристик, послужившие основой для получения эмпирических данных о близости между парами объектов.

Моделирование структурными уравнениями или причинное моделирование. Прогресс в разделе многомерного статистического анализа и анализа корреляционных структур, который объединен с самыми новыми вычислительными алгоритмами, стал началом моделирования структурными уравнениями известными как – Sepath. Данный вид многомерного анализа включает в себя методы из разных областей статистики. В данном методе факторный анализ и множественная регрессия стали активно развиваться и объединяться. Главным объектом моделирования структурными уравнениями являются сложные системы, с неизвестной внутренней структурой, так называемый «черный ящик». При исследовании параметров системы используя Sepath, можно просмотреть ее структуру и установить причинно-следственные зависимости между элементами системы.

Временные ряды – это самое быстро развивающееся, направление в области математической статистики. Временной ряд или динамический ряд, состоит из последовательности наблюдений некоторого признака в последовательные моменты, которые имеют один промежуток времени. В данном методе уровень ряда – это отдельное наблюдение. Исследование и

анализ динамического ряда позволяют построить модели для экстраполирования значений определенного признака, при условии, что известны последовательности наблюдений прошлых уровней.

Нейронные сети. В настоящее время нейронные сети приобретают все большую актуальность в информационной сфере, а именно для решения задач классификации, предсказывания и распознавания. Например, имеется изображение в виде матрицы размером NxN, где каждый элемент матрицы может принимать значение «1» или «0». «1» – когда ячейка закрашена и «0» – не закрашена. В результате заполнения матрицы мы получим некое изображение или образ изображения. После чего данное изображение или образ используется для сравнения или классификации. Для того что бы сравнить изображение или образ недостаточно сравнивать элементы матрицы, т.к. это приведет к большой избыточности данных и не даст конструктивного результата. Следовательно, грамотным решением подобной задачи, будет использование нейронной сети, которая позволит обойти эти минусы [5].

Планирование экспериментов. Данный метод позволяет располагать исследования в заданном порядке или проводить определенно спланированные проверки с целью полного использования возможностей этих методов. В наше время экспериментальные методы широко используются в различных областях, как науки, так и практической деятельности. Главной целью данных методов является исследования влияния определенных факторов на рассматриваемый процесс, а также поиск оптимальных уровней факторов, которые определяют уровень изменения этого процесса.

Карты контроля качества. В настоящее время существует проблема качества выпускаемой продукции и услуг, которые оказывают обществу. Как показывает практика качество продукции и услуг, формируется из исследований и разработок ряда организаций, отсюда следует, что разработка методов контроля качества, позволяющие выявить слабые стороны производства продукции или оказания услуг, является актуальным.

Рассмотренные выше статистические методы анализа, предоставляют большой спектр выбора метода или совокупности методов для достижения цели выпускной квалификационной работы.

1.2.3. Машинное обучение

В настоящее время мы часто сталкиваемся с потребностью выявить или обнаружить определенные закономерности больших объемов данных. Например, распознавание и классификация текстов, прогнозирование погоды или динамики финансов. При исследовании и анализе подобного рода данных обнаружить закономерности или зависимости «вручную» бывает нереально, и такая ситуация встречается все чаще и чаще. Решением подобного рода проблем является использование технологий машинного обучения.

Машинное обучение – это широкий раздел искусственного интеллекта, который изучает методы построения алгоритмов, способные обучаться. Существует два типа обучения: индуктивное и дедуктивное обучение.

Индуктивное обучение – это обучение на выявлении основных закономерностей по частным эмпирическим данным.

Дедуктивное обучение – это обучение предполагающее формализацию знаний экспертов и перенос их в вычислительную технику (компьютер) в виде базы знаний.

Рассмотрим основные стандартные типы задач машинного обучения. Для более наглядной классификации задач, было принято решение отобразить их в виде схемы. Схема основных стандартных типов задач представлена ниже на рис 1.2.

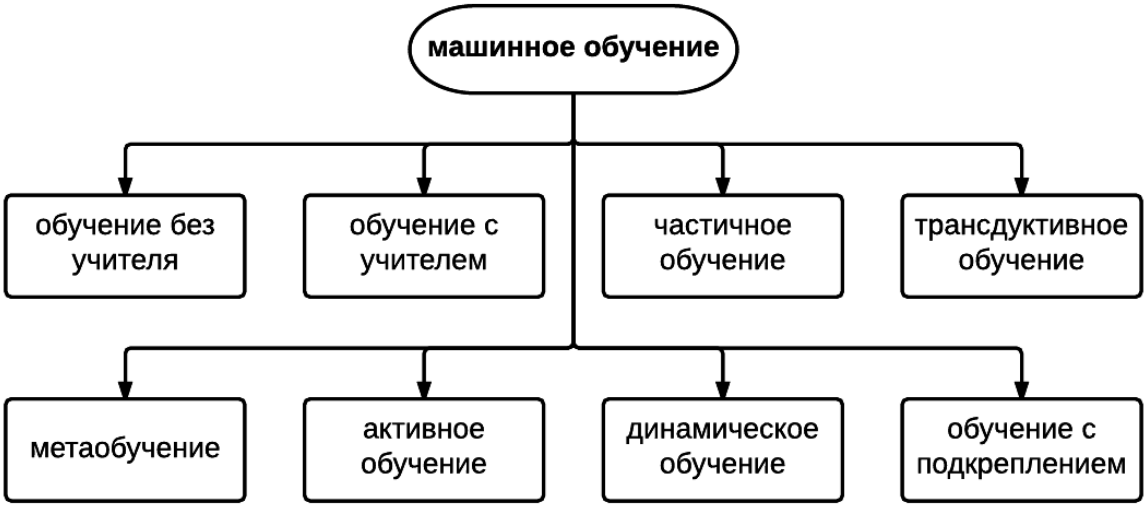

Рис. 1.2. Основные стандартные типы задач

Обучение с учителем – самый распространенный тип. Каждый прецедент представляет собой пару «объект, ответ», где необходимо найти функциональную связь (зависимость) ответов от описаний объектов и разработать алгоритм, который, принимает на вход описание объекта, а на выход выдает ответ. Функционал качества, как правило определяется средней ошибкой ответов, полученных алгоритмом, по всем объектам этой выборки.

К обучению с учителем можно отнести ряд задач:

- задачи классификации;
- задачи регрессии;
- задачи ранжирования;
- задачи прогнозирования.

В задачах классификации множество возможных (допустимых) ответов конечно. Множество – это метки классов, а класс – это множество всех объектов с данным значением множества (метки).

В задачах регрессии допустимыми ответами являются действительные числа или числовые вектора.

В задачах ранжирования ответы необходимо получать сразу на множестве объектов, затем произвести их сортировку по значениям ответов. Данную задачу можно свести к задачам классификации или регрессии.

В задачах прогнозирования в роли объектов выступают отрезки временных рядов (динамических рядов), разрывающиеся только когда необходимо произвести прогноз на будущее.

Обучение без учителя – это тип в котором ответы не задаются, и необходимо искать связи (зависимости) между объектами.

К обучению без учителя можно отнести ряд задач:

- задача кластеризации;
- задача поиска ассоциативных правил;
- задача фильтрации выбросов;
- задача построения доверительной области;
- задача сокращения размерности;
- задача заполнения пропущенных значений.

В задаче кластеризации суть в том, чтобы сгруппировать объекты в кластеры, при этом используя данные о попарном сходстве объектов.

В задачах поиска ассоциативных правил, исходные данные имеют вид признаковых описаний. Необходимо найти такие наборы признаков и их значения, чтобы они встречались особенно часто в признаковых описаниях объектов, при этом неслучайно.

Задача фильтрации выбросов заключается в идентификации в обучающей выборке малого числа нетипичных объектов.

Задачи построения доверительной области – это построения области минимального объема с гладкой границей, которая содержит заданную часть выборки.

Задача сокращения размерности похожа на свертку, т.е. преобразовать данные таким образом, чтобы получить новый набор данных, при этом не утратить существенную информацию об объектах выборки.

Задачи заполнения пропущенных значений сводятся к подстановке недостающих элементов в матрице объекты-признаки их экстраполированными значениями.

Следующий рассматриваемый тип – это частичное обучение, которые находится между обучением с учителем и без учителя. Каждый прецедент рассматривается как пара «объект, ответ», но при этом ответы известны только на определенную часть прецедентов.

Трансдуктивное обучение – это тип где дана конечная обучающая выборка прецедентов, и необходимо по этим частным данным произвести экстраполяцию относительно других данных тестовой выборки.

При обучении с подкреплением роль объектов играют пары «ситуация, принятое решение», а ответами будут являться значения функционала качества, которого характеризует правильность принятых решений.

Следующий тип – это динамическое обучение, при данном типе обучение возможно с учителем и без него. Смысл в том, что прецеденты идут потоком, и появляется необходимость принимать решение незамедлительно по каждому прецеденту и сразу доучивать модель относительно новых прецедентов.

Активное обучение – это тип, где обучаемый имеет возможность самостоятельно без учителя назначать следующий прецедент, который потом станет известным.

Метаобучение – это тип машинного обучения, где в роли прецедентов выступают ранее решенные задачи обучения. Необходимо определить, какие из разработанных решений (эвристик) работают эффективнее других. Итоговая цель – обеспечение постоянного совершенствования алгоритма с течением времени.

1.2.4. ДСМ метод

Также кроме статистического анализа данных в машинном обучении часто применяется интеллектуальный метод анализа данных, в частности индуктивные методы, в частности ДСМ-метод. ДСМ-метод – это метод автоматического порождения гитез, который формализует схему

правдоподобного и достоверного вывода, называемую ДСМ-рассуждением [7].

ДСМ-рассуждение – это синтез познавательных процедур таких как: индукция, аналогия и абдукция. Данный метод был разработан как механизм автоматизированного построения формализации знаний об определенной предметной области средствами, которые называют: квазиаксиоматические теории.

ДСМ-метод управляет (оперирует) сущностями трех видов:

- Объекты предметной области
- Свойства этих объектов
- Возможное причины свойств

Предполагается, что все объекты имеют свою модель (структуру), где причинами свойств этих объектов являются фрагменты данной модели.

На вход ДСМ-метод получает некоторое множество изучаемых объектов и сведения об их структуре, о наличии или отсутствии у них определенных свойств, а также, в некоторых случаях, о связи между структурой объектов и их свойств. Кроме того, имеется ряд целевых признаков, каждый из которых разбивает исходное множество объектов на четыре непересекающихся подмножества:

- объекты, про которые известно, что они обладают данным признаком,
- объекты, про которые известно, что они не обладают данным признаком,
- объекты, для которых существуют аргументы как за, так и против того, что они обладают данным признаком,
- объекты, о которых неизвестно, обладают они этим признаком или нет.

Результатом применения ДСМ-метода являются гипотезы двух типов:

- гипотезы о связи определенных структурных фрагментов изучаемых объектов со свойствами, которыми они обладают,

- гипотезы о наличии или отсутствии целевых признаков у объектов, для которых изначально это было неизвестно, формируемые на основании установленной взаимосвязи между свойствами объектов и их структурными компонентами [8].

Ниже описан один шаг ДСМ-метода в самом простом варианте [9].

- O — множество объектов;
- P — множество свойств объектов или целевых признаков;
- C — множество всевозможных причин свойств объектов (элементы структуры объектов);
- V — множество оценок. $V = \{+1, -1, 0, T\}$.

Пусть есть функция $P: O \rightarrow 2C$, сопоставляющая каждому объекту о подмножество фрагментов (элементов структуры), которые встречаются в объекте о.

Введём новую функцию $F: O \times P \rightarrow V$, которая представляет начальную ситуацию.

- $F(o, p) = +1$ — известно, что объект о обладает свойством p;
- $F(o, p) = -1$ — известно, что объект о не обладает свойством p;
- $F(o, p) = 0$ — есть аргументы как за, так и против того, что объект o обладает свойством p;
- $F(o, p) = T$ — неизвестно, обладает ли объект о свойством p.

Функция F может быть представлена в виде матриц, где $fij = T$, тогда говорят, что для пары (oi, pj) функция $F(oi, pj)$ не доопределена. Таким образом основная задача ДСМ-метода заключается в том, чтобы с помощью формирования гипотез доопределить исходную матрицу.

Правила первого рода

Сформируем гипотезы о всевозможных причинах свойств. В итоге получим функцию $H: C \times P \rightarrow V$.

- $H(c, p) = +1$ — с является возможной причиной наличия свойства p или (+)-гипотезой;
- $H(c, p) = -1$ — с является возможной причиной отсутствия свойства p или (-)-гипотезой;
- $H(c, p) = 0$ — есть аргументы как за то, что c является причиной наличия свойства p, так и за то, что c есть причина отсутствия этого свойства или (+)-гипотезой (противоречивой гипотезой);
- $H(c, p) = T$ — неизвестно, является ли c причиной наличия p или причиной отсутствия этого свойства.

Значения функции H для каждой пары (c, p) определяется с помощью правил правдоподобного вывода. Данные правила называются правилами первого рода. Сокращенное обозначение — PIR_1. Правила первого рода можно рассматривать как функцию, использующую матрицу F для получения матрицы H, т.е. $H = PIR1(F)$.

Пусть p — некоторое свойство. Тогда объект o является:

- положительным примером или (+)-примером для p относительно исходной матрицы F, если $F(o, p) = +1$,
- отрицательным примером или (-)-примером для p относительно исходной матрицы F, если $F(o, p) = -1$,
- противоречивым примером или (0)-примером для p относительно исходной матрицы F, если $F(o, p) = 0$.

Через $F+[p]$, $F-[p]$, $F\,0[p]$ обозначим множество всех положительных, отрицательных и противоречивых примеров для p относительно F, соответственно.

