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

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

Кафедра «Прикладная математика и информатика»
(наименование кафедры)
01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)
Системное программирование и компьютерные технологии
(направленность (профиль)/специализация

БАКАЛАВРСКАЯ РАБОТА

на тему **Разработка системы предсказания пожаров в торговых центрах города Тольятти**

Студент В.Д. Ефремов _____
(И.О. Фамилия) (личная подпись)
Руководитель Н.В. Корнеев _____
(И.О. Фамилия) (личная подпись)
Консультанты Н.В. Андрюхина _____
(И.О. Фамилия) (личная подпись)

Допустить к защите

Заведующий кафедрой к.т.н., доцент, А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)
«_____»_____20_____г.

Тольятти 2019

АННОТАЦИЯ

55 с., 21 рисунок, 5 таблиц.

МАШИННОЕ ОБУЧЕНИЕ, ЗАДАЧА ПРОГНОЗИРОВАНИЯ, КЛАССИФИКАЦИЯ, ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ

Проблема пожаров имеет особую значимость во всем мире. Ранняя инспекция здания имеет большое значение для снижения риска возникновения пожара. Для повышения эффективности проведения инспекции зданий на возможные причины возгорания было решено разработать интеллектуальную систему прогнозирования риска возникновения пожара в торговых центрах с использованием технологии интеллектуального анализа данных.

Объектом исследования работы стал процесс прогнозирования пожара в торговых центрах. Предметом исследования является интеллектуальная система для прогнозирования риска возникновения пожара в торговом центре.

В данной бакалаврской работе предложен метод решения поставленной задачи. Решение содержит: персептрон, экспертную систему и классификационную модель, построенную с помощью алгоритма C4.5. Алгоритм написан на объектно-ориентированном языке программирования C#.

Классификационная модель обучалась и тестировалась на базе данных о пожарах, совершенных на территории Российской Федерации, взятых из сети интернет.

Внедрение разработанной системы позволит владельцам торговых центров повысить оперативность и точность предварительной диагностики торговых центров на предмет возгорания.

ABSTRACT

55 p., 21 pictures, 5 tables.

MACHINE LEARNING, PREDICTION, CLASSIFICATION, DATA MINING

The problem of fires is of particular importance worldwide. Early building inspection is essential to reduce the risk of fire. In order to improve the efficiency of building inspection for possible causes of fire, it was decided to develop an intelligent system for predicting the risk of fire in shopping centers using data mining technologies.

The object of the research work was the process of fire forecasting in shopping centers. The subject of the study is an intelligent system for predicting the risk of fire in a shopping center.

In this graduation work the method of solving the problem is proposed. The solution contains: a perceptron, an expert system and a classification model built using the C4.5 algorithm. The algorithm is written in an object-oriented C# programming language.

The classification model was studied and tested on the basis of data on fires in the Russian Federation taken from the Internet.

The implementation of the developed system will allow the owners of shopping centers to improve the efficiency and accuracy of the preliminary diagnosis of shopping centers for fire.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 5
1 Теоретическое обоснование задачи 7
1.1 Постановка задачи 7
1.2 Анализ существующих решений 7
2. Математическая формулировка модели 16
2.2 Общая структура алгоритма решения задачи 29
2.3 Реализация программных модулей 30
3 Анализ и верификация полученных результатов 37
3.1 Проведение вычислительного эксперимента 37
3.2 Корректировка разработанных моделей 47
ЗАКЛЮЧЕНИЕ 51
СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 53

ВВЕДЕНИЕ

Проблема пожаров в России имеет особую значимость, поскольку пожары только в 2010 году согласно данным Института космических исследований РАН проходили на территории 5,8 млн. га. Также на территории России зафиксировано свыше 40 крупных пожаров в местах массового пребывания людей: торгово-развлекательные центры (ТРЦ), торговые центры (ТЦ), бизнес центры (БЦ) за последние 8 лет.

Высокая пожарная нагрузка и отсутствие противопожарных преград при пожаре приводит к массовой гибели людей и крупному материальному ущербу. Согласно статистике, пожары в зданиях, сооружениях и помещениях предприятий торговли в среднем составляют только 2% от общего числа пожаров произошедших, в Российской Федерации. Однако, материальный ущерб от пожаров в ТРЦ достаточно высок, за последние три года он составил 13%, 12% и 5% соответственно от общего ущерба, причиненного государству.

Последний случай возгорания произошел в городе Кемерово в ТРЦ «Зимняя вишня». По официальным данным в результате пожара погибло 64 человека.

Это определяет необходимость создания системы предупреждения и прогнозирования пожаров, позволяющей повысить эффективность мероприятий по профилактике пожаров (подсистема предупреждения пожаров) и борьбе с ними (подсистема противопожарной защиты).

Современные научные методы прогнозирования опасных факторов пожаров основаны на математическом моделирования. Математическая модель пожара описывает в самом общем виде изменение параметров состояния среды в помещении в течение времени, а также изменение параметров состояния ограждающих конструкций этого помещения и различных элементов технологического оборудования.

Научными коллективами как отечественных, так и зарубежных авторов рассматриваются вопросы применения искусственного интеллекта для

решения различных задач, связанных с пожарной безопасностью. Анализ работ показал, что хотя в них и находят применение отдельные методы прогнозирования опасных факторов пожаров, которые работают по следующим принципам:

- математическое моделирование и применении аппарата нейронных сетей.
- интеллектуальные технологии для предупреждения возникновения пожаров.
- учет влияния объемно-планировочных, конструктивных решений и применяемых материалов на обеспечение пожарной безопасности зданий.
- систем мониторинга возникновения и развития пожароопасной ситуации,
- категорирование помещений по взрывопожарной и пожарной опасности,
- модели и алгоритмы поддержки управления комплексной безопасностью объектов.

Задача формирования модели классификации и системы предсказания пожаров торговых центров по прецедентам пожарной безопасности не рассматривалась не в отечественных ни в зарубежных научных публикациях.

1 Теоретическое обоснование задачи

1.1 Постановка задачи

Проблема пожаров имеет особую значимость во всем мире. Ранняя профилактика здания имеет большое значение для снижения риска возникновения пожара. Для повышения эффективности проведения профилактики зданий на возможные причины возгорания было решено разработать интеллектуальную систему прогнозирования риска возникновения пожара в торговых центрах с использованием технологии интеллектуального анализа данных.

Объектом исследования работы стал процесс прогнозирования пожара в торговых центрах. Предметом исследования является интеллектуальная система для прогнозирования риска возникновения пожара в торговом центре.

В данной бакалаврской работе предложен метод решения поставленной задачи посредством персептрона, экспертной системы и классификационной модели, построенной с помощью реализации алгоритма C4.5 на объектно-ориентрованном языке программирования C#.

Классификационная модель обучалась и тестировалась на базе данных о пожарах, совершенных на территории Российской Федерации, взятых из сети интернет.

Внедрение разработанной системы позволит владельцам торговых центров повысить оперативность и точность предварительной диагностики торговых центров на предмет возгорания.

1.2 Анализ существующих решений

На данный момент решений предсказания пожаров не существует, но существуют решения близкие к нашей теме.

Система моделирования пожаров в замкнутых пространствах FireTactics одно из таких решений. Данное программное обеспечение было разработано большой командой, в которой присутствовали теоретики,

практики пожаротушения, а также специалисты из области информационных технологий.

Графическое представление данного программного обеспечения представлено на рисунке 1.1.

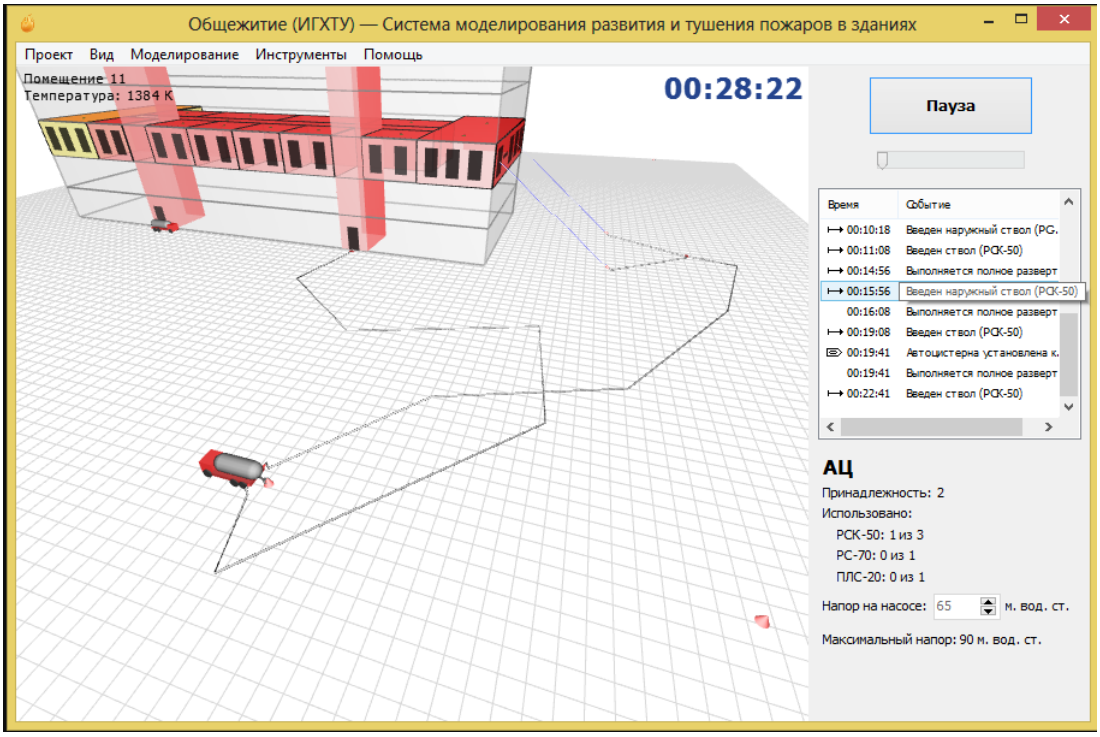

Рисунок 1.1 – Программа FireTactics

Данное программное обеспечение моделирует поведение пожара, его развитие и тушение пожаров в зданиях. Программное средство работает по следующему принципу:

1) Создание проекта здания в виде двумерной планировки. Программа автоматически произведет перевод данной планировки из двумерного в трехмерный вид.

2) Начало расчета динамики параметров развития и тушения пожаров.

3) Выбор количества и качества пожарной техники для ликвидации пожара.

4) В реальном времени производится процесс моделирования пожара.

5) Расстановка техники для тушения пожара.

6) Получение результатов

Программа Urban позволяет моделировать развитие пожара, эвакуацию людей и определять расчетную величину пожарного риска для объектов различных классов функциональной пожарной опасности.

Расчеты выполняются в соответствии с постановлением правительства РФ от 31 марта 2009 г. №272 и приказом МЧС от 30 июня 2009 г. №382.

Моделирование эвакуации. При моделировании эвакуации используется математическая модель индивидуально - поточного движения людей из здания, согласно приложению к приказу МЧС от 30.06.09 г. N 382.

Расчеты осуществляются в соответствии с «Методикой определения расчетных величин пожарного риска в зданиях, сооружениях и строениях различных классов функциональной пожарной опасности (с изменениями на 2 декабря 2015 года)».

Моделирование опасных факторов пожара (ОФП). При моделировании ОФП используется программа FDS, разрабатываемая Национальным институтом стандартов и технологии (НИСТ/NIST) министерством торговли США при содействии Технического научно-исследовательского центра VTT.

FDS моделирует развитие пожара в помещениях по вычислительной гидродинамической модели (CFD) тепломассопереноса при горении.

Моделирование горения осуществляется по всей площади нагрузки, где скорость тепловыделения сначала мала, а затем увеличивается со временем до максимальной.

На рисунке 1.2 представлено главное окно программы с примером открытого проекта.

Рассмотрим содержимое главного окна программы.

А – Меню программы. Выпадающее меню с основными командами программы.