В качестве всевозможных причин наличия или отсутствия свойств объектов рассматриваются подмножества набора (множества) фрагментов C. Множество $C' \subseteq C$ удовлетворяет (+)-условию для p относительно F, только тогда, когда существует $\Omega \subseteq F+[p]$ такое, что:

1. каждый объект из Ω содержит все фрагменты из множества C', и не существует дополнительных фрагментов, которые принадлежат (o) для всех $o \in \Omega$;

2. Ω содержит как минимум два элемента (–)- и (0)-условия - аналогично.

Через $M+(F, c, p)$ обозначим тот факт, который удовлетворяет (+)-условию для p относительно F. Через $M-(F, c, p)$ - тот факт, который удовлетворяет (-)-условию для p относительно F . Через $M0(F, c, p)$ - тот факт, который с удовлетворяет (0)-условию для p относительно F.

Тогда определим функцию H, такую что:

$$H(C,P) = \begin{cases} +1, \text{если } M^+(F,c,p) \wedge \neg M^-(F,c,p) \wedge \neg M^0(F,c,p), \\ -1, \text{если } M^-(F,c,p) \wedge \neg M^+(F,c,p) \wedge \neg M^0(F,c,p), \\ 0, \text{если } (M^+(F,c,p) \wedge M^-(F,c,p)) \vee M^0(F,c,p), \\ T, \text{если } \neg M^+(F,c,p) \wedge \neg M^-(F,c,p) \wedge \neg M^0(F,c,p). \end{cases} \quad (1.2)$$

Другими словами, множество фрагментов $Ci \subseteq C$, доопределяется как:

- возможная причина наличия свойства p, при условии, что оно вкладывается в два и более (+)-примера, и не более чем в один (-)-пример) и не более чем в один (0)-пример;

• возможная причина отсутствия свойства *p,* при условии, что оно вкладывается в два и более (-)-примера, не более чем в один (+)-пример и не более чем в один (0)-пример.

Правила второго рода

При использовании матрицы гипотез о возможных причинах, можно сформировать гипотезы о наличии или отсутствии свойства *p* у тех объектов из *O*, для которых изначально не было известно, обладают они этим свойством или нет, т.е. для тех $o \in O$, для которых $F(o, p) = T$.

В результате мы получим функцию $F': O \times P \rightarrow V$. $F'(o, p) = F(o, p)$, если $F(o, p) \neq T$. Если же $F(o, p) = T$, то $F'(o, p)$ может принимать любое значение из *V*:

• $F'(o, p) = +1$ - *o* возможно обладает свойством *p*,

• F’(o, p) = -1 - *o* возможно не обладает свойством *p*,

• $F'(o, p) = 0$ - есть аргументы как за, так и против того, что объект *o* обладает свойством *p*,

• $F'(o, p) = T$ - доопределить ячейку исходной матрицы *F* не удалось.

Значения функции *F*' находятся с помощью правил правдоподобного вывода. Данные правила называются правилами второго рода. Сокращенное обозначение — PIR_2. Правила второго рода можно рассматривать как функцию, использующую матрицы *F* и *H* для получения матрицы *F'*, т.е. *F' = PIR2 (F, H).*

Пусть *o* — объект, *p* — свойство. Тогда скажем, что объект *o* удовлетворяет

• (+)-условию для *p* относительно *H* (т.е. возможно обладает свойством p), если существует такое $c \in C$, что $c \subseteq o$ и $H(c, p) = +1$.

• (-)-условию для *p* относительно *H* (т.е. возможно не обладает свойством *p*), если существует такое $c \in C$, что $c \subseteq o$ и $H(c, p) = -1$.

• (0)-условию для p относительно H (т.е. есть аргументы как за, так и против того, что o обладает свойством p), если существует такое $c \in C$, что $c \subseteq o$ и $H(c, p) = 0$.

Через $П+(H, o, p)$, $П-(H, o, p)$, $П0(H, o, p)$ обозначим тот факт, что объект o для свойства p относительно H удовлетворяет (+)-условию, (–)-условию и 0-условию, соответственно. Предположим: $F'(o, p) = F(o, p)$, если $F(o, p) \neq T$; в противном случае:

$$\mathrm{F}'(o,p) = \begin{cases} +1, \text{если } \Pi^{+}(H,o,p) \wedge \neg\Pi^{-}(H,o,p) \wedge \neg\Pi^{0}(H,o,p), \\ -1, \text{если } \Pi^{-}(H,o,p) \wedge \neg\Pi^{+}(H,o,p) \wedge \neg\Pi^{0}(H,o,p), \\ 0, \text{если } (\Pi^{+}(H,o,p) \wedge \Pi^{-}(H,o,p)) \vee \Pi^{0}(H,o,p), \\ T, \text{если } \neg\Pi^{+}(H,o,p) \wedge \neg\Pi^{-}(H,o,p) \wedge \neg\Pi^{0}(H,o,p). \end{cases} \quad (1.3)$$

Процедура индукции и аналогии последовательно используются до тех порка, пока в итоге их работы порождается хотя бы одна новая гипотеза, другими слова применение индукции приводит к изменению матрицы гипотез о возможных причинах свойств объектов, а применение аналогии приводит к изменению матриц гипотез о возможном обладании или не обладании свойства p у объектов. При всем этом номер шага указывает на правдоподобие рассуждений, выдвинутых гипотез.

1.3. Постановка задачи

Современная наука развивается достаточно быстро. Одной из особенностей развития науки является ее детерминированность, которая позволяет независимо от других областей развиваться со своей скоростью. Однако, несмотря на то, что каждая область науки или сфера деятельности человека имеет свою направленность, она может быть тесно связана с другими областями или как минимум пересекаться. В данной дипломной работе наблюдается тесная взаимосвязь таких областей науки как: психологический

анализ, статистика, программирование и интеллектуальные системы, которая позволяет решить большой спектр задач в ходе исследования.

Основной целью выпускной квалификационной работы является разработка программного обеспечения для анализа данных. Для достижения поставленной цели необходимо решить следующие задачи:

- Разработать программу для анализа экспериментальных данных.
- Протестировать программу на известном наборе данных.
- Провести анализ полученных результатов.
- Исследовать экспериментальные данные.

Важно отметить, что практическая значимость дипломной работы заключается в разработке программы для выявления причинно-следственных связей (отношений) в наборе данных, которая позволит произвести анализ больших объемов данных. Необходимо сказать, что в ходе решения поставленной задачи, охватываются научные области, которые занимают одну из новейших ступеней развития такие как машинной обучение, искусственный интеллект, анализ данных (сложных по своей структуре), что в свою очередь подчеркивает новизну и актуальность данной дипломной работы.

2. ПРОЕКТНАЯ ЧАСТЬ

2.1. Обоснование проектных решений

В настоящее время существует большой спектр средств разработки, проектирования, мониторинга и исследования программного продукта. Для выполнения поставленной задачи, было принято решение составить ряд требований к разрабатываемой программе и к используемым технологиям.

Основным требованием к разрабатываемому программному обеспечению является кроссплатформенность, так как при соблюдении данного требования конечный программный продукт можно «собрать» на большинстве современных операционных систем: Windows, OS X, Linux.

Главным требованием к используемым технологиям для разработки программного продукта является простота и скорость разработки. При соблюдении данного требования разработка программного продукта не будет трудоемкой.

Следующим требованием к используемым технологиям для разработки является его доступность и относительно небольшая ресурсоемкость. Доступность – одно из самых важных требований не только разработки программ, но и других областей. Так как если не будет доступен продукт, теряется актуальность его использования. Ресурсоемкость – требование которое отражает на сколько эффективно и комфортно удастся работать за данной технологией или продуктом.

Исходя из вышеперечисленных требований оптимальным решением для разработки программы является язык программирования Python. Основными преимуществами данного языка являются:

- Простота и скорость разработки. Разработка программ или приложений с использованием языка Python, когда требуется создать

небольшой программный продукт с минимальным графическим интерфейсом или вовсе без него требует меньше временных и вычислительных ресурсов в сравнении с языками C#, C, C++, Java.

- Открытость. Язык Python является открытым проектом и поддерживается немалым количеством программистов. Интерпретатор Python распространяются бесплатно. Также в свободном доступе есть больше количество программ, которые написаны без специальных технологий, что позволяет производить изучение и исследование без особых проблем.

- Кроссплатформенность. Программные продукты, которые написаны на языке Python можно «собрать и редактировать» на самых популярных операционных системах: Windows, OS X, Linux.

- Большое количество библиотек и модулей, которые находятся в свободном доступе. Данные библиотеки и модули позволяют сократить значительное количество времени, путем использования данных библиотек, т.е. не приходиться «изобретать велосипед заново».

В качестве используемых технологий (среды разработки) было принято решение использовать «PyCharm» от компании «JetBrains». JetBrains – одна из крупных компаний по производству программного обеспечения. Данная компания предоставляет большой набор продуктов таких как: InteliJ IDEA, PhpStorm, PyCharm, RubyMine, WebStrom, AppCode и другие. Одним из самых главных преимуществ данной компании является то, что есть возможность получить лицензионный продукт с официальной поддержкой бесплатно, при соблюдении условий регистрации со образовательным доменом (*.edu.*). Предоставленный лицензионный продукт дает возможность студентам обучаться и производить исследования без особых затрат. Также для исследования в рамках дипломной работы было принято решение установить IPython и Jupyter Notebook.

IPython – это интерактивная оболочка для Python, предоставляющая расширения для языка, дополнительный синтаксис, подсветку кода и

автодополнение кода. Данные возможности позволяют сократить значительное количество времени.

Jupyter Notebook – это одна из самых популярных бесплатных интерактивных оболочек для языка программирования Python, которая позволяет объединять код, тексты и диаграммы, которые можно распространять и использовать для исследований.

2.2. Описание алгоритма программы

2.2.1. Математическая структура программы

В разработанной программе структуру можно разделить на 3 этапа:

- Формирование описания группы испытуемых.
- Построение причинно-следственных связей (зависимостей).
- Графическое представление (построение графа).

Программа разработана на основе алгоритма, представленного в работе [12].

Пусть имеются три множества: $O=\{o_i\}$ – множество из n объектов, представляющие данные тестирования (исследования); $C=\{c_k\}$ – множество классов объектов, соответствующие группам, на которые были разбиты испытуемые; $P = \{p_j\}$ – множество из m признаков, которые обычно соответствуют определенным шкалам методик и опросников. Составляется матрица значений признаков $A_{ij}=\{a_{ij}\}$, в которой каждому признаку соответствует столбец его значений из матрицы A_{ij} $p_j \rightarrow (a_{1j}, a_{2j}, \ldots, a_{nj})$, а каждому объекту соответствует его описание $o_i \rightarrow (p_1=a_{i1}, p_2=a_{i2}, \ldots, p_m=a_{im})$, где пара $p_j=a_{ij}$ называется свойством объекта. Множество значений каждого интервального признака p_j делится на части, количество которых обычно равно трем: w_1 - высокое значение признака, w_2 - среднее значение и w_3 - низкое значение - $(a_{1j}, a_{2j}, \ldots, a_{nj}) = w_1 \cup w_2 \cup w_3$. В качестве алгоритма разбиения интервальной шкалы используется метод χ-слияния. На подготовительном

этапе алгоритма все объекты сортируются по величине интервального признака, а затем запускается итеративный процесс попарного слияния тех соседних интервалов, для которых оказалось минимальным значение χ^2. Слияние продолжается до тех пор, пока χ^2 не превысит некоторый порог, обычно равный 0,9. Значение χ^2 вычисляется по формуле:

$$\chi^2 = \sum_{i=1}^{m} \sum_{j=1}^{k} \frac{(A_{ij} - E_{ij})^2}{E_{ij}}, \quad (2.1)$$

где m – количество сравниваемых интервалов, k – количество классов (в нашем случае m=2, k=3), A_{ij} – количество примеров класса j в i-ом интервале,

$$E_{ij} = \frac{\sum_{j=1}^{k} A_{ij} \sum_{i=1}^{m} A_{ij}}{\sum_{j=1}^{k} \sum_{i=1}^{m} A_{ij}}, \quad (2.2)$$

где A_{ij} ожидаемая частота.

После чего начинается построение так называемых AQ-правил, которые представляют собой конъюнкцию свойств объектов. Правила строятся с использованием метода AQ-обучения [10, 11].

Первый шаг AQ-обучения представляет собой объявление всех объектов исследуемого класса C_k положительными примерами, а объекты прочих классов – отрицательными. На следующем этапе произвольным образом выбирается опорный пример, в свою очередь для которого формируется начальное правило из свойств объекта:

$$R_k^1 = \{P_1 = w_{i_1}, p_2 = w_{i_2}, \dots, p_m = w_{i_m}\}. \quad (2.3)$$

После правило дополняется путем, добавления интервалов (промежутков, расстояния в зависимости от данных) к некоторому признаку, или удалением из правила одного из признаков так, что результирующее

правило не покрывало ни один отрицательный пример. После выполнения данных «шагов» сформируется дерево правил, у которого листья являются более общими правилами, описывающими определенное количество положительных (удачных) примеров. После из более общих правил выбирается лучшее, покрывающее максимальное количество положительных (удачных) примеров и имеет минимальную длину. В качестве меры длины правила *R* выступает его сложность *Cx(R)*, вычисляемая как сумма весов, которые составляют данное правило. Веса операторов представлены в таблице ниже:

Таблица 2.1

Веса операторов

Оператор	Вес оператора
Оператор больше или меньше	1
Оператор равенства	1
Дизъюнкция интервалов	2
Интервал	2
Оператор «не равно»	2
Конъюнкция свойств	4
Дизъюнкция свойств	10

В итоге после применения AQ-метода, каждый класс C_k, ставится в соответствие наборам правил, которые представляют собой конъюнкцию свойств:

$$\Re_k = \left\{ R_{ki} | R_{ki} = \bigcap\nolimits_j \left(P_j = \bigcup\nolimits_q w_q \right) \right\} \quad (2.4)$$

Таким образом из-за того, что начальный объект с которого начинает формироваться правило, может быть любой, это значит, что результаты будут иметь различия, следовательно, AQ-метод запускается несколько раз.