В – Лента приложения. Лента содержит инструменты для работы с топологией здания, эвакуацией, пожарами, отчетами и дополнительные инструменты.

Инструменты сгруппированы по вкладкам в соответствии с назначением конкретного инструмента.

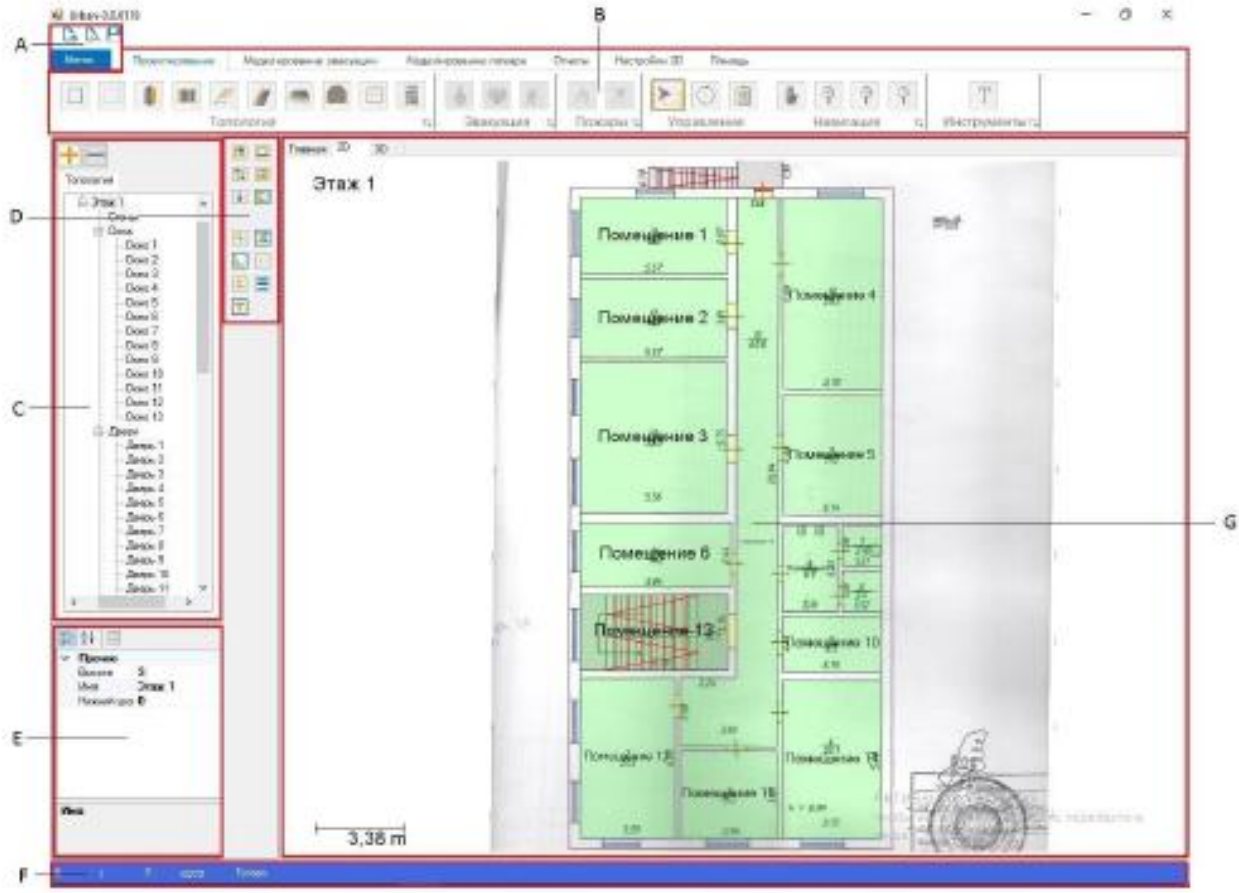

Рисунок 1.2 – Содержимое главного окна программы Urban

C – Дерево проекта. В дереве проекта отображается иерархическая структура текущего здания в проекте со всеми существующими элементами в нем.

При создании сценариев, к дереву добавляются дополнительные вкладки, соответствующие новым сценариям.

Кнопки для создания/удаления сценариев располагаются над деревом проекта.

D – Панель дополнительных инструментов. Панель инструментов содержит вспомогательные инструменты для работы с топологией и подложкой.

E – Окно свойств. Окно свойств предоставляет доступ к свойствам и позволяет настраивать параметры выбранного элемента в дереве проекта или

в редакторе топологии. Для просмотра свойств элемента, выберите его в дереве проекта или в редакторе топологии.

F – Строка состояния. В строке состояния отображаются текущие координаты курсора мыши при работе во вкладке 2D редактора топологии, а также другая вспомогательная информация.

G – Редактор топологии. Редактор топологии представляет собой рабочую область, разделенную на две вкладки – 2D и 3D.

Создание топологии осуществляется на вкладке 2D. Данная вкладка отображает двухмерное представление выбранного в дереве этажа.

При нажатии на кнопке Меню в левом верхнем углу приложения, появляется окно, которое позволяют создать новый проект, открыть уже существующий, сохранить проект или закрыть его, получить доступ к настройкам программы и справочнику материалов.

Расположенные выше выпадающего меню кнопки позволяют получить быстрый доступ к созданию нового проекта, открытию уже существующего или сохранению проекта.

Лента приложения является ключевым элементов управления и содержит основные инструменты для работы с программой и открытым проектом. Лента разбита на шесть вкладок со сгруппированными по типу инструментами.

Список доступных вкладок:

- Вкладка «Проектирование».
- Вкладка «Моделирование эвакуации».
- Вкладка «Моделирование пожара».
- Вкладка «Отчеты».
- Вкладка «Настройка 3D».
- Вкладка «Помощь».

Инструменты вкладки «Проектирование» предназначены для работы с 2D редактором топологии.

Работа с инструментами производится путем выбора необходимого инструмента на вкладке и взаимодействием с редакторами топологии. Если выбранный инструмент отвечает за какой-либо объект, то производится отрисовка соответствующего объекта в редакторе.

Инструменты, не связанные с отрисовкой объектов (Управление, Навигация и другие), позволяют взаимодействовать с редакторами или отрисованными объектами.

Инструменты топологии. Рассмотрим инструменты для создания объектов топологии. Данные инструменты доступны только при работе с вкладкой «Топология» в дереве проекта. При работе со сценариями указанные инструменты становятся недоступными.

Проектирование осуществляется последовательным обозначением левой кнопки мыши вершин проектируемого помещения на рабочей области редактора топологии. Завершение проектирования осуществляется нажатием правой кнопки мыши в редакторе топологии или клавиши «Enter». Отмена проектирования сложного помещения осуществляется нажатием клавиши «Esc».

Инструменты вкладки «Моделирование эвакуации» отвечают за запуск моделирования эвакуации, настройку параметров моделирования эвакуации, а также за управление визуализацией моделирования эвакуации людей из здания, отображаемой в редакторе топологии.

На вкладке «Моделирование пожара» расположены инструменты для запуска моделирования распространения пожара, настройки параметров моделирования пожара и запуска визуализации моделирования пожара. На рисунке 1.3 показано окно расчета модели пожара.

Вкладка «Отчеты» содержит инструменты, которые на основании результатов моделирования эвакуации и пожара, позволяют определить величину индивидуального риска или сгенерировать готовый отчет для рассматриваемых сценариев.

Сценарии в программе «Urban» строятся на основе созданной пользователем топологии здания, но с использованием дополнительного набора объектов. Количество, расположение и значения свойств этих объектов задают текущий сценарий и влияют на конечный результат при расчете ОФП и моделировании эвакуации людей.

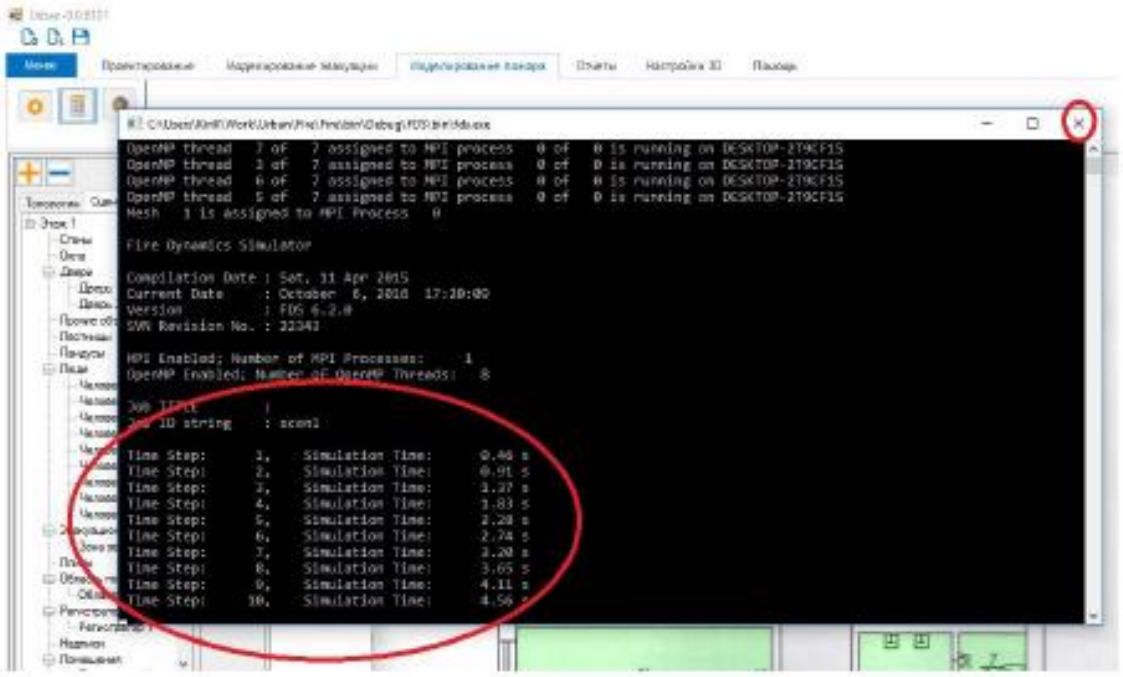

Рисунок 1.3 – Окно расчета модели пожара

Объектами, использующимися исключительно в сценариях программы являются: человек, регистратор, зона эвакуации, область пожара. Зона эвакуации и область пожара являются неотъемлемыми составляющими каждого сценария.

После построения, каждый из этих объектов представляет собой прямоугольную область, закрашенную соответствующим для данного типа объектов цветом, который можно поменять в настройках программы. Зона эвакуации представляет собой конечную точку маршрута людей при их эвакуации. Любой человек в сценарии достигший границ области (вошедший в область) эвакуации считается эвакуированным.

Каждый сценарий может иметь одну или несколько областей эвакуации, расставленных в разных местах. Все они должны быть расположены снаружи здания неподалеку от эвакуационного выхода

(незаблокированной двери, ведущей из помещения или коридора наружу). Каждый, из расставленных в сценарии людей бежит по кратчайшему пути до ближайшей зоны эвакуации.

Как только все расставленные в сценарии люди достигают ближайших зон эвакуации, таймер эвакуации людей останавливается, и эвакуация считается завершенной. При неверном построении топологии проекта, люди, для которых после расчета эвакуации невозможно проложить путь до зоны эвакуации, остаются на месте при проигрывании посчитанной модели эвакуации. Такие люди в расчет времени эвакуации не берутся.

Область пожара представляет собой очаг пожара, необходимый для моделирования распространения пожара по зданию при расчете ОФП. Для более точного расчета область пожара должна быть расположена так, чтобы в сценарии реализовывались наихудшие условия пожара.

В качестве сценариев с наихудшими условиями пожара следует рассматривать сценарии, характеризуемые наиболее затрудненными условиями эвакуации людей и (или) наиболее высокой динамикой нарастания ОФП, а именно пожары: в помещениях, рассчитанных на единовременное присутствие 50 и более человек; в системах помещений, в которых из-за распространения ОФП возможно быстрое блокирование путей эвакуации (коридоров, эвакуационных выходов и т.д.).