В итоге после получения всех наборов правил $\mathfrak{R}_k$ для определенного класса C_k выбирается самый минимальный по сложности набор правил $\mathfrak{R}`_k$. После на основе $\mathfrak{R}`_k$ производится анализ структуры класса. В свою очередь объекты класса, описывающиеся одним правилом, формируют подкласс. Тогда, подклассы формируют структуру исследуемого класса объектов. В случае, когда подкласс состоит из одного объекта, данный объект может представлять собой ошибку или выйти за границу общей выборки, поэтому данному объекту уделяется особое внимание. В ходе многократных использований процедуры построения AQ-правил идет подсчет встречаемости свойства в правилах данного класса.

Описание определенного класса D_k составляют значимые свойства, такие что встречаемость свойства выше 1/3:

$$D_k = \{h_j \mid v(h_j) > 1/3\} \tag{2.5}$$

Для того чтобы определить, насколько описание класса подходит для выявления причинно-следственных отношений (связей, зависимостей) представлено два критерия.

1. Критерий сложности класса.

$$\theta_1(C_k) = \frac{|D_k|}{5m}, \tag{2.6}$$

где D_k является мощностью описания класса C_k, $5m$ максимальное количество возможных свойств тогда и только тогда, когда шкала делится на интервалы.

2. Критерий однородности класса.

$$\theta_2(C_k) = \frac{\sum_{R_k^*} \frac{C_x(R_{ik})}{Cov(R_{ik})}}{\sum_{R_k^*} Cx(R_{ik})}, \quad (2.7)$$

где $Cov(R_{ik})$ является количеством положительных примеров, которые описываются правилом R_{ik}. Данный критерий может принимать минимальное значение равное $1/N$, где N – количество объектов в класс тогда и только тогда, когда класс описан одним правилом и при этом покрывает все примеры, принимая максимальное значение равное одному, при условии, что каждый объект класса был описан свои правилом. Критерий однородности класса определяет, как объекты внутри класса схожи между собой, таким образом, что чем меньше значение критерия, тем более схожи объекты между собой.

Следующим этапом программы является построение причинно-следственных зависимостей, другими словами формирование гипотез. После того как значения критериев сложности и однородности класса, свидетельствуют о существовании случая выявления причинно-следственных зависимостей, используя описания классов формируется база фактов ДСМ-метода. Для того чтобы сократить пространство поиска для каждого класса производится сокращение множества свойств, используя два метода:

1. При условии, что два свойства $h_1: \left(p_1 = \bigcup_i w_i\right)$ *и* $h_2: \left(p_2 = \bigcup_j w_j\right)$ содержат одинаковый признак, т.е. $p_1=p_2$, и не вложенные, разные интервалы для данного признака, т.е. $\bigcup_i w_i \not\subset \bigcup_j w_j$ и $\bigcup_j w_j \not\subset \bigcup_i w_i$ и $\bigcup_j w_j \neq \bigcup_i w_i$, тогда такие свойства считаются конфликтными и эти будут исключены из базы фактов.

2. При условии, что два свойства содержат одинаковый признак, т.е. $p_1=p_2$, и вложенные, разные интервалы для данного признака, т.е. $\bigcup_i w_i \subset \bigcup_j w_j$ и $\bigcup_j w_j \subset \bigcup_i w_i$ и $\bigcup_j w_j \neq \bigcup_i w_i$, тогда такие свойства являются вложенными и исключается наиболее общее свойство.

Из всего множества свойств описания класса поэтапно выбирается целевое, где на основе остальных свойств формируются гипотезы о причинно-следственных отношениях. Объектами базы фактов являются не только объекты исследуемого класса, но и других классов тоже. Причинно-следственные отношения формируются в ДСМ-методе на основе следующих гипотез:

- Гипотеза сходства – если все случаи исследуемого явления имеют единственный общий фактор, тогда фактор является причиной исследуемого явления.
- Гипотеза различия – если оба случая схожи по всем факторам, кроме одного, и данный фактор присутствует в случае, когда явление возникает, тогда фактор является причиной исследуемого явления.
- Гипотеза абдукции – если множество факторов описывает (объясняет) множество выдвинутых гипотез, тогда данные гипотезы правдоподобны.

В результате работы ДСМ-метода выдвинутые гипотезы о наличии причины целевого свойства h_g: $\left(p_g = \bigcup_q w_q\right)$ для класса C_k имеют вид конъюнкции свойств: $H(h_g, C_k) = \bigcap_{j \neq g} \left(P_j = \bigcup_q w_q\right)$. Таким образом причина, которая состоит из одного свойства, называется простой, в противном случае называется комплексной. В ДСМ-методе гипотезы говорят о детерминации свойств, но не о зависимости признаков, в отличии от статистических гипотез. В свою очередь, свойство является конкретным подмножеством значений признака, таким образом связь признаков не подразумевает связь (зависимость) свойств и наоборот.

Множество гипотез, которые сформированы для свойства h_g, редуцируется по длине и по вложенности. Комплексная причина является незначимой, если содержит в себе более трех свойств. Также исключаются причины из множества гипотез, если они являются расширением других за счет конъюнкции свойств.

Ниже представлен алгоритм AQ-метода в сочетании с JSM-методом.

Вход: множество объектов $O=\{o_i\}$, разбитых на множество классов $C=\{c_k\}$, множество признаков $P=\{p_j\}$, матрица значений $A_{ij}=\{a_{ij}\}$.

- Шаг 1: из множества C выбирается исследуемый класс c_k и из него – начальный объект для AQ-метода.
- Шаг 2: с помощью AQ-обучения для класса c_k строится набор правил R_k, начиная с начального объекта.
- Шаг 3: для каждого свойства h_j из набора правил R_k подсчитывается встречаемость (с накоплением); выбирается новый начальный объект, если таковой имеется, и переходим к шагу 2, иначе выбирается новый класс c_k для исследования и также переходим к шагу 2; если все классы были рассмотрены переходим к шагу 4.
- Шаг 4: для каждого класса составляется его описание D_k и выбирается наилучшее правило $\mathcal{R}`_k$, вычисляются критерии $\theta1(c_k)$ и $\theta2(c_k)$; если значения какого-либо из них выше (ниже) критических порогов, то алгоритм заканчивает работу с пустым множеством причин для каждого класса, иначе переходим к шагу 5.
- Шаг 5: для каждого класса c_k на основе D_k формируется база фактов, в которой проводится сокращение множества свойств по вложенности и конфликтности.
- Шаг 6: выбирается исследуемый класс c_k и с помощью первого шага ДСМ-метода для него строятся гипотезы о наличии причинно- следственных связей между свойствами класса $H_k = \{H(h_g, c_k)\}$.
- Шаг 7: множество гипотез H_k сокращается путем исключения причин, длина которых больше критической, и причин, которые включены в другие причины, если остаются неисследованные классы, выбирается один из них и переходим к шагу 6.

Выход: $\{c_k, H_k\}$ – множество пар «класс – множество гипотез».

Следующим этапом структуры программы является построение графа. Ниже представлен алгоритм графопостроителя (graph_gen), который позволяет более подробно понять основную идею программы, также наглядно увидеть зависимость свойств.

Вход множество бинарных массивов (гипотез), список имен для гипотез.

- Шаг 1: Выбор положения и кол-ва вершин в строке, выбор метода отображения связей в графе, подготовка шаблона html для отображения графа.
- Шаг 2: Подсчет кол-ва гипотез и длины гипотез.
- Шаг 3: Построение вершин, сохранение пути ребер по вершинам.
- Шаг 4: Построение ребер графа.
- Шаг 5: Корректировка вершин (цвет, размер) для лучшего отображения.
- Шаг 6: Преобразование графа из JSON в HTML формат, передача графа в подготовленный шаблон.

Выход: Сгенерированных граф в формате HTML.

2.2.2. Структурная схема программы

Одним из главных этапов проектирования не только приложения, но и любого продукта является разработка структуры. Другими словами, то из каких «частей» будет состоять программа. Ниже на рис 2.1 представлена структурная схема программы, которая позволяет понять общую структуру программы.

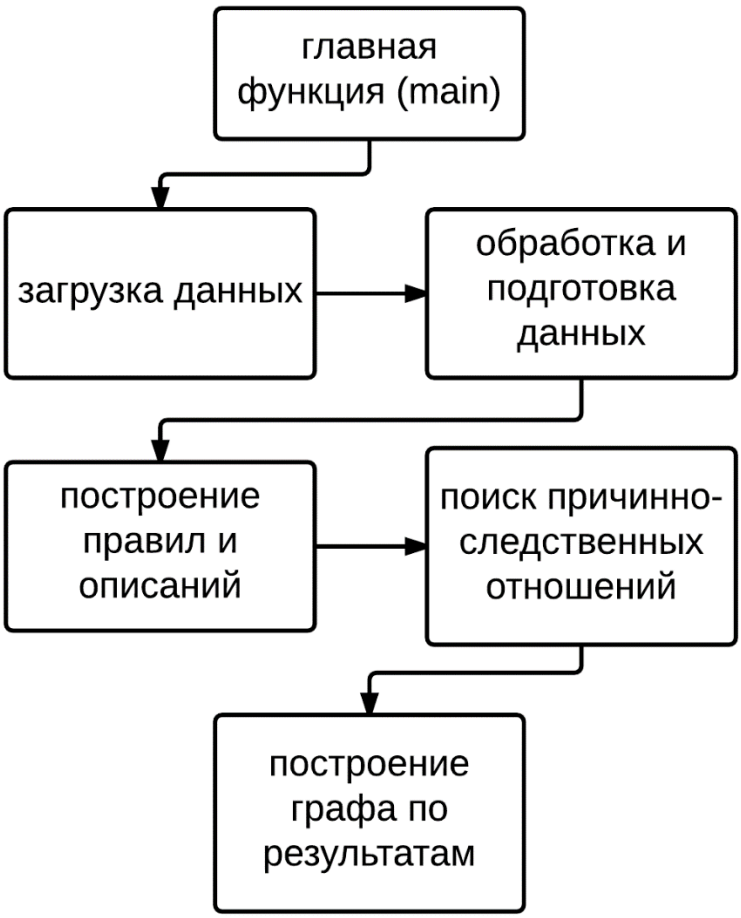

Рис. 2.1. Структурная схема программы

Глядя на структурную схему программы, есть представление об основных этапах работы программы. Данная схема позволяет визуально разделить программу на отдельные блоки, что упрощает ход разработки программы.

2.3. Разработка программы

Для разработки использовалась структурная схема программы, описанная в пункте 2.2.2. см. рис. 2.1 данной дипломной работы. Данная структура позволила создать функциональную схему программы, которая подробно описывает этапы и последовательность работы программы. Ниже на рис 2.2 представлена функциональная схема программы.

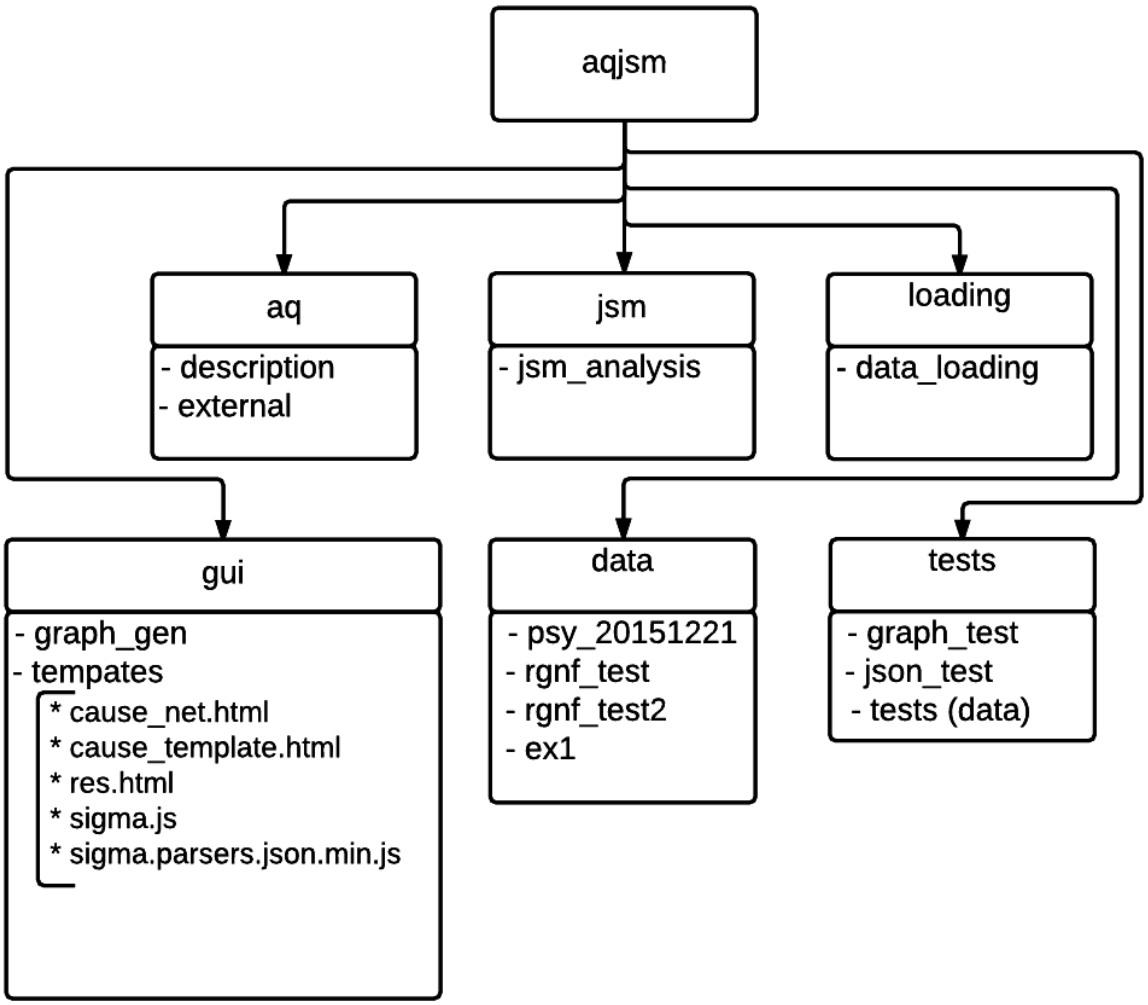

Рис. 2.2. Функциональная схема программы

Разработанная программа состоит из основных 6 модулей и одной главного модуля «aqjsm». Из главного модуля производится последовательный вызов других модулей, что позволяет распределить весь ход работы на отдельные этапы. Рассмотрим создание каждого модуля программы отдельно для более подробного понимания.