При этом очаг пожара выбирается в помещении малого объема вблизи от одного из эвакуационных выходов, либо в помещении с большим количеством горючей нагрузки, характеризующейся высокой скоростью распространения пламени; в помещениях и системах помещений атриумного типа; в системах помещений, в которых из-за недостаточной пропускной способности путей эвакуации возможно возникновение продолжительных скоплений людских потоков.

В помещении, имеющем два и более эвакуационных выхода (незаблокированной двери, ведущей из помещения наружу к зоне эвакуации),

очаг пожара следует размещать вблизи выхода, имеющего наибольшую пропускную способность.

При этом данный выход должен быть блокированным с первых секунд пожара, и при определении расчетного времени эвакуации не учитываться. (Для достижения этого, пользователь должен выбрать данную дверь левой кнопкой мыши и выставить в свойстве «Заблокировано» в панели свойств значение «Да».) В помещении с одним эвакуационным выходом, время блокирования выхода определяется расчетом.

Область пожара имеет свойство «Пожарная нагрузка», влияющее на модель распространения пожара и обязательное к заполнению при расчете ОФП. При наличии области пожара с не заполненным свойством пожарной нагрузки, при попытке расчета ОФП, расчет произведен не будет, а пользователю будет выдано соответствующее уведомление. При необходимости задания собственной пожарной нагрузки или изменения значений существующей, вам необходимо открыть главное меню программы и выбрать пункт «Справочник».

2. Математическая формулировка модели

Информационные технологии развиваются со стремительно нарастающей скоростью. В последние годы все больше внимания фокусируется вокруг данных, с которыми работают IT-специалисты. Придумываются новые методы хранения, сбора, анализа и обработки информации. Создаются новые дизайны программирования компьютерных систем вокруг данных, таких как data oriented design. Такое движение масс вокруг информации происходит не случайно, ведь количество людей, использующих информационные технологии с каждым годом, только увеличивается, а вместе с ними увеличиваются и данные, которые должны хранить информационные системы. Именно подобная ситуация на рынке вызывает спрос в новых и не очень методах для автоматического анализа данных, увеличивающихся ежедневно.

Деревья решений – один из таких методов. Рассмотрим базовые понятия, представленные в таблице 1. [15]

Таблица 1. Понятия, встречающиеся в теории деревьев

Название	Описание
Объект	Пример, шаблон, наблюдение, запись
Атрибут	Признак, независимая переменная, свойство, входное поле
Метка класса	Зависимая переменная, целевая переменная, выходное поле
Узел	Внутренний узел дерева
Лист	Конечный узел дерева, узел решения
Проверка	Условие в узле

Атрибуты – это признаки, описывающие объекты, которые подлежат классификации. Ядром работы деревьев принятия решений является разбиение, обычно рекурсивного, исходного множества объектов на подмножество. Данное подмножество ассоциируется с классами. По заданному условию производится проверка значения атрибутов в контексте правил для разбиения.

Алгоритм работает в стандартном для итеративных алгоритмов режиме, то есть для каждого последующего шага используется результат, который был получен на предыдущем шаге.

Деревья решений – это способ представления правил в иерархической древовидной структуре, который состоит из правил вида «если…то…», и позволяет выполнять классификацию объектов при условии, что каждому объекту соответствует единственный узел, дающий решение как показано на рисунке 2.1.

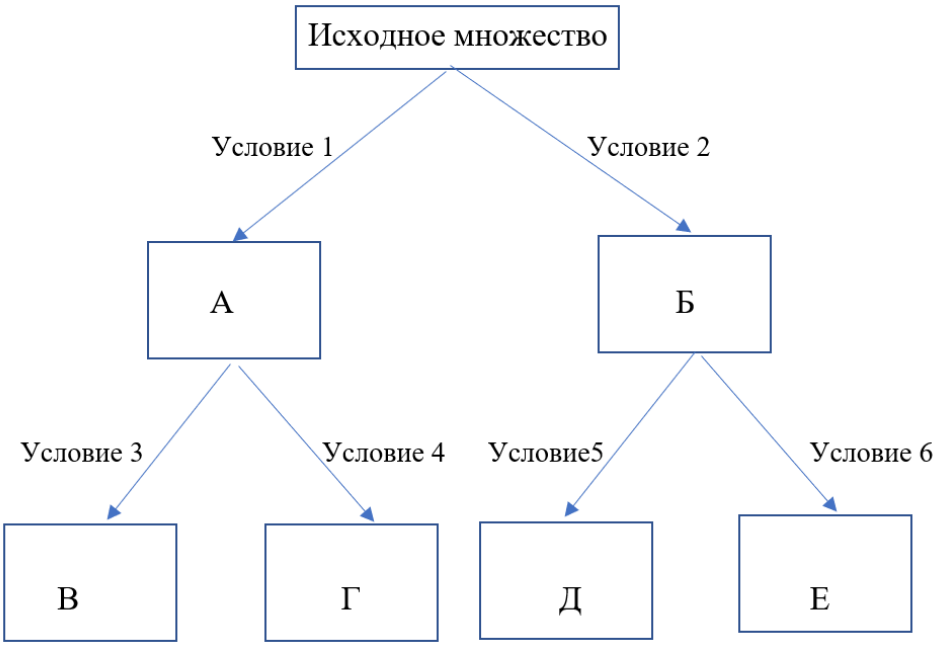

Рисунок 2.1 - Дерево решений

Следует рассказать о структуре дерева решений. В состав деревьев решений входят два вида объектов. Это – узлы и листья. В узлах сформулированы правила. С их помощью производится проверка атрибутов

и множество объектов разбивается на подмножества. Листья – это конечные узлы дерева, содержащие подмножества, ассоциированные с классами.

В дереве, представленном на рисунке 2.1 , объекты А и Б – узлы, а В,Г,Д,Е – листья. Для каждого листа в дереве существует уникальный путь. Первый узел является входным, обычно его называют корневым узлом. Поэтому дерево растет сверху вниз. Узлы и листья, расположенные ниже- дочерние узлы, подчинены узлу более высокого иерархического уровня – родительскому узлу. Обобщенная структура дерева представлена на рисунке 2.2.

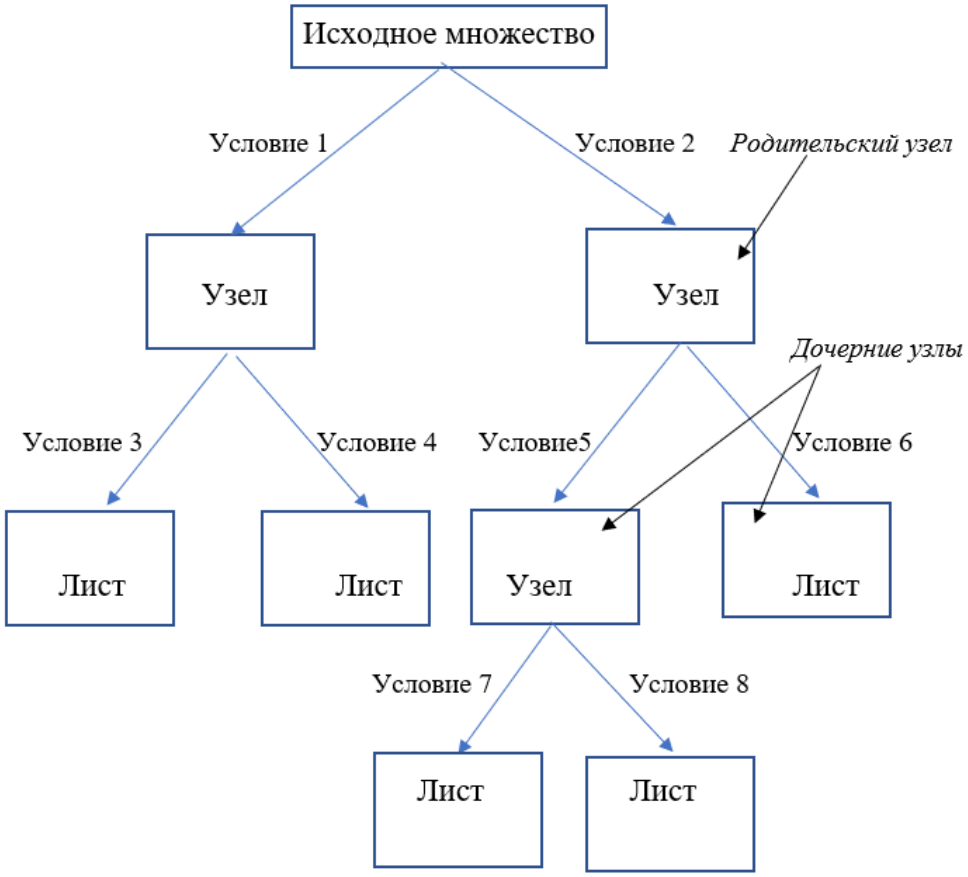

Рисунок 2.2 - Обобщенная структура дерева

Рассмотрим вопрос, связанный с построением дерева решений.

Пусть М- некоторое обучающее множество, содержащее объекты, каждый из которых характеризуется х атрибутами. Следует учесть, что один из них указывает на принадлежность объекта к определенному классу. Пусть

$К_1$, $К_2$, …, $К_n$ – классы, определенные на множестве М. Тогда существует 3 возможных варианта разбиения:

1. Множество М содержит один или более примеров, которые относятся к классу $К_j$. Деревом решений для множества М будет лист, который идентифицирует класс $К_j$. Графически его можно интерпретировать так, как показано на рисунке 2.3.

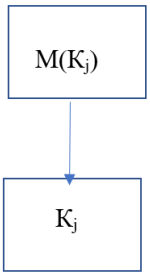

Рисунок 2.3 – Лист дерева

2. Множество М не содержит ни одного примера, а значит является пустым множеством. Тогда дерево решений для множества М будет снова лист. Класс, ассоциированный с листом, выбирается из другого множества, например, родительского.Этот случай проиллюстрирован на рисунке 2.4.

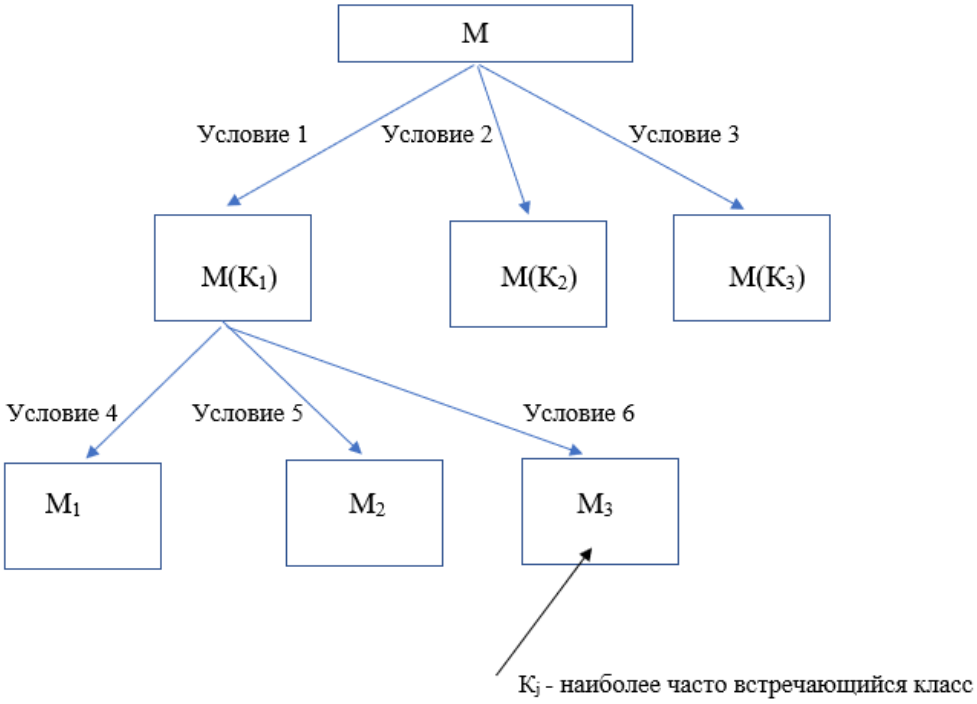

Рисунок 2.4 - Иллюстрация случая 2

В том случае, если в обучающем множестве нет ни одного наблюдения, где M= условие 1 и M($К_1$)=условие 4, алгоритм должен создать три дочерних узла, и в M_3 окажется пустое множество, а будет сгенерирован узел, который станет ассоциироваться с классом, наиболее часто встречающимся в родительском множестве.