2.3.1. Создание модуля «aqjsm»

Создание главного модуля, который будет производить вызов других и в котором можно произвести мониторинг хода работы программы. Ниже на рис. 2.3 показан пример создания модуля в среде разработки PyCharm.

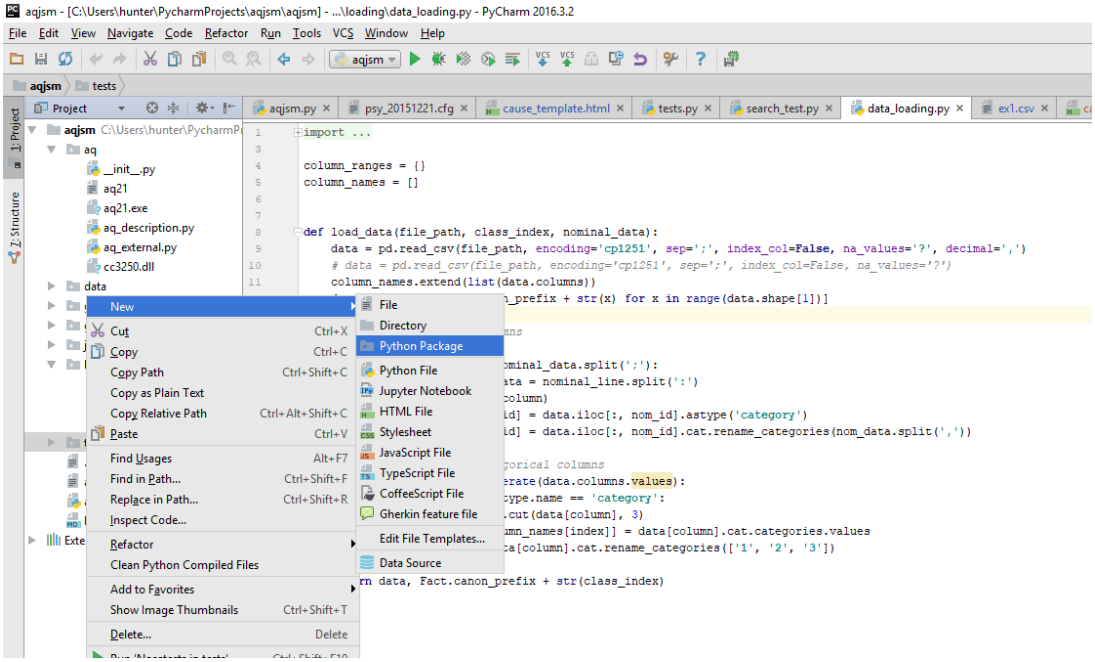

Рис. 2.3. Пример создания модуля в среде разработки PyCharm

Для создания необходимо вызвать контекстное меню в области проекта и выбрать соответствующее поле.

Данный модуль (файл) «aqjsm» является «главным», другими словами он выполняет роль функции «main». Из него производится запуск программы с конфигурациями, также он производит вызов функций загрузки данных, графопостроителя и других.

2.3.2. Создание модуля «aq»

Модуль «aq» имеет два основных файла «aq_external» и «aq_description». На рис. 2.4 изображен модуль «aq».

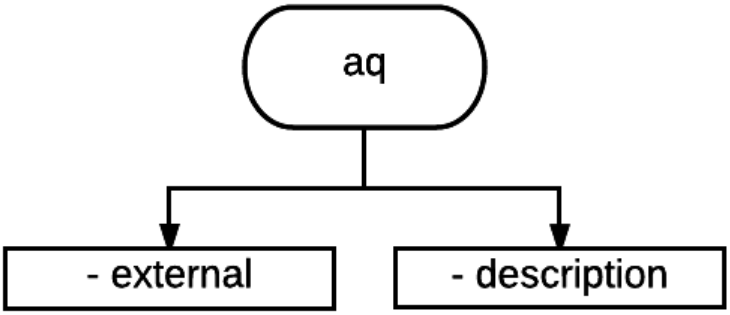

Рис. 2.4. Модуль «aq»

«aq_external» содержит функции, которые позволяют отображать ход программы в консоли при этом выводить данные о работе программы, также содержит описание запуска метода aq21 [11].

«aq_description» содержит функции, которые отображают интервалы, используемые для группировки элементов внутри класса. Также строится описание построения набора правил, начиная с начального объекта. Формируется описание каждого класса и выбирается наилучшее правило, после чего вычисляются критерии для оценки порогов.

2.3.3. Создание модуля «jsm»

Модуль «jsm» имеет единственный файл «jsm_analysis», но данный модуль выполняет основную часть работы всей программы. На данном шаге формируются гипотезы, которые описывают зависимость (отношения) данных. На рис. 2.5 изображен модуль «jsm».

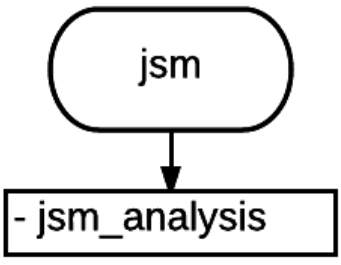

Рис. 2.5. Модуль «jsm»

На данном этапе выбирается исследуемый класс и с помощью первого шага ДСМ-метода формируются причинно-следственные зависимости.

После того как построены множества гипотез происходит сокращение методом исключения причин, длина который больше критичной, и причин, которые включены в другие причины, если остаются неисследованные классы, то выбирается один из них.

2.3.4. Создание модуля «gui»

Важное место в анализе данных имеет представление полученных результатов. Для более наглядного понимания результатов файл «graph_gen», генерирующий граф, который отображает важные свойства признаков. На вход графа поступают гипотезы в виде бинарных массивов, которые генерируются в процессе работы обучения и анализа данных. В качестве вершин выступают свойства признаков, а дуги (ребра) графа связь (зависимость) свойств признаков.

В данном модуле в папке «templates» содержатся файлы шаблона web-страницы для графа «cause_template.html», а также выходной файл «cause_net.html». В данном модуле содержатся файлы библиотеки, которые преобразуют данные из json в html формат «sigma.parsers.json.min.js» и отображают граф на web-странице «sigma.js».

Ниже на рис. 2.6 изображен модуль «gui».

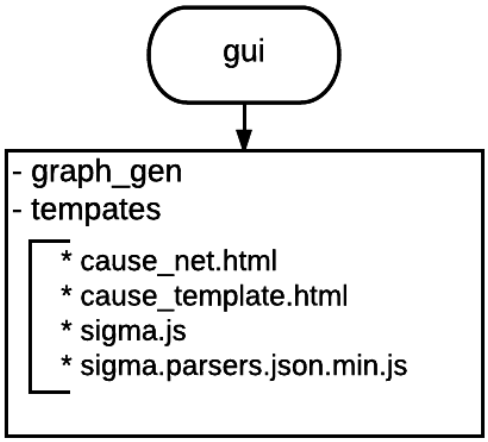

Рис. 2.6. Модуль «gui»

2.3.5. Создание модулей «loading», «data», «tests»

В программе также следует отметить «вспомогательные» модули. Иными словами, программу можно использовать и без них, но они предоставляют более удобное использование.

Модуль «loading» производит загрузку и предварительную обработку данных.

Модуль «data» выполняет роль обычной папки для хранения файлов, в ней содержатся файлы с экспериментальными данными, а также файлы конфигурации запуска программы, так как для каждого набора данных необходимо корректировать ход работы программы.

Модуль «tests» позволяет провести промежуточные тесты с экспериментальными данными, а также с построением графа по тестовым данным для настройки отображения на web-странице.

Ниже на рис. 2.7 изображен модули «loading», «data», «tests».

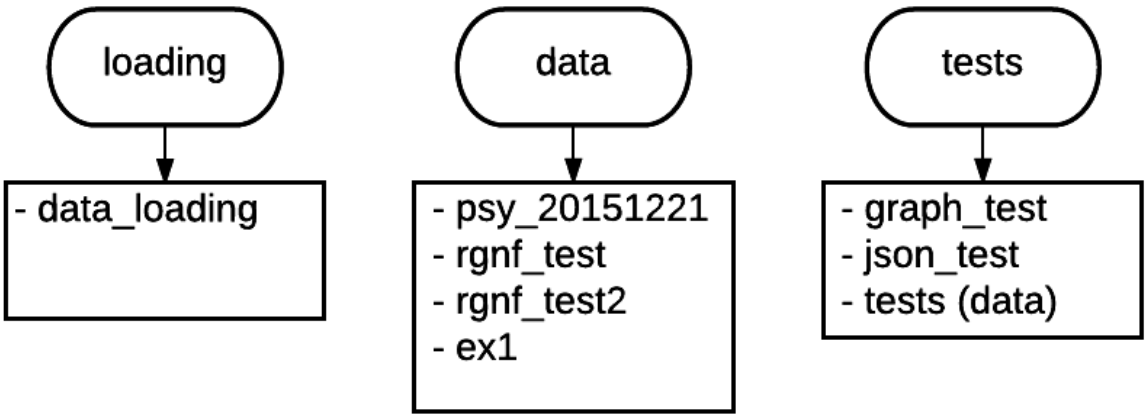

Рис. 2.7. Модули «loading», «data», «tests»

3. ТЕСТИРОВАНИЕ

3.1. Выбор тестовых данных

Тестирование – это относительно краткое испытание, использование, которое позволяет за короткие промежутки времени получить оценку результативности исследования. В данной дипломной работе процесс тестирования представляет один из основных этапов исследования, который позволяет проанализировать эффективность работы программы, позволяет в полной мере провести настройку и корректировку параметров запуска, а также вывод результатов.

В ходе тестирования программы было принято решение проверить работу на уже решенном примере. Для тестирования был выбран пример «квадрат», который представлен в описании ДСМ-метода [9]. Также для полной достоверности было принято решение использовать данные психологического тестирования представленные в работе [12]. Для основного исследования были использованы экспериментальные данные предоставленные институтом системного анализа РАН, так как в ходе исследования бывают моменты использования случайных данных, в следствии чего при использовании выходного продукта (программное обеспечение или устройство) на реально поставленной задаче может нести большую погрешность. Однако, в исследовании участвовала группа испытуемых, которая дала согласие на обработку данных тестирования, что подтверждает реальность экспериментальных данных и позволяет избежать погрешности между рандомизированными и реальными данными.

Выбранный пример «квадрат» ставит следующую задачу: необходимо дать ответ на следующий вопрос: какими свойствами должен обладать выпуклый четырехугольник с нетривиальной симметрией, чтобы вокруг него

можно было описать окружность, или напротив, нельзя было описать окружность.

Исходные данные для задачи представлены ниже в виде таблицы.

Таблица 3.1

Исходные данные

	p	c1	c2	c3	c4	c5	c6	c7	c8	c9
o1 (квадрат)	+	+	+	+	+	-	-	+	-	-
o2 (прямоугольник)	*T*	+	+	-	+	+	-	+	-	-
o3 (ромб)	-	+	+	+	-	+	-	-	+	+
o4 (параллелограмм)	-	+	-	-	-	+	+	-	+	+
o5 (равнобокая трапеция)	+	-	+	-	+	-	+	-	+	-
o6 (дельтоид)	-	-	+	+	-	-	+	-	+	+
o7 (прямоугольный дельтоид)	+	-	+	+	-	-	+	+	-	+

Где:

- o1, …, o7 – объекты;
- c1, …, c9 – набор структурных фрагментов C;
  - c_1 - есть центр симметрии;
  - c2 – есть ось симметрии;
  - c3 - есть ось симметрии, являющаяся диагональю;
  - c4 - есть ось симметрии, которая не является диагональю;
  - c5 - только один поворот на 180^{o} переведет фигуру в себя;
  - c6 -порядок группы симметрий равен двум;
  - c7 -есть пара противолежащих прямых углов;
  - c8 - нет прямого угла;
  - c9 - нет оси симметрии или любая ось симметрии является диагональю.
- p – множество целевых признаков;
- *T* – пример для которого необходимо определить обладает ли объект целевым свойством p или нет;
- -/+ – отсутствие и наличие свойства или признака у объекта.

3.2. Конфигурация запуска программы

Для примера «квадрат» конфигурация запуска программы имеет следующие параметры, представленные на рис. 3.1. Для более удобного конфигурирования запусков программы было принято решение для каждого набора данных использовать отдельный файл конфигурации.

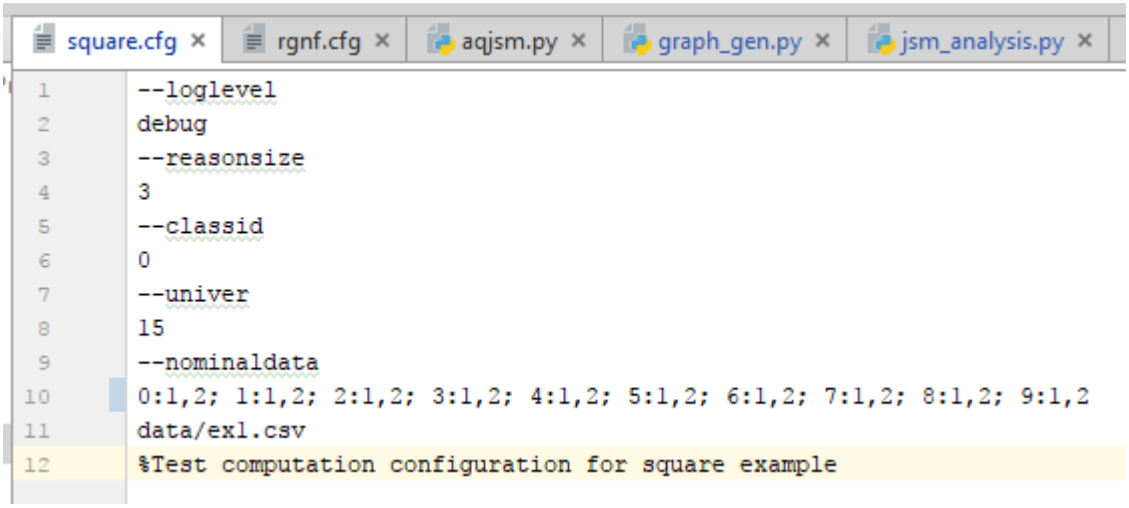

Рис. 3.1. Конфигурация запуска программы для примера «квадрат»

На рис. 3.2 представлен пример конфигурации запуска программы для данных психологического тестирования. Главным отличием от предыдущего примера «квадрат» является то, что в данном наборе данных нет номинальных элементов, т.е. только те элементы, которые распределяются по интервалам, следовательно, не обязательно указывать параметр «nominaldata». Параметр «univer» указывает максимальный размер набора свойств класса. Параметр «reasonsize» указывает максимальный размер причин фильтрации. Другими словами, данные параметры указывают длину правил, которые сформируются в результате работы программы.

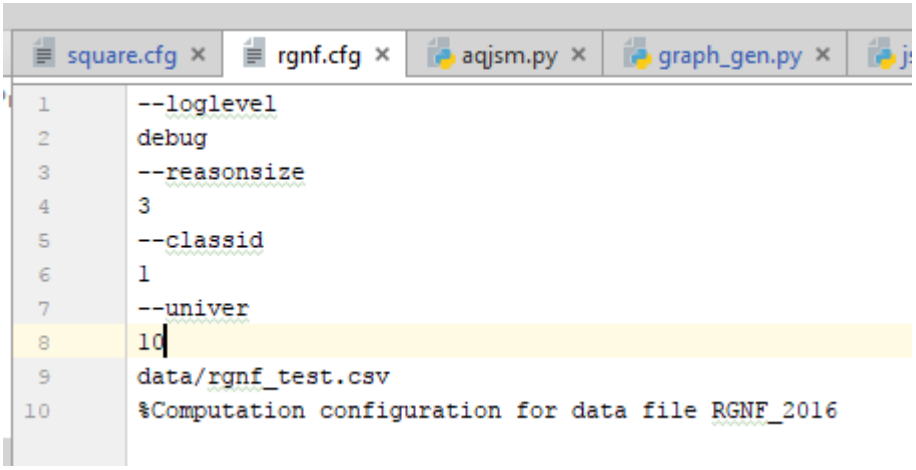

Рис. 3.2. Конфигурация запуска программы для данных психологического тестирования

3.3. Анализ полученных результатов

3.3.1. Анализ примера «квадрат»

В ходе проведенного эксперимента, мы получили следующие необходимые условия, при выполнении которых можно было описать окружность вокруг выпуклого четырехугольника, который обладает нетривиальной симметрией:

- Имеется ось симметрии, которая не является диагональю;
- Имеется ось симметрии диагональ, и при этом имеется пара противолежащих прямых углов.

Ниже на рис. 3.3 представлен результат работы программы, где в консоли отображается сформированная гипотеза. Другими словами, программа сформировала правило, при выполнении которого можно описать окружность вокруг выпуклого четырехугольника, который обладает нетривиальной симметрией. Следовательно, полученный бинарный массив дает ответ на поставленную задачу, т.е. в массиве элемент которого равен «1» говорится о необходимости обладания свойств, которые представлены в табл. 3.1.

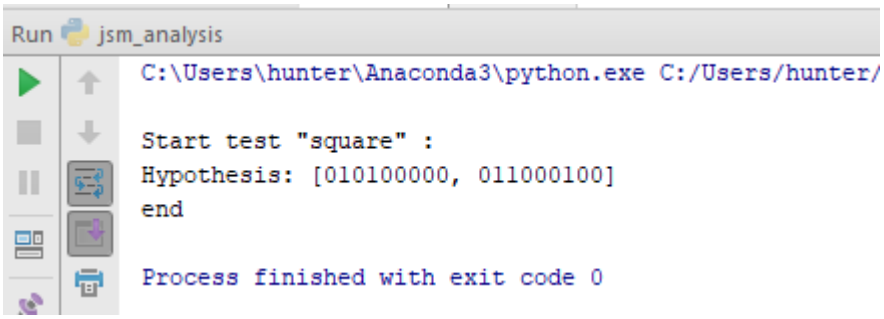

Рис. 3.3. Результат работы программы

На рис. 3.4 представлен результат работы программы, а именно построение графа, который отображает причинно-следственные зависимости. В ходе выполнения дипломной работы, было принято решение усовершенствовать отображение графа, т.к. на приведенном примере достаточно сложно понять причинно-следственные зависимости.

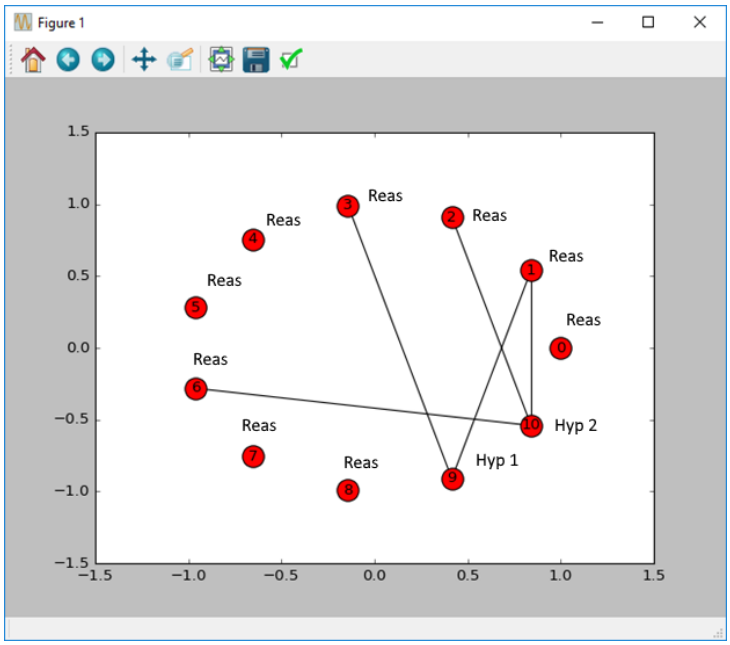

Рис. 3.4. Построение графа

Приведенный выше пример отображения графа является промежуточным, т.к. промежуточное тестирование работы программы позволяет устранить ошибки, которые возникают на этапе разработки. В ходе корректировки программы и устранения ошибок, разработанный графопостроитель более наглядно отображает причинно-следственные

зависимости. На рис. 3.5 представлен пример работы графопостроителя на примере «квадрат». На рисунке ниже видно, что для выполнения поставленной задачи в примере «квадрат», необходимо обладания свойств: c2, c4, а также свойств: c2, c3, c7. Вершина графа c2 имеет желтый цвет и относительно больший, который показывает, что данное свойство используется в двух сформированных правилах гипотезы.

Ребра графа также имеют отличительный цвет, который показывает связь одного правила. В левой части рис. 3.5 можно увидеть, как делятся вершины графа (структурные фрагменты или свойства) по уровню важности (High, Middle, Low). Видно, что свойства c2 (есть ось симметрии), связан с двумя условиями разных правил, что делает его более важным (Middle) по сравнению с другими (Low).

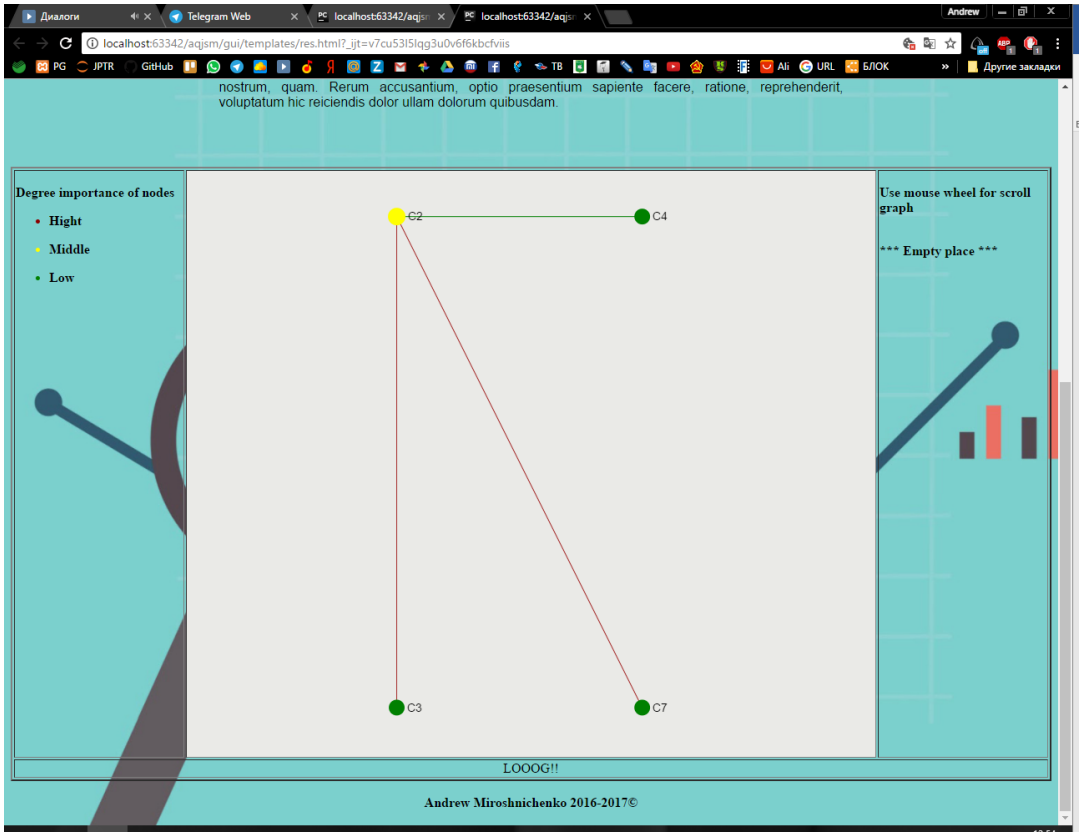

Рис. 3.5. Результат работы графопостроителя с примером «квадрат»

3.3.2. Анализ с использованием набора экспериментальных данных психологического тестирования

В ходе выполнения тестирования программы с использованием экспериментальных данных был проведен сравнительный анализ полученных результатов с корреляционным анализом. Ниже на рис. 3.6 показан этап формирования таблицы корреляций методами: Пирсона, Спирмена и Кендалла с использованием среды Jupyter Notebook.

```
In [36]: res.columns = ['col_p', 'pirson', 'col_s', 'spearman', 'col_k', 'kendall']

In [37]: res.head(10)
```

Out[37]:

	col_p	pirson	col_s	spearman	col_k	kendall
ОШКМ	ШЭС	0.920260	ШЭС	0.896268	ШЭС	0.744478
ШЭС	ОШКМ	0.920260	ОШКМ	0.896268	ОШКМ	0.744478
Ер	Е	0.817276	Е	0.786770	Е	0.636634
Е	Ер	0.817276	Ер	0.786770	Ер	0.636634
Аналитический стиль	Директивный стиль	0.788191	Директивный стиль	0.654156	Директивный стиль	0.475290
Директивный стиль	Аналитический стиль	0.788191	Аналитический стиль	0.654156	Аналитический стиль	0.475290
Общий	Воспитание	0.787401	Воспитание	0.783782	Воспитание	0.604592
Воспитание	Общий	0.787401	Общий	0.783782	Общий	0.604592
ED	Е	0.786457	NP	0.780230	NP	0.607366
Личный опыт	Общий	0.782662	Общий	0.748800	Общий	0.578286

Таблица сравненй корреляционных данных

Рис. 3.6. Корреляция в среде Jupyter Notebook

Ниже в табл. 3.2 представлена корреляция признака «ШЭС» при (p <0.05)

Таблица 3.2

Корреляция признака «ШЭС»

СКС	Воспитание	СМИ	ОШКМ	ШЕМ	ШВ
-0,405	-0,518	-0,598	0,806	-0,426	0,507

В ходе работы программы было построено четыре гипотезы для свойства «ШЭС», из анализа полученных гипотез следует, что свойство «СКС» и «ШВ» входит во все гипотезы, что в свою очередь говорит о значительном влиянии на исследуемое свойство, однако не является его причиной. В этом рассматриваемом случае корреляционное и индуктивное

исследование показали два общих признака «ШЭС» и «ОШКМ», при этом симметрия в корреляциях между данными признаками очевидна, однако с использованием ДСМ-метода взаимная обусловленность не будет являться необходимым результатом.