3. Множество M содержит примеры, который относятся к разным классам. В этом случае разобьём множество M на некоторые подмножества. Для этого выбирается один из признаков, имеющий два или более отличных друг от друга значений P_1, P_2, …,P_n. Множество M разбивается на подмножества M_1, M_2, …,M_n, где каждое подмножество M_i содержит все примеры , имеющие значение P_i для выбранного признака. Эта процедура будет рекурсивно продолжаться до тех пор, пока конечное множество не будет состоять из примеров, относящихся к одному и тому же классу. Этот случай проиллюстрирован на рисунке 2.5.

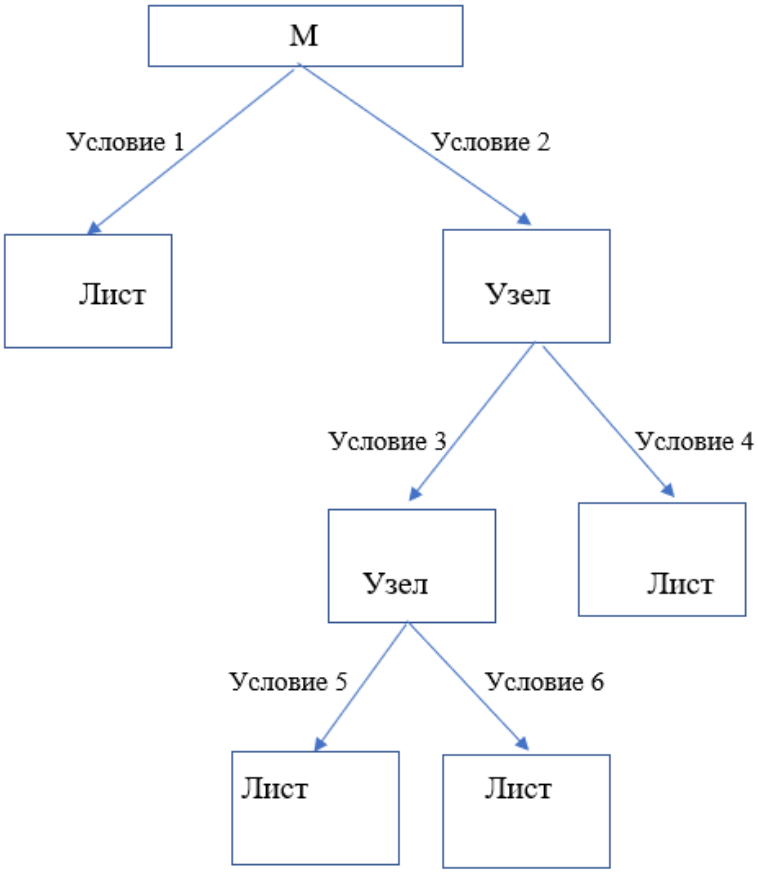

Рисунок 2.5 – Пример принадлежности к одному классу

Пусть на первом шаге алгоритм разбивает исходное множество на два подмножества согласно атрибуту 1. Одно из подмножеств удовлетворяет условию 1 и будет объявлено листом и дальнейшее разбиение в нем остановится. Другое же подмножество подвергается проверке с помощью атрибута 2. Предположим, что оно не удовлетворяет условию 2. Тогда в этом случае происходит дальнейшее разбиение этого подмножества на подмножества, удовлетворяющие и не удовлетворяющие атрибуту 2 и т.д.

Расмотренный процесс лежит в ядре многих современных алгоритмов для построения деревьев принятия решений. На сегодняшний день существует несколько таких алгоритмов: CART, C4.5, NewId, ITrule, CHAID, CN2. Первые два алгоритма получили наибольшую популярность и распространение в анализе данных.

При построении деревьев решений особое внимание уделяется вопросам выбора критерия атрибута, по которому пойдёт разбиение, остановки обучения и отсечения ветвей.

Правило разбиения будет следующее. При построении дерева на каждом внутреннем узле необходимо найти такое условие, которое разбивало бы множество, ассоциированное с этим узлом на подмножества. В качестве такой проверки должен быть выбран один из атрибутов. Были разработаны различные критерии. Рассмотрим теоретико-информационный подход. Для выбора наиболее подходящего атрибута предлагается следующий критерий, называемый приростом информации или уменьшением энтропии:

$$Gain(X)=Info(M) - Info_x(M), \quad (1)$$

где Info(M) – энтропия множества M до разбиения, а $Info_x(M)$-энтропия после разбиения X, причем

$$Info_x(M)= \sum_{i=1}^{n} \frac{M_i}{M} * Info\ M_i\ , \quad (2)$$

Множества M_1, M_2, …, M_n получены при разбиении исходного множества M по проверке X. Выбирается атрибут, дающий максимальное значение по критерию (1).

Правило остановки слудующее. Для решения вопроса о необходимости дальнейшего разбиения узла применяются следующие правила:

- использование статистических методов для оценки целесообразности дальнейшего разбиения, так называемая «ранняя остановка». Этот подход строит менее точные классификационные модели. Авторитеты в этой области советуют следущее: «Вместо остановки используйте отсечение»
- ограничение глубины дерева. Остановка дальнейшего построения в том случае, когда разбиение ведет к дереву с глубиной, превышающей заданное значение.
- разбиение должно быть нетривиальным, то есть получившиеся в результате узлы содержат не менее заданного числа примеров.

В настоящее время не существует такого правила, которое имело бы большую практическую значимость, так как каждое может быть применено в каком-то частном случае.

Правило отсечения следующее. В достаточной мере часто алгоритмы построения деревьев дают сложные решения. Такие решения содержат в себе избыточные данные, большое количество ветвей и узлов. Деревья подобного рода очень трудны для восприятия человеком. Помимо этого деревья, включающие в себя большое количество узлов, разбивают обучающее множество на подмножества с меньшим количеством объектов – это понижает саму ценность такого правила, а также делает его непригодным для анализа. Важно помнить, что чем меньше узлов, обладающих большим количеством объектов из обучающей выборик, там больше значимость такого дерева. Однако стоит понимать, что не стоит давать оценку дереву только по признаку его глубины, подобный метод решения поиска наиболее эффективного дерева решений, попросту не имеет под собой никакого смысла, по крайне мере, на данный момент времени.

Для решения описанной проблемы часто применяется так называемое отсечение ветвей.

Точностью дерева решений называется отношение правильно классифицированных объектов при обучении к общему количеству объектов из обучающего множества, а ошибкой – количество неправильно классифицированных. Если известен способ оценки ошибки дерева, ветвей и листьев, то можно сначала построить дерево, затем - отсечь или заменить поддеревом те ветви, которые не приведут к ошибке.

Если процесс построения происходит сверху вних, то отсечение ветвей происходит наоборот- снизу вверх. Двигаясь, начиная с листьев, отмечая узлы как листья, или заменяя их поддеревом. Отсечение в большинстве практических задач даёт хорошие результаты.

Рассмотрим метод построения деревьев, который используется в алгоритме C4.5. Для работоспособности алгоритма C4.5 необходимо определить требования к структуре данных и к самим данным:

1. Описание атрибутов. Данные, необходимые для работы алгоритма, должны быть представлены в виде плоской таблицы. Вся информация об объектах из предметной области должна описываться в виде конечного набора признаков (атрибутов). Каждый атрибут должен иметь дискретное или числовое значение. Сами атрибуты не должны меняться от примера к примеру, количество атрибутоа должно быть фиксированным для всех объектов.

2. Определенные классы. Каждый пример должен быть ассоциирован с конкретным классом. Другими словами, один из атрибутов должен быть выбран в качестве метки класса.

3. Дискретные классы. Классы должны быть дискретными, т.е. иметь конечное число значений. Каждый пример должен однозначно относиться к конкретному классу. Случаи, когда примеры принадлежат к классу с вероятностными оценками, исключаются. Количество классов должно быть значительно меньше количества примеров.

Алгоритм построения дерева будет следующим. Пусть задано множество объектов M, каждый элемент которого описывается m

атрибутами. Количество объектов в множестве M назовем мощностью этого множества и обозначим M. Пусть метка класса принимает следующие значения: $К_1$, $К_2$,…,$К_s$. Построим иерархическую классификационную модель в виде дерева из множества объектов M. Процесс построения происходит сверху вниз. Сначала создадим корень дерева, затем потомки корня и т.д.

На первом шаге имеется пустое дерево (имеется только корень) и исходное множество M (ассоциированное с корнем). Требуется разбить исходное множество на подмножества. У нас есть m (по числу атрибутов) возможных вариантов, из которых мы должны выбрать самый подходящий.

Возьмем в качестве проверки некоторый атрибут X, который принимает n значений p_1, p_2,…, p_n. В этом случае разбиение M по проверке X даст нам подмножества M_1, M_2, …, M_n при X равном p_1, p_2,…, p_n соответственно. Единственная доступная нам информация- как классы распределены в множестве M и его подмножествах, получаемых при разбиении по X. Это поможет нам при определении критерия.

Пусть freq(K_j, N)-количество объектов из некоторого множества N, относящихся к одному и тому же классу K_j. Тогда вероятность того, что случайно выбранный объект из множества N будет принадлежать к классу K_j:

$$P = \frac{freq(K_j,N)}{N} \quad (3)$$

Согласно теории информации, количество содержащейся в сообщении информации, зависит от ее вероятности:

$$\log_2(\frac{1}{P}) \quad (4)$$

Поскольку мы используем логарифм по основании 2, то выражение (4) дает количественную оценку в битах. Выражение

$$\text{Info(M)= -} \quad \sum_{j=1}^{s} \frac{freq(K_j,M)}{M} * log_2 \frac{freq(K_j,M)}{M} \quad (5)$$

даёт оценку среднего количества информации, необходимого для определения класса объекта из множества M, которое называется энтропией множества M.

Ту же оценку, но только уже после разбиения множества M по X, даёт следующее выражение:

$$\text{Info}_x(M) = -\sum_{i=1}^{n} \frac{M_i}{M} * Info(M_i) \quad (6)$$

Тогда критерием для выбора атрибута будет являться следующая формула:

$$\text{Gain}(X) = \text{Info}(M) - \text{Info}_x(M), \quad (7)$$

Критерий (7) считается для всех атрибутов. Выбирается атрибут, максимизирующий данное выражение. Этот атрибут будет являться проверкой в текущем узле дерева, а затем по этому атрибуту производится дальнейшее построение дерева. Другими словами, в узле будет проверяться значение по этому атрибуту и дальнейшее движение по дереву будет производиться в зависимости от полученного ответа.

Такие же рассуждения можно применить к полученным подмножествам M_1, M_2, …,M_s и рекурсивно продолжить процесс построения дерева до тех пор, пока в узле не окажутся объекты из одного класса.

В том случае, если происходит работа с числовыми атрибутами, следует определить некоторый порог, с которым сравниваются все значения атрибута. Пусть числовой атрибут имеет конечное число значений. Обозначим их через t_1, t_2, …, t_n. Предварительно отсортируем все значения. Тогда любое значение, лежащее между t_i и t_{i+1}, делит все объекты на два множества. В первое входят все, лежащие слева от этого значения t_1, t_2, …, t_i и во второе : лежащие справа от него t_{i+1}, t_{i+2}, …, t_n. В качестве порога выбирается среднее между значениями t_i и t_{i+1}

$$\text{MH}_i = \frac{t_i + t_{i+1}}{2} \quad (8)$$

Таким образом, задача нахождения порога существенно упрощена. Она приведена к рассмотрению всего n-1потенциальных пороговых значений MH_1, MH_2, …, MH_{n-1}.