На рис. 3.7 представлен пример работы графопостроителя с использование экспериментальных данных.

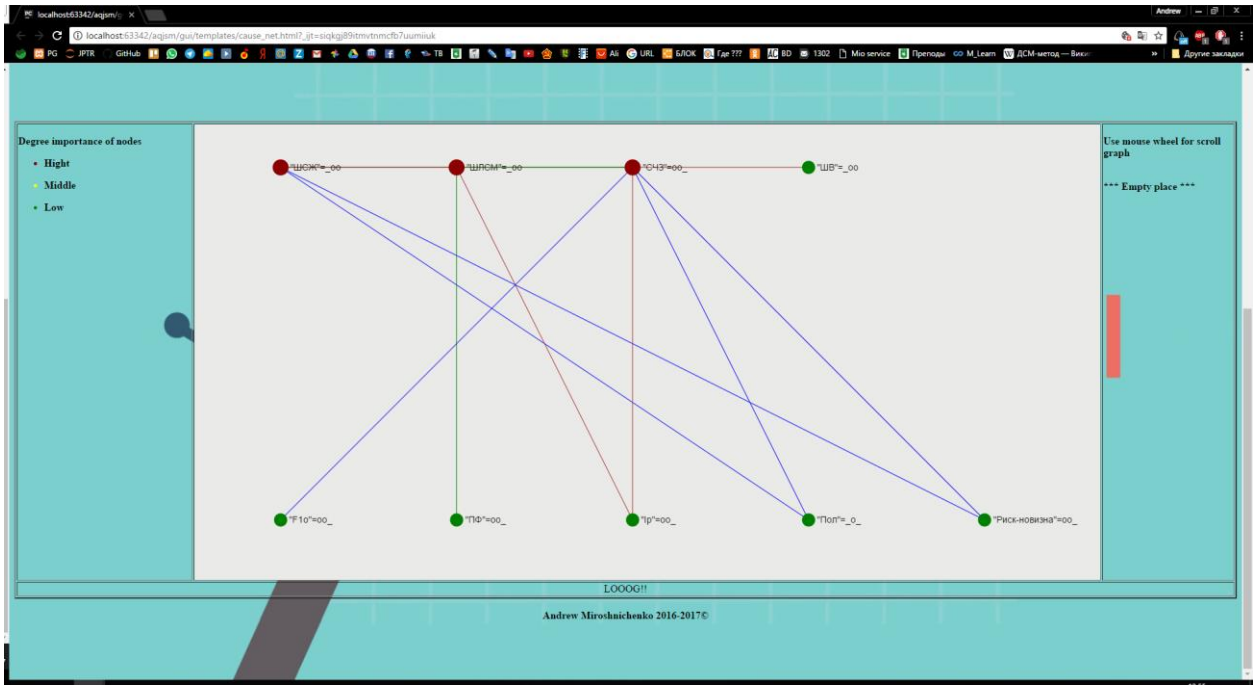

Рис. 3.7. Результат работы программы с экспериментальными данными

Также с другими иллюстрациями запуска, и результатами работы программы можно ознакомится в приложении 1.

В ходе выполнения дипломной работы, разработанная на основе алгоритма выявления причинно-следственных отношений программа, позволила выдвинуть ряд интерпретируемых гипотез о наличии каузальных зависимостей между свойствами класса объектов, которые представлены в наборе экспериментальных данных, а также программа позволила отобразить выдвинутые гипотезы в виде связного графа для более наглядного понимания зависимостей.

ЗАКЛЮЧЕНИЕ

В настоящее время объем данных, вчастности информационных становится все больше и больше, что создает проблему обработки, анализа и представления этих данных. Анализ данных является важной задачей почти в любой сфере деятельности человека, в том числе в научных исследованиях. Таким образом основная задача анализа данных – это выбор самой важной и ключевой информации. Отбор важной информации определяется некоторым набором правил, условий или соответствий. При анализе человеком тысяч и даже миллионов строк данных есть такой человеческий фактор, когда человек может допустить ошибку в следствии усталости или невнимательности. Однако вычислительная техника, такая как компьютер лишен этого фактора, что позволяет избежать таких ошибок. В настоящее время повышается популярность и актуальность таких понятий как: машинное обучение, искусственный интеллект, нейронные сети. Эти технологии повышают эффективность обработки и анализа колоссальных объемов данных.

Актуальность данной выпускной квалификационной работы заключается в том, что представленная программа выявления причинно-следственных отношений в экспериментальных данных, позволяет провести анализ большого набора данных и выявляет самую важную информацию, затем идет процесс обработки полученных данных после чего результат представляется в виде графа, так как по данных некоторых исследователей человек легче всего воспринимает графическую информацию нежели обычный текст.

Главной целью данной дипломной работы является разработка математического и программного обеспечения, позволяющего выявить причинно-следственные отношения в наборе экспериментальных данных большого объема.

Поставленная цель дипломной работы позволила выполнить следующие задачи:

- Разработать программу для анализа экспериментальных данных.
- Протестировать программу на известном наборе данных.
- Исследовать экспериментальные данные.
- Провести анализ полученных результатов.

При анализе данных часто требуется объяснить зависимость или независимость определенных свойств, признаков у группы исследуемых. Например, необходимо объяснить зависимость уровня подготовки спортсмена к соревнованиям от его личностных характеристик таких как: соперничество и агрессивность. В данном примере под фактом подразумевается уровень соперничества или агрессивности. С помощью которого описывается группа спортсменов. Следовательно, возникает основная задача описать каузальные отношения между значениями признаков или самими признаками. Важно отметить, что в данном примере прослеживается корреляция между признаками, как прямая зависимость, но это не значит, что изменение одной из характеристик с изменением другой будет давать точный ответ что является причиной, а что следствием. Точно также можно сказать о факторном анализе, в процессе выявления линейных комбинаций признаков, будут являться лишь факторами, по которым разделяются признаки. Проблема интерпретации результатов статистического анализа является, что результаты можно использовать для уровня достоверности уже выдвинутых гипотез о казуальной зависимости.

Отдельно следует отметить что имеет место случай, когда признак или совокупность признаков зависит от цело ряда признаков. Выделить такие зависимости с использованием только корреляции достаточно трудная задача.

Также кроме статистического анализа данных в машинном обучении часто применяется интеллектуальный метод анализа данных, в частности индуктивные методы, в частности ДСМ-метод. Данный метод является

автоматическим методом порождения гипотез, формализующий схему правдоподобного и достоверного вывода, который называется ДСМ-рассуждение. ДСМ-рассуждение является синтезом познавательных процедур: индукции, аналогии и абдукции. ДСМ-метод был создан как средство автоматизированного построения формализации знаний о предметной области средствами называемых квазиаксиоматических теорий.

Важное место в анализе данных имеет представление полученных результатов. Для более наглядного понимания результатов в данной дипломной работе представлен метод генерирующий граф, который отображает важные свойства признаков. На вход графа поступают гипотезы в виде бинарных массивов, которые генерируются в процессе работы обучения и анализа данных. В качестве вершин выступают свойства признаков, а дуги (ребра) графа отображают связь (зависимость) свойств признаков.

В дипломной работе представлена программа разработанная на основе алгоритма, выявления причинно-следственных отношений в наборе экспериментальных данных на примере теста с использованием AQ обучения и ДСМ-метода, а также графопостроитель для более наглядного понимания каузальных зависимостей.

Проведенное исследование в ходе выполнения дипломной работы показало, что разработанная программа представляет удобный инструмент для анализа слабоструктурированных статистических данных в различных областях знаний.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Психологическое тестирование [электронный ресурс]. – Режим доступа: http://www.psychologos.ru/articles/view/psihologicheskoe_testirovanie

2. Анализ данных [электронный ресурс]. – Режим доступа: http://cyberleninka.ru/article/n/kategoriya-dannye-ponyatie-suschnost-podhody-k-analizu

3. Малая советская энциклопедия. — М.: Советская энциклопедия, 1960. — Т. 8. — С. 1090.

4. Статистика [электронный ресурс]. – Режим доступа: http://statlab.kubsu.ru/node/4

5. Мирошниченко А.С., Гурьянова И.В. Разработка и исследование нейронных сетей // Роль и значение современной науки и техники для развития общества, 28 апреля 2017г., г. Екатеринбург, ч.3 С. 85–90

6. Машинное обучение [электронный ресурс]. – Режим доступа: http://www.machinelearning.ru/wiki/index.php?title=Машинное_обучение

7. Финн В.К. Правдоподобные рассуждения в интеллектуальных системах типа ДСМ // Итоги науки и техники. 1991. Т. 15. С. 54-101.

8. Мирошниченко А.С. Выявление причинно-следственных отношений в наборе экспериментальных данных // Инновационные механизмы решения проблем научного развития, 18 марта 2017г., г. Уфа, ч.2 67-72

9. ДСМ-метод [электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/ДСМ-метод

10. Michalski R.S. AQVAL/1-Computer Implementation of Variable-Valued Logic System VL1 and Examples of its Application to Pattern Recognition // Proc. Of the First Int. Joint Conf. on Pattern Recognition. Washington, DS, 1973. P. 3-17.

3

11. The aq21 natural induction program for pattern discovery: Initial version and its novel features / Janusz Wojtusiak, Ryszard S. Michalski, Kenneth A. Kaufman, Jaroslaw Pietrzykowski // ICTAI. 2006. P. 523–526.

12. Панов А. И. Выявление причинно-следственных связей в данных психологического тестирования логическими методами // Искусственный интеллект и принятие решений. — 2013. — № 1. — С. 24–32.

ПРИЛОЖЕНИЕ 1

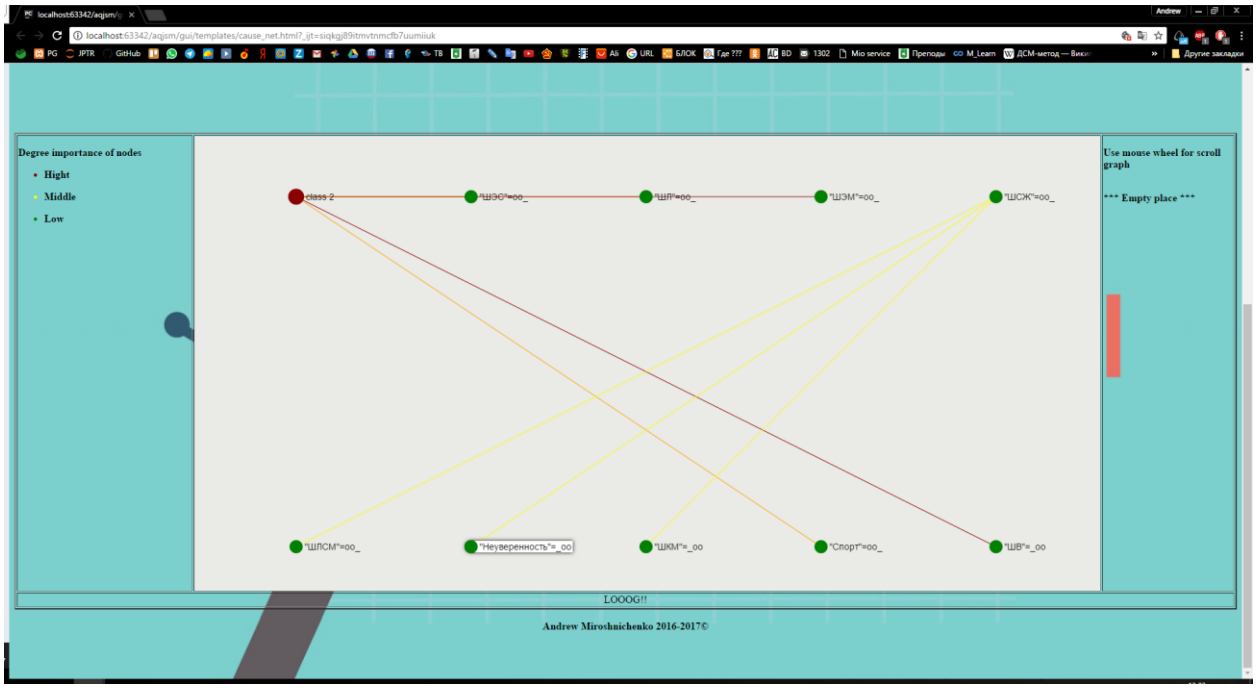

Рис. 1. Результат работы программы с «экспериментальными данными 1»