Формулы (5), (6), (7) последовательно применяются ко всем потенциальным пороговым значениям и среди них выбирается то, которое

даёт максимальное значение по критерию (7). Потом это значение сравнивается со значениями критерия (7), вычисленными для остальных атрибутов. Если выяснится, что среди всех атрибутов данный числовой атрибут имеет максимальное значение по критерию (7), то в качестве проверки выбирается именно он. Нужно отметить, что все числовые тесты являются бинарными, т.е. делят узел дерева на две ветви.

Итак, мы имеем дерево решений и планируем использовать его для распознавания нового объекта. Обход дерева решений начинается с корня дерева. На каждом внутреннем узле проверяется значение объекта Y по атрибуту, который соответствует проверке в данном узле. В зависимости от полученного ответа находится соответствущее ветвление, продолжаем движение к узлу, находящемуся на уровень ниже и т.д. Обход дерева заканчивается как только встретится узел решения, который и даёт название класса объекта Y.

Критерий (7) всегда будет приводить к выбору атрибутов с наибольшим количеством уникальных значений. В этом случае в результате разбиения будут получены множества, которые содержат по одному объекту, образующему один класс, а частота появления этого класса станет равной числу примеров, т.е. 1. Следовательно, в выражении для оценки количества информации появится $\log_2 1=0$, а, поэтому, всё выражение обратится в ноль, т.е. $Info_N(M)=0$. Ясно, что всегда будет обеспечен максимальный прирост информации $Gain(X)=max$, что приведёт к выбору алгоритмом соответствующего атрибута. Если рассмотреть эту ситуацию с точки зрения предсказательных способностей построенной модели, то становится очевидным вся бесполезность такой модели.

Данная проблема решается введением некоторой нормировки.Рассмотрим дополнительный показатель, который представляет собой оценку информации, созданной при разбиении множества M на n подмножеств M_i:

$$\text{Split info (N)} = -\sum_{i=1}^{n} \frac{M_i}{M} * log_2(\frac{M_i}{M}), \quad (9)$$

С помощью этого показателя модифицируется критерий (6) путём перехода к отношению:

$$\text{Split info (N)} = \frac{Gain(M)}{Split\ info(M)} \quad (10)$$

Новый критерий позволяет оценить долю информации, полученной при разбиении, которая является полезной, т.е. способствует улучшению классификации. Использование данного отношения обычно приводит к выбору более удачного атрибута, чем обычный критерий прироста.

Несмотря на улучшение критерия выбора атрибута для разбиения, алгоритм может создавать узлы и листья, содержащие незначительное количество примеров. Чтобы избежать этого, воспользуемся еще один правилом: при разбиении множества M, по крайней мере два подмножества должны иметь не меньше заданного минимального количества примеров k(k>1) ; обычно оно равно 2. В случае невыполнения этого правила, дальнейшее разбиение этого множества прекращается, и соответсвующий узел омечается как лист.

Рассматривая алгоритм построения деревьев, предполагалось, что все значения атрибутов, используемых при разбиении, известны. Но типичной проблемой в реальной ситуации является отсутствие значений атрибутов в отдельных наблюдениях. Первое решение – это исключить все объекты с пропущенными данными. Часто такой выход из сложившейся ситуации неприемлем из-за того, что в наборе данных может присутствовать большое количество записей с пропущенными значениями. Альтернативное решение – создать или модифицировать существующий алгоритм таким образом, чтобы он мог работать с пропусками в данных. В алгоритме C4.5 предполагается, что наблюдения с неизвестными значениями имеют статистическое распределение соответствующего атрибута согласно относительной частоте появления известных значений.

Пусть M – множество обучающих примеров и X-проверка по некоторому атрибуту A. Обозначим через F количество неопределенных значений атрибута A. Изменим формулы (5) и (6) таким образом, чтобы учитывать только те объекты, у которых существуют значения по атрибуту A.

$$\text{Info(M)= -} \quad \sum_{j=1}^{k} \frac{freq(K_j,M)}{M\ -F} * log_2(\frac{freq\ K_j,M}{M\ -F}), \tag{11}$$

$$\text{Info}_x\text{(M)=} \quad \sum_{i=1}^{n} \frac{M_i}{M\ -F} * Info(M_i), \tag{12}$$

В этом случае при подсчете $\text{freq}(K_j, M)$ учитываются только объекты с существующими значениями атрибута A. Тогда критерий (7) можно записать так:

$$\text{Gain (X) =} \frac{M\ -F}{M}(Info\ M\ - Info_x\ M\ ), \tag{13}$$

Подобным образом изменяется и критерий (7). Если проверка имеет n выходных значений, то критерий (7) считается как в случае, когда исходное множество разделено на n+1подмножеств.

Пусть теперь проверка X с выходными значениями P_1, P_2, …,P_n выбрана на основе модифицированного критерия (8). Если пример из множества M с известным выходом P_1 ассоциирован с подмножеством M_i, вероятность того, что объект из множества M_i равна 1. Пусть тогда каждый пример из подмножества M_i имеет вес, указывающий вероятность того, что объект принадлежит M_i. Если пример имеет значение по атрибуту A, тогда вес равен 1, в противном случае объект ассоциируется со всеми множествами M_1, M_2,…, M_n, с соответствующими весами

$$\frac{\text{M}_1}{\text{M}\ -F}, \frac{M_2}{M\ -F}, \ldots, \frac{M_n}{M\ -F}. \tag{14}$$

Легко убедиться, что

$$\sum_{i=1}^{n} \frac{M_i}{M\ -F} = 1 \tag{15}$$

Другими словами, этот подход можно охарактеризовать так: предполагается, что пропущенные значения по атрибуту вероятностно распределены пропорционально частоте появления существующих значений.

Подобная методика применяется и в том случае, когда дерево используется для классификации новых объектов. Если на каком-то узле дерева при выполнении проверки выясняется, что значение соответствующего атрибута классифицируемого примера пропущено, то алгоритм исследует все возможные пути вниз по дереву и определяет, с какой вероятностью объект относится к различным классам. Как только распределение классов установлено, то класс, который имеет наибольшую вероятность появления, выбирается в качестве ответа дерева решений.

2.2 Общая структура алгоритма решения задачи

Структура работы алгоритма для решения задачи может быть поэтапно расписана. Кратко рассмотрим все этапы работы данного алгоритма.

На первом этапе работы алгоритма происходит анализ данных по реальным пожарам. В данных представлены следующие параметры: название торгового центра, площадь здания, этажность, назначение, дата возгорания, причина возгорания, город и площадь возгорания. Далее происходит связывание причины возгорания с атрибутами из нормативных документов. В результате для каждого торгового центра получается массив, состоящий из бинарных значений, то есть ноль или единица. Эти данные используются в персептроне, который обучает веса атрибутов. Вес означает значимость отклонения от нормы для конкретного атрибута, если такое нарушение присутствует. Значения весов передаются далее экспертной системе. Сумма весов складывается и в случае нарушения происходит проверка суммы весов и порога, если сумма превышает порог, это означает, что данное нарушение может привести к пожару, а если сумма не превышает или равна порогу, значит нарушение к пожару не приведет.

Второй этап можно назвать этапом подготовки данных. Были проанализированы данные из нормативных документов, на основе этих данных были составлены правила, регулирующие некоторое здание на соблюдение норм, описанных в ГОСТ. После происходит генерация данных из допустимых значений. Для каждого множества существует сорок

атрибутов, каждый атрибут имеет минимум два значения. Диапазон значений регулируется соответствующим нормативным документом: федеральный закон №123-ФЗ [1]. После того, как 40 атрибутов были сгенерированы, происходит высчитывание прогноза для данного множества. Прогнозирование происходит по правилам алгоритма, он является своего рода экспертной системой, выдающей два прогноза – да или нет.

На третьем этапе происходит построение дерева принятия решений. Здесь используется классификационный алгоритм C45. Дерево призвано ускорить процесс прогнозирования пожара проверяемого множества.

На четвертом идет прогнозирование. Проверяемое множество прогоняется через дерево. В результате получится ответ “Да”, что означает положительное прогнозирование пожара или “Нет” – отрицательный прогноз.

2.3 Реализация программных модулей

Для более полного понимания программы, которую мы разрабатываем следует уделить время на описание программных модулей. В программе основными являются следующие модули: модуль с реализацией экспертной системой и классификационным алгоритмом C4.5, модуль с взаимодействием с Excel, модуль с персептроном. Неосновным можно считать модуль с пользовательским интерфейсом, включающий в себя базовые средства для использования паттерна MVVM.

Программный модуль с реализацией C4.5 состоит из следующих классов:

1. Tree.cs.
2. Leaf.cs.
3. Classifier.cs.
4. Node.cs.
5. Perceptron.cs.
6. Input.cs.
7. IO.cs.

8. Additional.cs.
9. TreeGraph.cs.
10. FileDotEngine.cs.

Tree.cs – это класс, содержащий корень дерева и метод, строящий дерево принятия решений, а также метод для получения результирующего значения дерева по заданному множеству.

Leaf.cs – это класс, описывающий лист дерева. Содержит потомков текущего листа, класс классификатор, родительский узел, класс входящего множества, булевый флаг для обозначения ответа.

Classifier.cs – это класс классификатор, отвечает за разбиение дерева. Содержит список узлов, массив атрибутов и массив ответов. В данном классе происходит нахождение: Entropy, Gain, GainRatio.

Node.cs – это класс узла разбиения. Содержит имя атрибута, количество позитивных и негативных ответов для данного атрибута, сумму количества ответов для атрибута, энтропию данного атрибута.

Input.cs – это класс для работы с входящим множеством. Содержит данные входящего множества, метод взятия колонны атрибутов по номеру атрибута и метод взятия подмножества по атрибуту и его значению.

IO.cs – это класс для работы с вводом в файл и выводом из файла. Содержит функции для работы загрузкой данных из файла, записью данных в файл, экспертной системой

Additional.cs – это класс описывающий дополнительные методы для работы модуля, метод нахождение энтропии.

TreeGraph.cs – это класс, являющийся абстракцией над классом Tree.cs, необходимый для работы дерева с расширяемым классом FileDotEngine.cs.

FileDotEngine.cs – это класс являющийся реализацией dot engine. Содержит единственный метод для записи dot – файла на диск компьютера.

Программный модуль для взаимодействия с Excel API состоит из следующих классов:

1. FireTable.cs.

2. ExcelAPIClass.cs.

FireTable.cs – это класс, который необходим для хранения данных, считываемых из excel таблицы. В данном классе содержится свойство AttributeValues с автоматическими геттером и сеттером. Данное свойство необходимо для работы персептрона, содержит бинарные значения 0 и 1.

ExcelAPIClass.cs – этот класс предоставляет возможность считывания excel документа и запись в него, а также производить сравнение причины пожара с имеющимися атрибутами, взятыми из документа ГОСТ.

Модуль с реализацией персептрона содержит всего один класс - Peceptron. Perceptron.cs – это класс, который является обычным однослойным персептроном, необходим для обучения весов, которые позже используются в экспертной системе для определения значимости отклонений от нормы.

UML-диаграмма выглядит следующим образом как показано на рисунке 2.6.