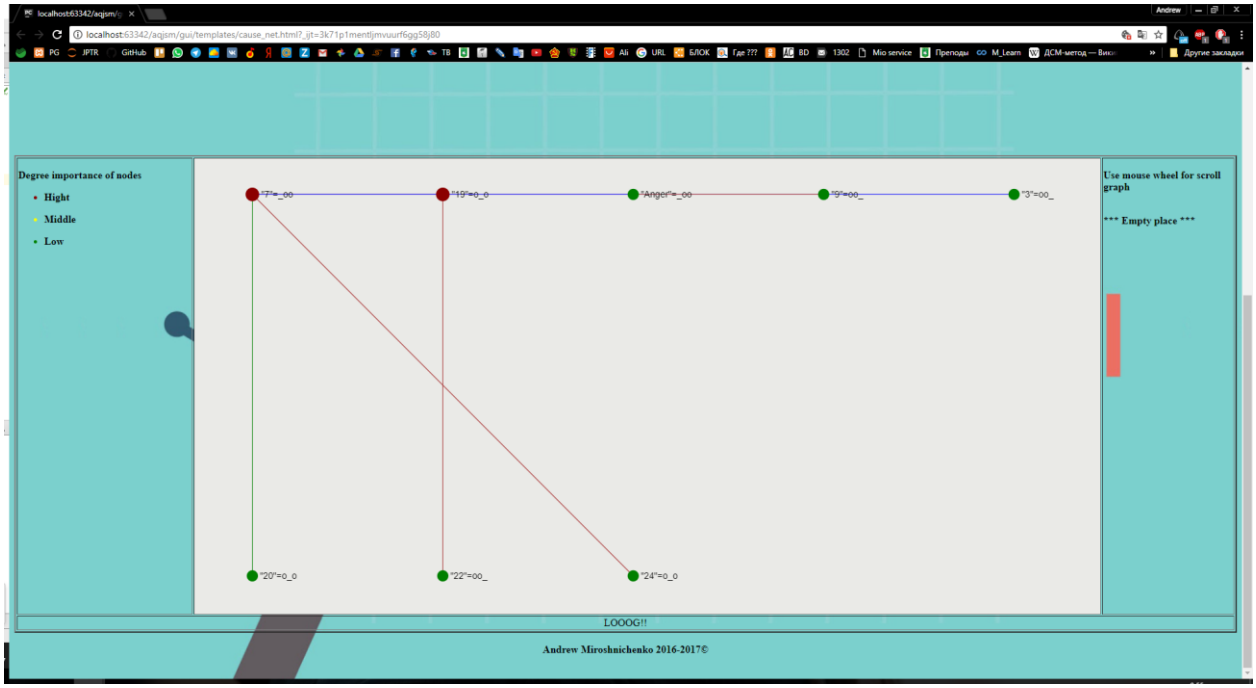

Рис. 2. Результат работы программы с «экспериментальными данными 2»

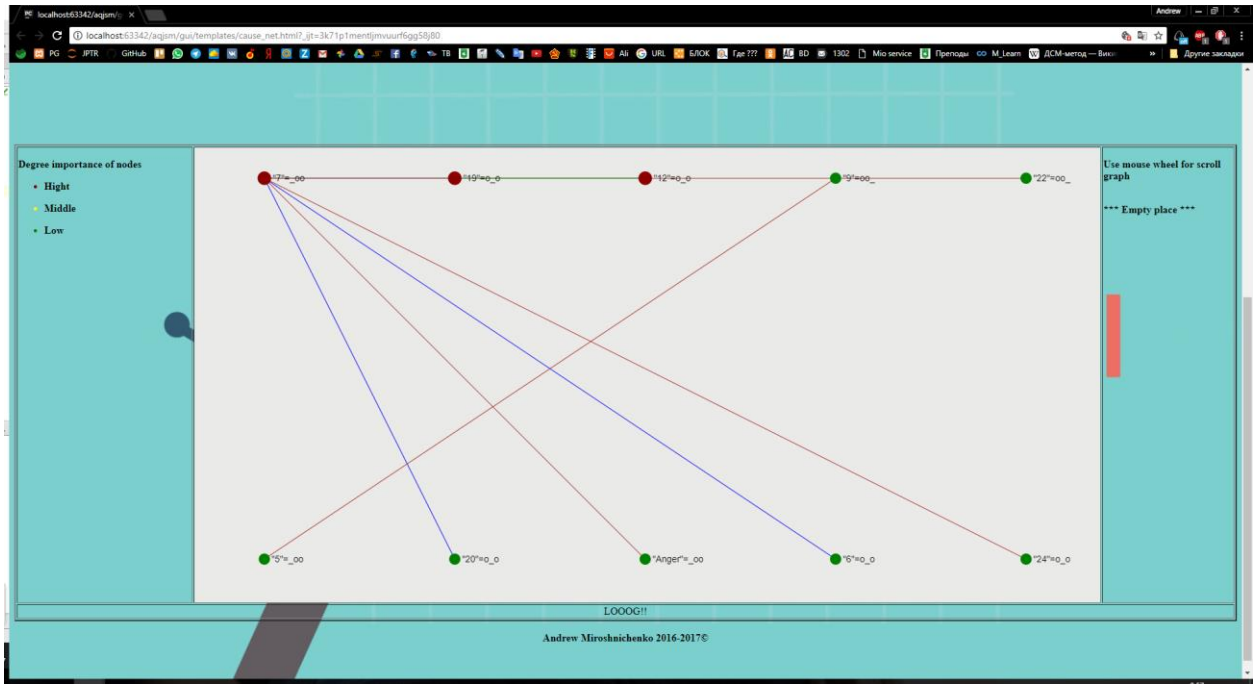

Рис. 3. Результат работы программы с «экспериментальными данными 3»

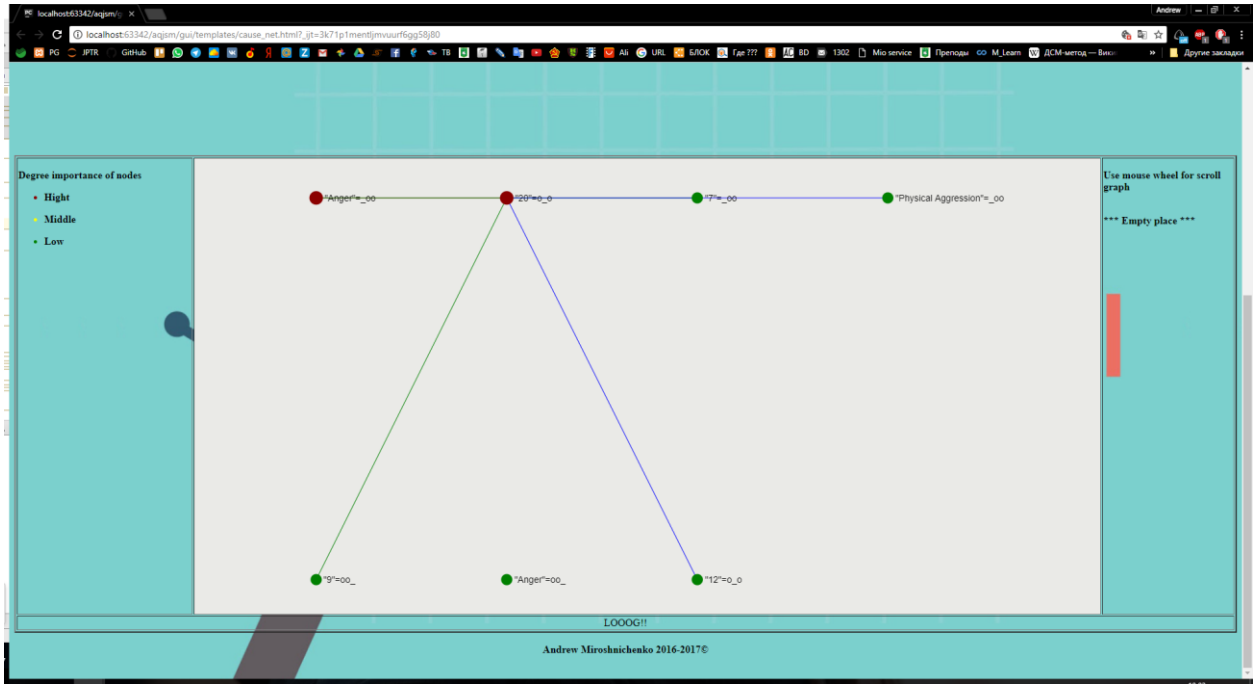

Рис. 4. Результат работы программы с «экспериментальными данными 4»

ПРИЛОЖЕНИЕ 2

Листинг «aqjsm.py»

```
import argparse
import datetime
import logging
import platform

import aq.aq_external as aq
import loading.data_loading as dl
from aq.aq_description import Fact
from jsm.jsm_analysis import FactBase, search_norris
import pandas as pd
from gui.graph_gen import generate_graph

log_levels = ['debug', 'info', 'warning', 'error']

def parse_args():
    argparser = argparse.ArgumentParser(description='AQJSM causal relations
miner',

formatter_class=argparse.ArgumentDefaultsHelpFormatter,
                                fromfile_prefix_chars='@')
    argparser.add_argument(dest='datafile')
    argparser.add_argument('-l', '--loglevel', choices=log_levels, default='info',
                           help='Logging level')
    argparser.add_argument('-s', '--reasonsize', type=int, default='3',
                           help='Maximum size of causes for filtering')
    argparser.add_argument('-u', '--univer', type=int, default='30',
                           help='Maximum size of the set of class properties')
    argparser.add_argument('-c', '--classid', type=int, required=True,
                           help='Index of class column in data file (starting from 0)')
    argparser.add_argument('-n', '--nominaldata',
                           help='Data string of information about nominal columns in
format: <col_id1>:<nom1>,<nom2>,...;<col_id2>:<nom1>...')
    args, comment = argparser.parse_known_args()

    return args, comment


def configure_logger():
    rootLogger = logging.getLogger()
    logFormatter = logging.Formatter('%(asctime)s %(levelname)-8s  %(message)s',
datefmt='%H:%M:%S')
```

```
    consoleHandler = logging.StreamHandler()
    consoleHandler.setFormatter(logFormatter)
    rootLogger.addHandler(consoleHandler)
    fileHandler = logging.FileHandler('aqjsm.log', encoding='cp1251')
    fileHandler.setFormatter(logFormatter)
    rootLogger.addHandler(fileHandler)
    rootLogger.setLevel(args.loglevel.upper())

    logging.info('OS: {0}, date: {1}'.format(platform.platform(),
datetime.datetime.now().strftime("%Y-%m-%d")))


if __name__ == "__main__":

    args, comment = parse_args()
    configure_logger()
    logging.info(args)
    logging.info(comment)

    max_universe_size = args.univer
    max_reason_length = args.reasonsize
    class_index = args.classid
    nominal_data = args.nominaldata

    data, class_column = dl.load_data(args.datafile, class_index, nominal_data)
    logging.info('Data file {0}: {2} columns, {3} objects, class column is
"{1}"'.format(args.datafile,

dl.column_names[class_index],
                                                    *reversed(data.shape)))
    logging.debug('\n\t'.join(['"{0}": {1}'.format(key, dl.column_ranges[key]) for
key in sorted(dl.column_ranges)]))

    class_descriptions = aq.run_aq(data, class_column, dl.column_names)
    for desc in class_descriptions.values():
        desc.build(max_universe_size)
    logging.info('\n'.join([str(class_descriptions[d]) for d in class_descriptions]))

    save_mas = []  # for graph_gen()
    name_reas = []  # for Nodes name graph_gen()

    for klass in data[class_column].unique():
        logging.info('\n' * 3 + '*' * 5 + 'Start search reasons for class
{0}'.format(klass) + '*' * 5)
```

```
        logging.info('Start search reasons for class property {0}'.format(klass))
        fb = FactBase(Fact(class_index, {klass}, 'class'))
        fb.build(data, class_descriptions[klass])
        fb.clear()



        def _search_in_fb(data_fb, target):
            hypotheses = search_norris(data_fb)
            reasons = []
            for hyp in hypotheses:
                if hyp.value.count() <= max_reason_length:
                    reasons.append((hyp.generator.count(),
                                    [data_fb.properties[i] for i in range(len(hyp.value)) if
                                     hyp.value[i]]))
            if reasons:
                reasons.sort(key=lambda x: x[0], reverse=True)
                logging.info('\tFound {0} reasons for {1}:\n\t'.format(len(reasons),
target) + '\n\t'.join(
                    ['[{0}]: '.format(q) + ' & '.join([str(f) for f in r]) for q, r in reasons]))
            else:
                logging.debug('\tWas not found reasons for {0}'.format(target))

            if hypotheses:
                save_mas.append(hypotheses)
                name_reas.append(str(target))

        for prop in class_descriptions[klass].properties:
            logging.info('Start search reasons for property {0}'.format(prop))
            fb = FactBase(prop)
            fb.build(data, class_descriptions[klass])
            fb.clear()

            _search_in_fb(fb, prop)


    generate_graph(save_mas, 'gui/templates/cause_net.html', name_reas)
```