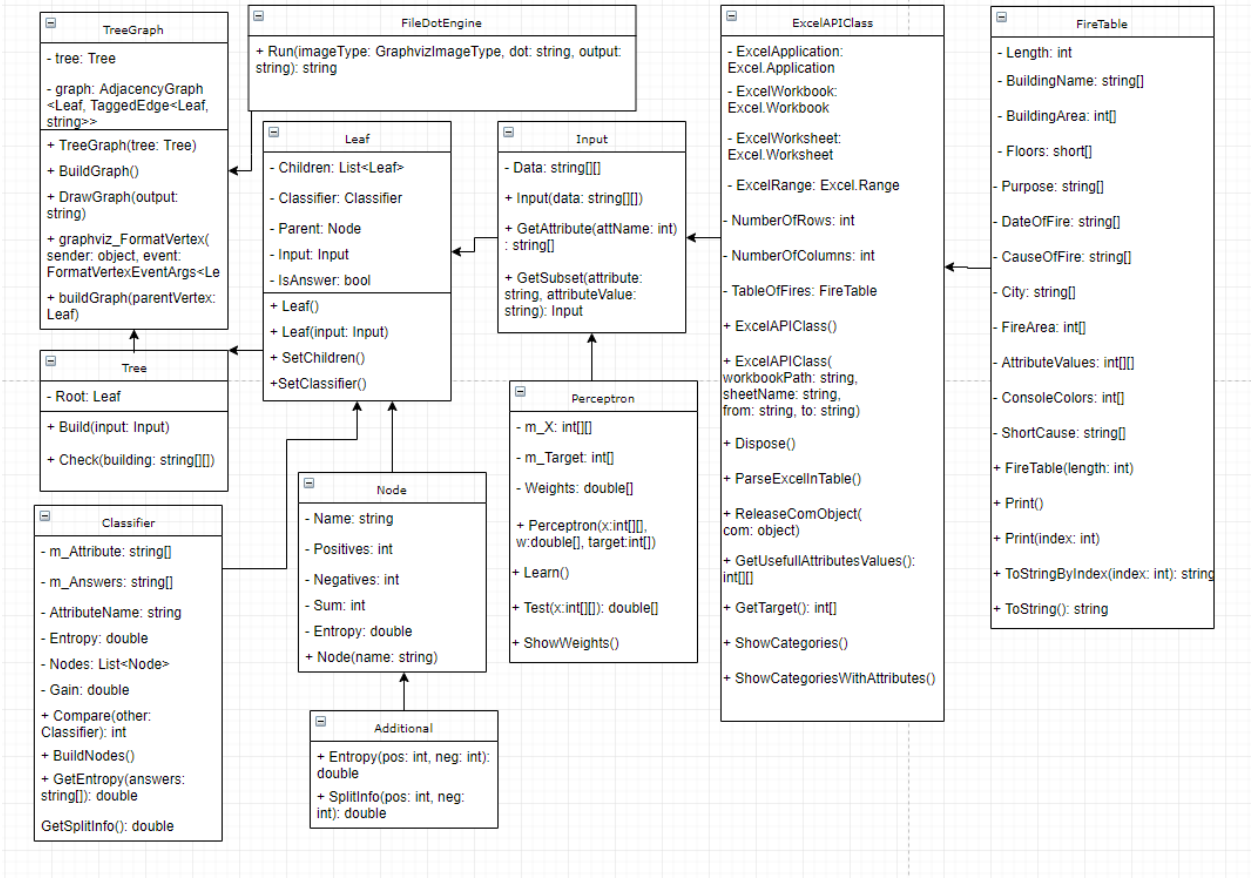

Рисунок 2.6 – UML-диаграмма

Uml-диаграмма, изображенная на рисунке 2.6 дает более точное представление того каким образом устроена реализуемая в данной работе программа.

В программе существуют два разных набора модулей, которые не связаны между собой зависимостями: модуль для генерации обучающих данных, модуль для построения дерева и прогнозирования.

Каждый из данных подмодулей более подробно разбирается ранее в этой подглаве.

В модуль для генерации обучающих данных входят следующие подмодули:

- Perceptron.cs
- ExcelAPI.cs
- Input.cs
- IO.cs

В модуль для построения дерева и прогнозирования входят следующие подмодули:

- Tree.cs
- Leaf.cs
- Classifier.cs
- Node.cs
- Perceptron.cs
- Additional.cs
- TreeGraph.cs
- FileDotEngine.cs

Оба набора используются в модуле с графическим интерфейсом. Так как разработка графического интерфейса не является основной для данной работы, поэтому мы не будем разбирать все элементы графического пользовательского интерфейса, а только основные моменты связанные с архитектурой и с работой программы.

Архитектура приложение строится с использование архитектурного паттерна проектирования MVVM. Основной принцип работы данного паттерна заключается в разделении модели и её представления – это необходимо для программирования каждой части отдельно. Например, один разработчик прописывает логику работы приложения с базой данных или с сервером, а другой разработчик имея должные навыки программирует пользовательский интерфейс.

Шаблон MVVM делится на три части:

- Модель (англ. Model) описывает данные, с которыми работает приложения.
- Представление (англ. View) — графический интерфейс.
- Модель Представления (англ. ViewModel) — содержит в себе модель в преобразованной для представления виде, а также команды для изменения состояния модели.

Можно графически отобразить данный паттерн следующим образом как показано на рисунке 2.2.

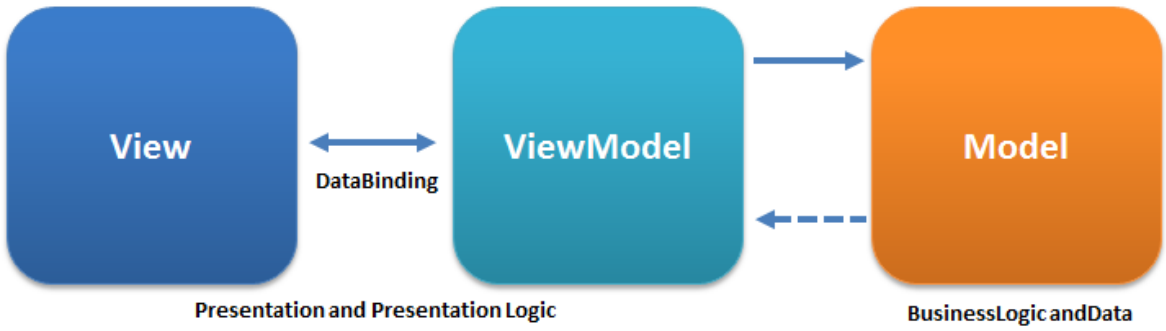

Рисунок 2.7 – Графическое отображение паттерна

Форма для генерации данных представлена на рисунке 2.3.

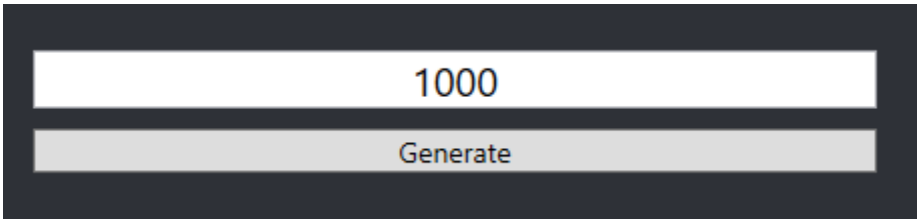

Рисунок 2.8 – Форма генерации данных

В данной форме присутствует только элементы

- textblock.
- button.

Данные из textblock считываются и при нажатии кнопки Generate в обработчике события нажатия кнопки произойдет вся логика с генерацией данных, обучением, загрузкой данных и записью данных в файл.

Форма загрузки данных показана на рисунке 2.4. Данная форма состоит из следующих элементов:

- textblock.
- button.

В текстовый блок вводится путь до файла, вместо использования текстового блока, можно нажать кнопку Search, после чего откроется специальное диалоговое окно операционной системы для поиска файла. Кнопка Load загружает данные из файла.

Рисунок 2.9 – Форма загрузки данных

Фрагмент загруженных данных продемонстрирован в таблице на рисунке 2.5. На данной форме присутствует только один графический элемент DataGridView.

K1	K2	K3	K4	K5	K6	K7	K8
Отсутствует	Отсутствует	Отсутствует	120	<=5	Отсутствует	<500	Над_конько
Отсутствует	Отсутствует	Отсутствует	90	<=5	Присутствует	>=500	Над_конько
Отсутствует	Отсутствует	Отсутствует	90	5<x<=15	Присутствует	>=линии_проведенной_от_конька_вниз_под_углом_10	Над_конько
Отсутствует	Отсутствует	Отсутствует	90	5<x<=15	Присутствует	<500	Над_конько
Отсутствует	Отсутствует	Отсутствует	90	<=5	Присутствует	<500	Над_конько
Отсутствует	Отсутствует	Отсутствует	120	<=5	Присутствует	<500	Над_плоско
Отсутствует	Отсутствует	Отсутствует	90	<=5	Присутствует	<500	Над_конько
Отсутствует	Отсутствует	Отсутствует	90	<=5	Присутствует	<500	Над_конько
Отсутствует	Отсутствует	Отсутствует	90	<=5	Присутствует	<=конька_кровли_или_парапета	Над_конько
Отсутствует	Отсутствует	Отсутствует	90	5<x<=15	Присутствует	<=линии_проведенной_от_конька_вниз_под_углом_10	Над_плоско
Отсутствует	Отсутствует	Присутствует	90	<=5	Присутствует	<500	Над_конько
Отсутствует	Присутствует	Отсутствует	90	<=5	Присутствует	>=линии_проведенной_от_конька_вниз_под_углом_10	Над_конько
Отсутствует	Отсутствует	Отсутствует	90	<=5	Присутствует	>=линии_проведенной_от_конька_вниз_под_углом_10	Над_плоско
Отсутствует	Отсутствует	Отсутствует	90	<=5	Присутствует	>=линии_проведенной_от_конька_вниз_под_углом_10	Над_плоско
Отсутствует	Отсутствует	Отсутствует	90	<=5	Присутствует	>=500	Над_плоско
Отсутствует	Отсутствует	Отсутствует	90	5<x<=15	Присутствует	<500	Над_конько
Отсутствует	Отсутствует	Отсутствует	90	<=5	Присутствует	<=линии_проведенной_от_конька_вниз_под_углом_10	Над_конько
Отсутствует	Отсутствует	Отсутствует	90	<=5	Присутствует	<=линии_проведенной_от_конька_вниз_под_углом_10	Над_плоско

Рисунок 2.10 – Таблица с загруженными данными

Основные элементы управления программой показаны на рисунке 2.6. На форме присутствует следующие графические элементы:

- button.
- label.

В данной форме можно запустить программу, обновить изображение, запустить процесс прогнозирования, найти и загрузить новый файл для программы. Также здесь отображается результат прогноза.

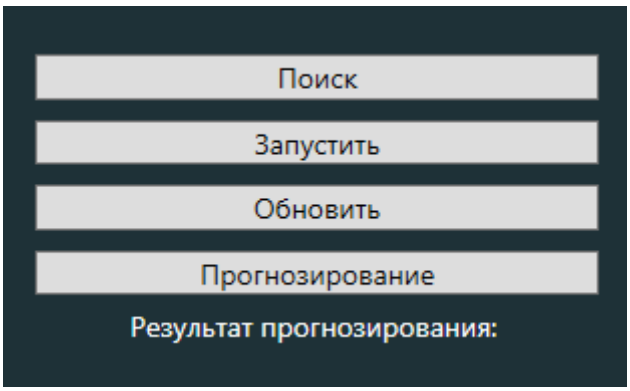

Рисунок 2.11 – Форма для запуска и тестирования программы

Программа написана модульно, поэтому вместо графического интерфейса, можно использовать веб-браузер или вовсе не использовать графику и работать только из-под оболочки.

В данном подразделе перечислены основные моменты, связанные с программой, ей реализацией и использованием.

3 Анализ и верификация полученных результатов

3.1 Проведение вычислительного эксперимента

В конечном результате программа выдает дерево принятие решений, подставив в него множетсво, мы можем получить предсказание касательно случится пожар или нет.

В данном эксперименте мы подставим значения первого множества, представленные в таблице 3.1 в программу.

Таблица 3.1 – TestNo атрибуты

Номер атрибута	Значение
A1	Отсутствует
A2	Отсутствует
A3	Отсутствует
A4	90
A5	5<x<=15
A6	Присутствует
A7	>=500
A8	Над_коньком_кровли_или_парапетом
A9	>3
A10	Из_двух_рядов_кирпичей
A11	Не_защищен
A12	Длительного_горения
A13	350
A14	140x270
A15	>7

A16	Нержавеющая_сталь_заводской_готовности
A17	<60
A18	400
A19	Керамика
A20	Отсутствует

Продолжение таблицы 3.1

Номер атрибута	Значение
A21	>=250
A22	Листовая_сталь
A23	Штукатурка_толщиной_25_мм
A24	По_асбестовому_картону_толщиной_8_мм
A25	Длинная_сторона_вдоль_печи
A26	>=3
A27	B1
A28	Присутствует
A29	Присутствует
A30	Присутствует
A31	Присутствует
A32	Присутствует
A33	Алюминий
A34	16-25

A35	03.фев
A36	>50мм
A37	Водонагревательный_прибор
A38	90
A39	05.апр
A40	ГЛВД

Данное множество выдает отрицательный прогноз, то есть было получено значение «пожара не будет» как показано на рисунке 3.1.

Стоит отметить, что данный ответ не является абсолютно достоверным, поэтому не стоит считать, что отрицательный прогноз полностью гарантирует или в какой-то степени гарантирует невоспламеняемость здания, его общую безопасность. Точность конкретного решения не может на данный момент в полной мере быть оценена, поэтому стоит принимать данный ответ, как приблизительную оценку того, что может случиться.

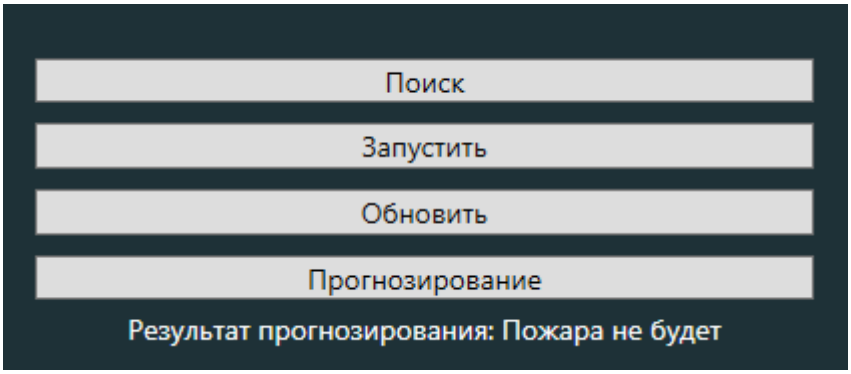

Рисунок 3.1 – Отрицательный прогноз

Далее подставим второе множество представленное в таблице 3.2. Назовем данное множество TestYes, так как мы заранее выбирали множества таким образом, чтобы получилось два ответа из двух возможных.

Таблица 3.2 – TestYes атрибуты

Номер атрибута	Значение

A1	Отсутствует
A2	Отсутствует
A3	Присутствует
A4	90
A5	<=5
A6	Присутствует
A7	<=линии_проведенной_от_конька_вниз_под_углом_10
A8	Над_плоской_кровлей
A9	>3
A10	Из_трех_рядов_кирпичей
A11	Защищен
A12	С_периодической_топкой
A13	1200
A14	140x270
A15	>7
A16	Нержавеющая_сталь_заводской_готовности
A17	>=120

Продолжение таблицы 3.2

Номер атрибута	Значение
A18	400
A19	Кирпич
A20	Отсутствует
A21	130<=x<=250
A22	Штукатурка_толщиной_25_мм

A23	Штукатурка_толщиной_25_мм
A24	По_металлической_сетке
A25	В_пределах_горизонтальной_проекции_печи
A26	>=3
A27	B1
A28	B1
A29	Присутствует
A30	Присутствует
A31	Отсутствует
A32	Присутствует
A33	Алюминий
A34	<4
A35	4
A36	<50мм
A37	Водонагревательный_прибор
A38	90
A39	IP53
A40	ЛЛ

Если в случае отрицательного прогноза, мы не можем говорить о чем-либо с абсолютной уверенностью, то в случае положительного прогноза мы можем утверждать, что проверяемый торговый центр имеет явную необходимость как минимум в дополнительной проверке, а как максмум в полной перестройки всего здания, так как значимость атрибутов варьируется и в общем наборе присутствуют как малозначимые, так и атрибуты, нарушение в которых приведут к возникновению пожара.

Программа выдает положительный прогноз для данного множества как показано на рисунке 3.2.

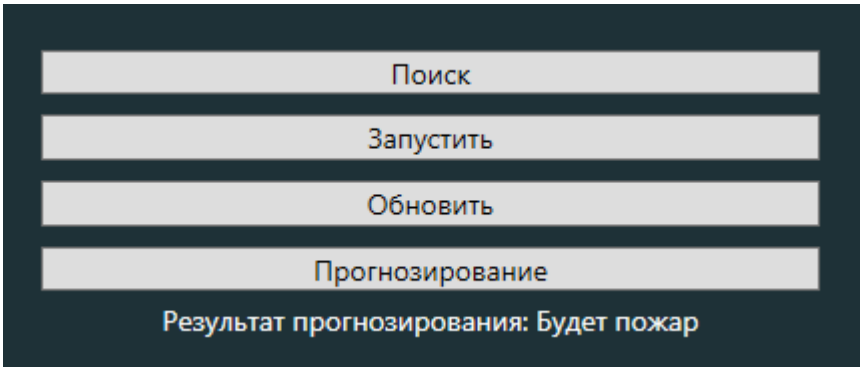

Рисунок 3.2 – Положительный прогноз

Таким образом, у нас есть два множества, значение атрибутов данных множеств, а также два разных прогноза – положительный и отрицательный.

Первое, что нам следует сделать это сравнить два множества и их значения. Больший интерес для нас вызывают значения второго множества, отличающиеся от первого множества. Мы можем сделать такое предположение потому, что именно отличающиеся значения привели к положительному прогнозу.

Несовпадающие атрибуты приведены в таблице 3.3.

Таблица 3.3 – Несовпадающие атрибуты

Атрибут	Значение
A3	Стена из горючих материалов
A5	Площадь печи от размера помещения в процентах
A7	Высота дымовых труб
A8	Расположение дымовой трубы
A11	Потолок из горючих материалов
A12	Тип печи
A13	Расстояние между верхом

	перекрытия печи и потолком

Продолжение таблицы 3.3

Атрибут	Значение
A17	Толщина стенок дымовых труб
A19	Вид материала дымовой трубы
A21	Расстояние от наружных поверхностей дымовых труб до стропил, обрешеток и других деталей кровли
A22	Тип защиты пола из горючих материалов
A24	Нанесение защиты
A25	Расположение защиты
A28	Вытяжные установки для вентиляционного оборудования
A31	Транзитные воздуховоды
A34	Максимальное сечение жилы
A35	Толщина стенки трубы
A36	Длина проводов ответвлений от групповых линий к электроустановочным изделиям и к светильникам

A39	Степень защиты
A40	Тип источников света
A41	Условия среды помещения

В данном наборе атрибутов могут присутствуют лишние – атрибуты, не влияющие на конечный результат. Найдем такие атрибуты, если они действительно присутствуют в данном наборе и исключим их из конечного списка.

Сначала проверим какой из данных атрибутов был последним на пути программы к положительному прогнозу. Запустив программу в отладчике и посмотрев последнее значение и номер атрибута, мы получим атрибут с именем K21 как показано на рисунке 3.3, данный атрибут соответствует 21 атрибуту или атрибуту с названием «Расстояние от наружных поверхностей дымовых труб до стропил, обрешеток и других деталей кровли».

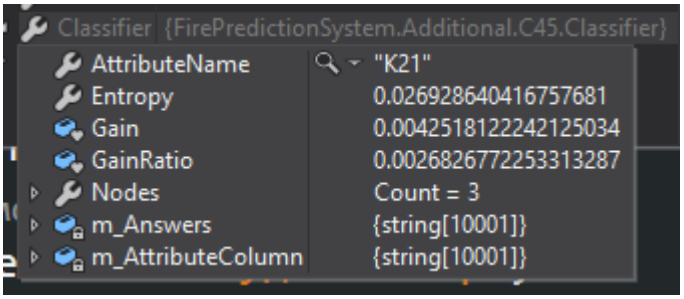

Рисунок 3.3 – Имя атрибута

Значение данного атрибута 130<=x<=250, создадим множество, которое будет содержать все те же значения, что и множество TestYes, заменив только атрибут 21 на значение из множества TestNo, назовем данное множество TestYes2 и проверим его в нашей программе.. Заменим значение и получим результат как показано на рисунке 3.4.

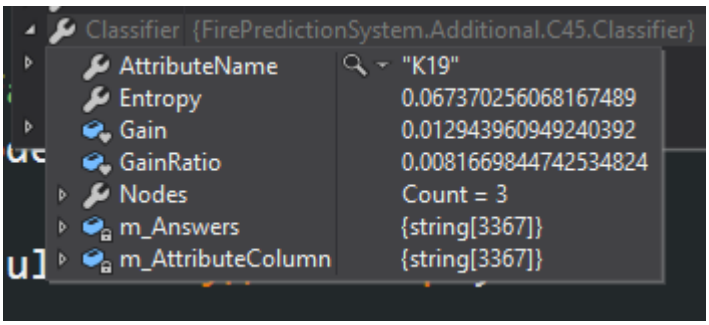

Рисунок 3.4 – Конечный атрибут для множества TestYes2

В данном случае мы ожидаем одно из трех возможных развитий: прогноз изменится – это будет означать, что атрибут был единственным некорректным, прогноз останется неизменным, но конечный атрибут на пути вынесения прогноза поменяется – это означает, что данный атрибут не влиял на результат прогноза, конечный атрибут останется неизменным как и прогноз – это означает, что мы не сможем дальше отбрасывать «лишние» атрибуты.

Ответ программы не изменился, но конечный атрибут поменялся, из-за этого можно сделать вывод, что атрибут 21 не влиял на ответ. Проведем туже операцию для 19 атрибута, назовем множество TestYes3 как показано на рисунке 3.5.

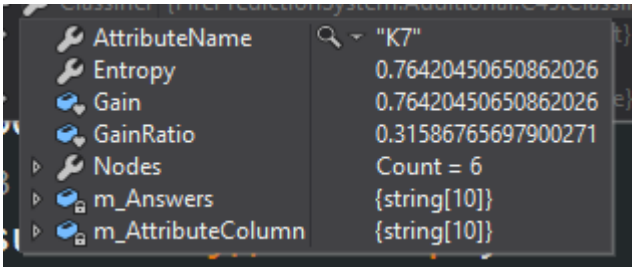

Рисунок 3.5 – Конечный атрибут для множества TestYes3

Ответ снова не изменился, следовательно A19 не влияет на результат прогнозирования. Изменим значение атрибута и назовем множество TestYes4 как показано на рисунке 3.6.

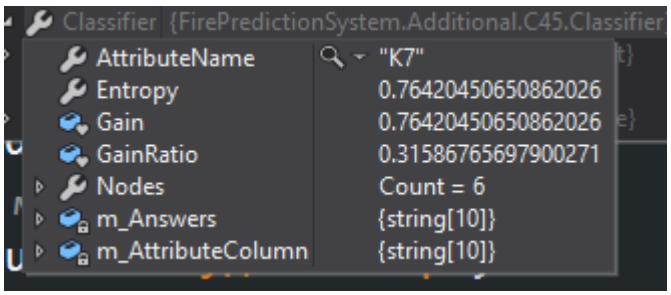

Рисунок 3.6 – Конечный атрибут для множества TestYes3

Как мы видим, программа снова была остановлена на атрибуте «K7» и ответ остался неизменным.