Листинг «jsm_analysis.py»

```
import itertools
import logging

import pandas as pd
from bitarray import bitarray
from aq.aq_description import Fact

class FactBase:
    def __init__(self, target_prop):
        self.target_prop = target_prop
        self.positives = {}
        self.negatives = {}
        self.properties = []
        self.num_events = 0
        self.num_props = 0

    def __str__(self):
        return 'FactBase for property {0} ({1} props, {2} events: p={3},
n={4}):\n\t'.format(self.target_prop,
          len(self.properties),  self.num_events, len(self.positives), len(self.negatives))
+ '\n\t'.join(
            [b.to01() for b in itertools.chain(self.positives.values(),
self.negatives.values())])

    def __repr__(self):
        return 'FactBase for property {0} (p={1}, n={2}):\n\t'.format(self.target_prop,
                                                    len(self.positives), len(self.negatives))

    def build(self, data, class_description):
        target_index = data.columns.get_loc(self.target_prop.canon_attr_name)
        self.properties = [prop for prop in class_description.properties if not prop ==
self.target_prop]
        self.num_props = len(self.properties)
        dup_counter = 0
        miss_counter = 0
        for i, row in data.iterrows():
            data_value = row[target_index]
            if not pd.isnull(data_value):
                b = bitarray(self.num_props)
                for j, prop in enumerate(self.properties):
                    value = row[prop.attr_id]
                    b[j] = False if pd.isnull(value) else value in prop.values
                if data_value in self.target_prop.values and b not in
```

```
self.positives.values():
                    self.positives[i] = b
                elif data_value not in self.target_prop.values and b not in
self.negatives.values():
                    self.negatives[i] = b
                else:
                    dup_counter += 1
            else:
                miss_counter += 1
        self.num_events = len(self.positives) + len(self.negatives)
        logging.debug('\tDelete {0} duplicated events'.format(dup_counter))
        logging.debug('\tMiss {0} missing target column
events'.format(miss_counter))

    def clear(self):
        counter = 0
        for key in list(self.negatives.keys()):
            if self.negatives[key] in self.positives.values():
                del self.negatives[key]
                counter += 1
        self.num_events -= counter
        logging.debug('\tDelete {0} conflicted events'.format(counter))

class JSMHypothesis:
    def __init__(self, value, generator):
        self.value = value
        self.generator = generator
    def __str__(self):
        return 'Hypothesis {0} by {1}'.format(self.value.to01(), [i for i, x in
enumerate(self.generator) if x])
    def __repr__(self):
        return self.value.to01()
    def __eq__(self, other):
        return self.value == other.value and self.generator == other.generator
    def __ge__(self, other):
        return ((self.value | other.value) == self.value) and ((self.generator |
other.generator) == self.generator)
    def __hash__(self):
        return 3 * hash(self.value.to01()) + 5 * hash(str(self.generator))

def search_norris(fb):
    if fb.positives and fb.negatives:
        pos_inters = _search_norris(fb.positives)
        neg_inters = _search_norris(fb.negatives)
```

```
        logging.debug('\tIt was found {0} pos and {1} neg
hypothesis'.format(len(pos_inters), len(neg_inters)))
        conf_counter, dup_counter = 0, 0
        for p_inter in pos_inters[:]:
            for n_inter in neg_inters:
                unit = p_inter.value | n_inter.value
                if len(p_inter.generator) < 2 or unit == p_inter.value or unit ==
n_inter.value:
                    pos_inters.remove(p_inter)
                    conf_counter += 1
                    break

        l = pos_inters[:]
        for i, tmp1 in enumerate(l):
            for j, tmp2 in enumerate(l):
                if not i == j and (tmp1.value | tmp2.value) == tmp1.value:
                    pos_inters.remove(tmp1)
                    dup_counter += 1
                    break

        logging.debug('\tIt were deleted {0} conflicted and {1} surplus
hypothesis'.format(conf_counter, dup_counter))

        return pos_inters
    else:
        logging.debug('\tThere is no positives or negatives examples in FB')
        return []

def _search_norris(positives):
    # Relation R=AxB, A - objects, B - features, Mk - maximal rectangles (maximal
intersections)
    hypotheses = []
    # print(positives.keys())
    b = bitarray(max(positives.keys()) + 1)
    # print(b)
    b.setall(0)
    for key, value in positives.items():  # find object xkR
        # compute collection Tk={Ax(B intersect xkR): AxB in Mk-1}
        tmp_gen = [JSMHypothesis(value & h.value, h.generator.copy()) for h in
hypotheses]
        # eliminate the members of Tk which are proper subsets of other members of
Tk;
        # remaining sets are the members of T'k
```

```
        tmp_hyps = []
        for i, tmp1 in enumerate(tmp_gen):
            if tmp1.value.any():
                for j, tmp2 in enumerate(tmp_gen):
                    if not i == j and tmp2 >= tmp1:
                        tmp_hyps.append(None)
                        break
                else:
                    tmp_hyps.append(tmp1)
            else:
                tmp_hyps.append(None)

        # for each CxD in    Mk-1
        new_hyps = []
        add_example = True
        for i, hyp in enumerate(hypotheses):
            # if D subsetoreq xkR then (C unite xk)xD in Mk
            if (hyp.value | value) == value:
                hyp.generator[key] = 1
            else:
                # if D not susetoreq xkR then CxD in Mk, and (C unite xk)x(D intersect
xkR) in Mk
                # if and only if emptyset noteq Cx(D intersect xkR) in T'k
                new_hyp = tmp_hyps[i]
                if new_hyp:
                    new_hyp.generator[key] = 1
                    new_hyps.append(new_hyp)
            if not value.any() or (hyp.value | value) == hyp.value:
                add_example = False

        hypotheses.extend(new_hyps)
        # xk x xkR in Mk if and only if emptyset noteq xkR notsubsetoreq D for all
XxD in Mk - 1
        if add_example:
            c = b.copy()
            c[key] = 1
            hypotheses.append(JSMHypothesis(value, c))
            # print(hypotheses)
    return hypotheses


def test1():
    fb = FactBase(Fact(0, {'1'}))
```

```
    fb.positives = {1: bitarray('11000'), 2: bitarray('11010'), 3: bitarray('11100')}
    fb.negatives = {4: bitarray('00101'), 5: bitarray('00110'), 6: bitarray('00011')}

    hypotheses = search_norris(fb)
    hyp1 = hypotheses
    # print('\n'.join(map(str, hypotheses)))
    return hyp1


def test2():  # square
    fb = FactBase(Fact(0, {'1'}))
    fb.positives = {1: bitarray('111100100'), 2: bitarray('010101010'), 3:
bitarray('011001101')}
    fb.negatives = {4: bitarray('111010011'), 5: bitarray('100011011'), 6:
bitarray('011001011')}

    hypotheses = search_norris(fb)
    hyp2 = hypotheses
    # print('\n'.join(map(str, hypotheses)))
    return hyp2

def test3():  # my test
    fb = FactBase(Fact(0, {'1'}))
    fb.positives = {1: bitarray('11100'), 2: bitarray('10011'), 3: bitarray('11011')}
    fb.negatives = {4: bitarray('10001'), 5: bitarray('01110'), 6: bitarray('01010')}

    hypotheses = search_norris(fb)
    hyp3 = hypotheses
    # print('\n'.join(map(str, hypotheses)))
    return hyp3

if __name__ == '__main__':
    # print('\nStart test 1 :')
    # t1 = test1()
    print('\nStart test "square" :')
    t2 = test2()
    print('Hypothesis:', t2)

    # print('\nanswer: ', isinstance(t2, list))
    # print('\nanswer: ', isinstance(t2[0], list))
    print('end')
    # print('\nStart test 3 :')
    # t3 = test3()
```

Листинг «graph_gen.py»

```
import json

import networkx as nx
from networkx.readwrite import json_graph
import pandas as pd
from jsm.jsm_analysis import test2


_CODE_TMPL = '*GRAPH_TMPL*'
_BR_TMPL = 'br'


def generate_graph(hypothesis, path, name_reas):
    # DEFAULT _draw_edges_1()

    G = nx.Graph()
    scale = 2  # space near node
    count_line_reas = 3  # point in line reas
    smesh_row = 1  # left margin of next row

    if (isinstance(hypothesis[0], list) == True):
        _draw_nodes2(G, hypothesis, scale, 8, name_reas, count_line_reas,
smesh_row)
    else:
        _draw_nodes1(G, hypothesis, scale, 8, name_reas, count_line_reas,
smesh_row)


    d = json_graph.node_link_data(G)
    d['edges'] = d['links']
    d['links'] = []
    s = json.dumps(d)

    _generate_cause_html(path, s)
    print(s)


def _generate_cause_html(path, s):
    tmpl = open('gui/templates/cause_template.html')
    # tmpl = open('templates/cause_template.html') # square
    text = tmpl.read()
    # print(text)
    tmpl.close()
```

```
    text = text.replace(_CODE_TMPL, s)

    dest = open(path, 'w', encoding='utf-8')
    dest.write(text)
    dest.close()


def _draw_nodes2(G, hypothesis, scale, size_node, name_reas, count_line_reas,
smesh_row):
    mas_edge = []
    mas_reas = []
    last_id = 0
    count_node = 0
    x_t, y_t = 1 * scale, 1 * scale

    colors = ['green', 'darkred', 'brown', 'yellow', 'blue', 'orange']
    node_colors = ['darkblue', 'purple', 'gray']
    sel_nod_col = 0  # select color for NODES

    d1 = len(hypothesis)  # count of pairs in hypotheses
    for i in range(d1):
        d2 = len(hypothesis[i])
        for k in range(d2):
            lengmass = len(hypothesis[i][k].value)  # count of reasons
            for j in range(lengmass):
                if hypothesis[i][k].value[j] == True:
                    if mas_reas.count(j) > 0:
                        mas_edge.append(mas_edge[j])
                    else:
                        mas_edge.append(last_id)
                    lab = name_reas[j + 1]
                    # lab = 'test'
                    if mas_reas.count(j) < 1:
                        if (count_node % count_line_reas == 0) & (j > 0):
                            y_t += 2 * scale
                            x_t = smesh_row + 1 * scale
                        G.add_node(last_id, x=x_t, y=y_t, size=size_node, label=lab,
color=node_colors[sel_nod_col])  # add node reas
                        last_id += 1
                        x_t += 1 * scale
                        count_node += 1
                    mas_reas.append(j)
                else:
```

```
                mas_edge.append(None)
            mas_reas.append(None)
        sel_nod_col += 1
        if sel_nod_col > 1:
            sel_nod_col = 0
    num = mas_edge.count(None)
    for i in range(num):
        mas_edge.remove(None)

    for i in range(len(mas_reas)):
        if mas_reas[i] is None:
            mas_edge.insert(i, None)

    _draw_edges_1(G, colors, mas_edge)
    # _draw_edges(G, colors) # next variant of draw grah

def _draw_nodes1(G, hypothesis, scale, size_node, name_reas, count_line_reas,
smesh_row):
    mas_edge = []
    mas_reas = []
    last_id = 0
    count_node = 0
    x_t, y_t = 1 * scale, 1 * scale

    colors = ['green', 'darkred', 'brown', 'yellow', 'blue', 'orange']
    node_colors = ['darkblue', 'purple', 'gray']
    sel_nod_col = 0  # select color for NODES

    d1 = len(hypothesis)  # count of pairs in hypotheses
    for i in range(d1):
        lengmass = len(hypothesis[i].value)  # count of reasons
        for j in range(lengmass):
            if hypothesis[i].value[j] == True:
                if mas_reas.count(j) > 0:
                    mas_edge.append(mas_edge[j])
                else:
                    mas_edge.append(last_id)
                lab = name_reas[j + 1]
                # lab = 'test'
                if mas_reas.count(j) < 1:
                    if (count_node % count_line_reas == 0) & (j > 0):
                        y_t += 2 * scale
                        x_t = smesh_row + 1 * scale
                    G.add_node(last_id, x=x_t, y=y_t, size=size_node, label=lab,
```

```
                        color=node_colors[sel_nod_col])  # add node reas
                    last_id += 1
                    x_t += 1 * scale
                    count_node += 1
                mas_reas.append(j)
            else:
                mas_edge.append(None)
        mas_reas.append(None)
        sel_nod_col += 1
        if sel_nod_col > 1:
            sel_nod_col = 0
    num = mas_edge.count(None)
    for i in range(num):
        mas_edge.remove(None)

    for i in range(len(mas_reas)):
        if mas_reas[i] is None:
            mas_edge.insert(i, None)

    _draw_edges_1(G, colors, mas_edge)
    # _draw_edges(G, colors, mas_edge) # next variant of draw grah

def _draw_edges_1(G, colors, mas_edge):
    # mas_edge = []

    select_color = 0
    source = mas_edge[0]
    target = mas_edge[1]
    for i in range(len(mas_edge) - 1):
        if source is None or target is None:
            source = mas_edge[i]
            target = mas_edge[i + 1]
            select_color += 1
            if select_color >= 6:
                select_color=0
        else:
            G.add_edge(source, target, id=i, color=colors[select_color])
            target = mas_edge[i + 1]

    _change_node_size(G, mas_edge)

def _draw_edges(G, colors, mas_edge):   # posled draw
    # mas_edge = []
```

```
    select_color = 0
    source = mas_edge[0]
    target = mas_edge[1]
    for i in range(len(mas_edge) - 2):

        if source is None or target is None:
            source = mas_edge[i + 1]
            target = mas_edge[i + 2]
            select_color += 1
        else:
            G.add_edge(source, target, id=i, color=colors[select_color])
            source = mas_edge[i + 1]
            target = mas_edge[i + 2]

    _change_node_size(G, mas_edge)

def _change_node_size(G, mas_edge):


    mas_node_size = []

    for i in range(len(mas_edge)):
        if (mas_edge.count(i) > 0):
            mas_node_size.append(mas_edge.count(i))
            if (mas_node_size[i] >= 10):
                mas_node_size[i] = 10
            if ((mas_node_size[i] < 10) and (mas_node_size[i] >= 2)):
                mas_node_size[i] = 9
            if (mas_node_size[i] < 2):
                mas_node_size[i] = 8

    for i in range(len(mas_node_size)):
        G.node[i]['size'] = mas_node_size[i]

    _change_node_color(G, mas_node_size)

def _change_node_color(G, mas_node_size):
    node_color = ['darkred', 'yellow', 'green']
    for i in range(len(mas_node_size)):
        if (mas_node_size[i] == 10):
            G.node[i]['color'] = node_color[0]
        if (mas_node_size[i] == 9):
            G.node[i]['color'] = node_color[1]
        if (mas_node_size[i] == 8):
```

```
            G.node[i]['color'] = node_color[2]


if __name__ == '__main__':

    data = pd.read_csv('../data/ex1.csv', encoding='cp1251', sep=';',
index_col=False, na_values='?')
    name_reas = list(data.columns.values)
    print(name_reas)
    hypothesis = test2()

    path = 'templates/res.html'

    generate_graph(hypothesis, path, name_reas)
```

Выпускная квалификационная работа выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«___» _____ _____ г.

_____ _____

(подпись) *(Ф.И.О.)*