Мы получили атрибут, значение которого неразрывно связано со значениями других атрибутов из данного списка. На данном этапе мы можем считать, что были исключены все атрибуты, не влияющие на результат прогноза. Выведем оставшиеся атрибуты в отдельную таблицу (таблица 3.4)

Таблица 3.4 – Атрибуты, влияющие на результат прогнозирования

Номер атрибута	Название
A3	Стена из горючих материалов
A5	Площадь печи от размера помещения в процентах

Продолжение таблицы 3.4

Номер атрибута	Название
A7	Высота дымовых труб
A8	Расположение дымовой трубы
A11	Потолок из горючих материалов
A12	Тип печи
A13	Расстояние между верхом перекрытия печи и потолком

A17	Толщина стенок дымовых труб
A22	Тип защиты пола из горючих материалов
A24	Нанесение защиты
A25	Расположение защиты
A28	Вытяжные установки для вентиляционного оборудования
A31	Транзитные воздуховоды
A34	Максимальное сечение жилы
A35	Толщина стенки трубы
A36	Длина проводов ответвлений от групповых линий к электроустановочным изделиям и к светильникам
A39	Степень защиты

В данном случае мы получили атрибуты для данного множества, которые напрямую влияют на результирующий прогноз. Стоит сказать, что набор атрибутов для другого множества может полностью отличаться.

Программа выдает набор атрибутов для конкретного множества и в случае несоответствия нормам выдает конкретные слабые стороны, конкретные атрибуты для данного множества, которые в будущем могут вызвать проблемы.

3.2 Корректировка разработанных моделей

Конечным результатом работы алгоритма является дерево принятия решений. Следует рассмотреть дерево при разных количествах входных множеств как показано в таблице 3.5.

Таблица 3.5 – Анализ деревьев при разном количестве входных множеств

	1000	5000	10000	20000	30000	50000	100000	200000
Глубина дерева	4	5	7	8	9	9	12	13
Количество ответов	4	13	29	55	72	143	250	419
Количество листов в дереве	23	51	120	217	283	552	871	1488

Данные и их значения, рассматриваемые в контексте нашей задачи, берутся из нормативных документов. Важным показателем полезности таких данных является соотношение количества ответов к листьям дерева, данный показатель указывает на связанность данных между собой, чем сильнее связанность, тем выше будет данное соотношение в результирующем дереве.

Если мы проведем расчет соотношения, то у нас получатся следующие результаты:

1. Для 1000 входных множеств соотношение составляет 0,1739.
2. Для 5000 входных множеств соотношение составляет 0,2549.
3. Для 10000 входных множеств соотношение составляет 0,2416.
4. Для 20000 входных множеств соотношение составляет 0,2442.
5. Для 30000 входных множеств соотношение составляет 0,2544.
6. Для 50000 входных множеств соотношение составляет 0,2590.
7. Для 100000 входных множеств соотношение составляет 0,2870.
8. Для 200000 входных множеств соотношение составляет 0,2815.

Данный результат является ожидаемым, так как в нормативных документах атрибуты связываются группами и не могут быть все связаны между собой. Пример связанных между собой атрибутов можно увидеть на рисунке 3.3.

№¤	Вид·здания¤	Дополнительные· условия¤	Максимальная· температура·¶ поверхности·печей¶ ¤
1¤	Детские·дошкольные·и· амбулаторно-поликлинические·учреждения¤	-¤	90°С¤
2¤	В·других·зданиях·и· помещениях¤	Площадь·печи··не·более· 15·%¤	110°С¤
3¤	В·других·зданиях·и· помещениях¤	Площадь·печи··не·более· 5·%¤	120°С¤
4¤	Помещения·с·временным· пребыванием·людей¤	При·установке·защитных· эранов¤	Можно·и·выше·120°С¤

Рисунок 3.3 – Пример связанных между собой атрибутов

Атрибуты, приведенные на рисунке 3.3 между собой связаны, но если мы рассмотрим атрибуты из другой таблицы, то мы получим малосвязанные между собой атрибуты.

Важной частью программы является её производительность. В данной реализации существует только три модуля, в которых может возникнуть проблемы с производительностью: модуль загрузки и обработки данных из файла, модуль тестирования входного множества, модуль построения дерева принятия решений. Из трех модулей более ресурсоемким является последний. Нахождение энтропии, разбиение дерева на узлы, нахождение узлов для каждого листа – это все требует значительных вычислительных мощностей рабочей станции, особенно, если входные данные измеряются сотнями мегабайт.

Так как измерение производительности алгоритма не является основной темой данной работы, мы проведем измерения только потенциально самого ресурсоемкого процесса, а точнее модуля построения дерева принятия решений на разном количестве входных данных.

На рисунке 3.7 представлен график зависимости количества затрачиваемых миллисекунд от количества входных данных.

Рисунок 3.7 - Замеры количества времени построения дерева

На рисунке 3.7 отчетливо видно, что данный алгоритм эффективен настолько, что справляется с построением дерева для 50000 множеств, в каждом из которых 40 атрибутов, менее чем за секунду. Тем не менее, данный алгоритм в худшей степени справляется с большими данными. В промежутке от 200000 до 300000 мы можем видеть практически двукратный прирост времени.

Одним из показателей проделанной работы может служить размер исходных файлов. Для данной программы модуль с C45 и персептроном занимает 1618 строк кода, программный модуль с реализацией Excel API занимает 399 строк исходного кода. Модуль Additional занимает 399 строк кода. Несмотря на то, что UI не является основной частью работы, он занимает 1457 строк кода. Всего программа занимает 3610 строк кода.

ЗАКЛЮЧЕНИЕ

В ходе проведенного исследования установлены основные категории причин возникновения пожаров на основе изученных прецедентов. Для создания интеллектуальной системы, реализующей предсказание пожара, были использованы персептрон, экспертная система, классификационная модель на основе алгоритма C4.5. Программная реализация была выполнена в объектно-ориентированной среде .NET на языке программирования C#.

В процессе написания дипломной работы были подробно изучены методы решения задач классификации и выбран наиболее оптимальный из них – метод деревьев. В качестве механизма реализации использован алгоритм C 4.5, который приобрел популярность благодаря понятному представлению и качественному механизму. Для определения значимости отклонений от нормы путем обучения весов в экспертной системе выбран персептрон, как простейшая нейронная сеть.

Для описания причины пожара были сформированы наборы атрибутов, которые позволяют транслировать человеческое описание причины пожара в понятную для компьютера форму, удобную для программного анализа. Также это способствует дальнейшей работе с алгоритмами. Следует сказать, что набор атрибутов не является полным и для данной работы был ограничен. Это необходимо по понятной причине, так как каждый атрибут требует большого количества времени на обработку и включение его в систему. Одним из возможных способов улучшения данной программы может являться увеличение количества атрибутов и увеличение диапазона их значений.

При проведении компьютерного эксперимента было установлено, что ряд атрибутов оказывают большее влияние, чем другие. В работе был изучен один из главных показателей значимости атрибутов, их связность. При работе программы важной характеристикой становится производительность. Если число рассматриваемых входных множеств меньше 5000, то время выполнения основных модулей практически не меняется от изменения

количества входных множеств. При работе с большими данными можно говорить о двукратном приросте времени.

Практическая значимость работы заключается в переносе акцента с исследования распространения пожара и его последствий на изучение предпосылок появления пожара, что привело к созданию уникальной программы предупреждения пожара.

Выполненная работа не является идеальной программой предупреждения пожаров. Для развития и дальнейшего исследования вопроса возможно увеличение количества причин, связанных с требованиями, изложенными в разного рода нормативных документах. В целях совершенствования разработанного программного продукта следует использовать подробные исследования государственных комиссий, связанных с реальными пожарами, позволяющими более детально изучить вопросы значимости тех или иных факторов пожаров.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Закон Российской Федерации "Технический регламент о требованиях пожарной безопасности" от 22 июля 2008 № 123 // Собрание законодательства Российской Федерации. 2008 г. Ст. 12.

2. Ерыгин В.В. Обеспечение пожарной безопасности зданий и сооружений предприятий сервиса / Технико-технологические проблемы сервиса. – 2011. – № 16. – С. 82-88.

3. Калач А.В. Расчет категории помещения на основе методики прогнозирования пожароопасных свойств продуктов нефтепереработки / А.В. Калач, А.С. Крутолапов, Д.С. Королев, Е.В. Калач / Пожаровзрывобезопасность. – 2017. – Том 26. – № 9. – С. 29-34.

4. Кипер А.В. Алгоритмическое обеспечение интеллектуальной поддержки принятия решений, предназначенной для руководителя тушения пожара / А.В. Кипер, Т.С. Станкевич / Пожаровзрывобезопасность. – 2014. – Том 23. – № 9. – С. 45-56.

5. Корнеев Н.В. Концептуальные подходы к оснащению современными системами безопасности предприятий социально-культурного сервиса и туризма / Естественные и технические науки. – 2009. – № 3(41). – С. 447-449.

6. Корнеев Н.В. Категорирование объектов при разработке специального математического и программного обеспечения динамического программирования модели нарушителя антитеррористической и противокриминальной защиты / Н.В. Корнеев, Ю.В. Колесникова / Программная инженерия и информационная безопасность. – 2013. – № 2. – С. 32-40.

7. Корнеев Н.В. Подходы к распознаванию данных из социальных сетей для систем поддержки принятия решений в условиях чрезвычайной ситуации / Н.В. Корнеев, В.А. Гончаров / Информационные технологии. Проблемы и решения. – 2017. – № 1. – С. 162-165.

8. Корнеев Н.В., Гончаров В.А. Модель поддержки управления комплексной безопасностью объектов социальной сферы с массовым

пребыванием людей / Н.В. Корнеев, В.А. Гончаров / Естественные и технические науки. – 2018. – № 4(118). – С. 277-281.

9. Луценко Е.В. Применение автоматизированного системно-когнитивного анализа для прогнозирования рисков при эксплуатации электроустановок в АПК / Научный журнал КубГАУ. – 2015. – № 113(09). – С. 1455-1472.

10. Сазонов С.Ю. Структурно-функциональная организация информационной системы мониторинга возникновения и развития пожароопасной ситуации в дата-центре / С.Ю. Сазонов, Н.А. Ханис / Известия Юго-Западного университета. Серия Управление, вычислительная техника, информатика, медицинское приборостроение. – 2017. – Том 7. – №1(22). – С. 20-26.

11. Свирин И.С. Обзор моделей распространения пожара в зданиях / Проблемы безопасности и чрезвычайных ситуаций. Научный информационный сборник. – 2013. – № 6. – С. 114-129.

12. Яблоков А.В. Заключение общественной комиссии по расследованию причин и последствий природных пожаров в России в 2010 году [Электрон. ресурс] / А.В. Яблоков, Е.Н. Кобец. // Экологический правозащитный центр «Беллона». – Электрон. дан. Режим доступа: http://www.bellona.ru.

13. visualstudio.microsoft [Электронный ресурс] /. — Электрон. текстовые дан. — Режим доступа: https://visualstudio.microsoft.com/ru/vs/getting-started/, свободный

14. C# 6.0. IN A NUTSHELL /Joseph Alahar, Веп Alahar. — 6-е изд. — Beijing: O'REILLY, 2016. — 1040p.

15. Expert .NET 2.0 IL Assembler /Serge Lidin. — 1-е изд. — Beijing: O'REILLY, 2006. — 530p.

16. C# in Depth /Jon Skeet. — 3rd Edition. — NY: Manning Publications Co, 2013. — 614p.

17. CLR via C# /Jeffrey Richter. — 4th Edition. — NY: Microsoft Press, 2012. — 896p.

18. Effective C#. 50 specific ways to improve your C# /Bill Wagner. — 2nd Edition. — New York: Addison-Wesley, 2010. — 343p.

19. Pro WPF is C# 2010: Windows Presentation Foundation in .Net 4 /Matthew MacDonald. — 2nd Edition. — New York: Addison-Wesley, 2010. — 1019p.

20. C# 5.0 Unleashed /Bart De Smet. — 1st Edition. — New York: Sams Publishing, 2013. — 1728p.