Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И
РАДИОЭЛЕКТРОНИКИ (ТУСУР)
Кафедра комплексной информационной безопасности электронно-
вычислительных систем (КИБЭВС)

К ЗАЩИТЕ ДОПУСТИТЬ
заведующий каф. КИБЭВС
д-р техн. наук, проф.
_____А.А. Шелупанов
«____»_____2017г.

ОПРЕДЕЛЕНИЕ АВТОРСТВА ИСХОДНОГО КОДА

Дипломная работа по направлению 10.05.03 –
Информационная безопасность автоматизированных систем
КИБЭВС.58.29.29.001 ПЗ

СОГЛАСОВАНО

Консультант по экономике:
ст. преподаватель каф. КИБЭВС
_____С.В. Глухарева
«____»_____ 2017г.

Студент гр. 722
_____М.В. Мейта
«____»_____ 2017г.

Консультант по безопасности жизнедеятельности:
канд. техн. наук, доцент каф. КИБЭВС
_____Е.М. Давыдова
«____»_____ 2017г.

Руководитель:
канд. техн. наук, доцент каф. БИС
_____ А.С. Романов
«____»_____2017г.

Томск 2017

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС)

УТВЕРЖДАЮ
заведующий каф. КИБЭВС
д-р техн. наук, проф.
_____А.А. Шелупанов
«____»_____2017г.

ЗАДАНИЕ

на выполнение выпускной квалификационной работы

студенту Мейта Марине Валерьевне

группы 722 факультета безопасности

1. Тема работы: Определение авторства исходного кода.

2. Срок сдачи студентом законченной работы: «____»_____2017г.

3. Исходные данные к работе:

– научно-техническая литература по методам машинного обучения, определения авторства текстов, анализу исходного кода;

– программа на языке программирования Python, разработанная в ходе научно-исследовательской работы по аналогичной теме.

4. Содержание расчетно-пояснительной записки:

– обзор существующих исследований, разработок, методов стилометрического анализа исходного кода программ;

– построение модели процесса определения авторства исходного кода;

– разработка программного обеспечения для анализа исходного кода программ с применением стилометрии для определения авторства программного обеспечения;

– подготовка и обработка тестового набора данных;

– исследование эффективности разработанной программы на основе модели процесса определения авторства исходного кода;

– анализ результатов;

– технико-экономическое обоснование;

– вопросы безопасности жизнедеятельности.

5. Перечень графического материала:

– презентация;

– примеры вывода результатов работы программы;

– UML-диаграммы, блок-схемы, графики, описывающие алгоритм работы программы.

6. Консультанты по работе:

Консультант по экономике:
ст. преподаватель каф. КИБЭВС _____С.В. Глухарева
«____»_____ 2017г.

Консультант по безопасности
жизнедеятельности:
канд. техн. наук, _____Е.М. Давыдова
доцент каф. КИБЭВС «____»_____ 2017г.

7. Дата выдачи задания: «____»_____2017г.

Руководитель:
канд. техн. наук, _____ А.С. Романов
доцент каф. БИС «____»_____2017г.

Задание принял к исполнению: _____М.В. Мейта
студент гр. 722 «____»_____ 2017г.

РЕФЕРАТ

Пояснительная записка содержит 87 страницу, 13 рисунков, 19 таблиц, 38 источников, 1 приложение.

СТИЛОМЕТРИЯ, ИСХОДНЫЙ КОД, ДЕАНОНИМИЗАЦИЯ АВТОРА, C/C++, КЛАССИФИКАЦИЯ, PYTHON, SKLEARN, JUPYTER NOTEBOOK, DECISION TREES, RANDOM FOREST CLASSIFIER, КРОСС-ВАЛИДАЦИЯ, ADA BOOST, EXTREMLY RANDOMIZED TREES, GITHUB, LATEX.

Цель работы — разработка программного обеспечения (ПО) для определения авторства исходного кода программ на языке C/C++, основанного на методах стилометрического анализа текста, с перспективой его дальнейшего применения в борьбе с киберпреступностью, в области лицензионных, патентных и иных судебных разбирательств.

В рамках дипломной работы были поставлены следующие задачи:

– обзор существующих исследований, разработок, методов стилометрического анализа исходного кода программ;

– построение модели процесса определения авторства исходного кода;

– разработка программного обеспечения для анализа исходного кода программ с применением стилометрии для определения авторства программного обеспечения;

– подготовка и обработка тестового набора данных;

– исследование эффективности разработанной программы на основе модели процесса определения авторства исходного кода;

– анализ результатов;

– технико-экономическое обоснование работы;

– рассмотрение вопросов безопасности жизнедеятельности.

Объект исследования: деанонимизация автора программного обеспечения.

Предмет исследования: стилометрия исходного кода программ на языках высокого уровня.

Достигнутые результаты: главным результатом дипломной работы являет-

ся программное обеспечение «WhoseCppCode», предназначенное для построения, тестирования и оценки модели классификации авторов исходного кода на языке C/C++, а также визуализации полученных результатов.

Пояснительная записка выполнена при помощи системы компьютерной вёрстки LaTeX.

ABSTRACT

Explanatory note contains 87 pages, 13 pictures, 19 tables, 38 sources, 1 appendix.

STYLOMETRY, SOURCE CODE, AUTHORSHIP ATTRIBUTION, C/C++, CLASSIFICATION, PYTHON, SKLEARN, JUPYTER NOTEBOOK, DECISION TREES, RANDOM FOREST CLASSIFIER, CROSS-VALIDATION, ADA BOOST, EXTREMLY RANDOMIZED TREES, GITHUB, LATEX.

The aim of this work is a software development of the tool for deanonimization of programmers, based on stylometry analysis of source code written in C/C++ programming language for future usage in information security field: copyright/copyleft research, cybercrime investigation and patent licensing.

This specialist work solves the following tasks:

– review of existing research, development, methods of stylometric analysis of source code;

– building a model for the process of determining the authorship of source code;

– software development of the system for source code analysis based on stylometry;

– development of software interface;

– preparation and processing of a test data set;

– study of the effectiveness of the developed program based on the analysis model of source codes;

– analysis of results;

– feasibility study;

– consideration of life safety issues.

Research object: deanonimization of the software developers.

Research subject: source code authorship attribution based on stylometry methods.

Achieved results: the main result of this work is developed software for C/C++ source code authorship attribution called «WhoseCppCode».

Explanatory note is made using a word processor LaTeX.

Содержание

DVD-R диск:
Программный продукт WhoseCppCode
presentation_Meyta.pptx
presentation_Meyta.pdf
Набор тестовых данных
report.pdf
Исходные файлы *.tex

В конверте на обороте обложки

Введение

Задача определения авторства является широко распространенной проблемой в рамках исследования естественных языков, однако в меньшей степени для языков программирования. Тем не менее, с распространением применения компьютерных систем и сетей возросло и количество преступлений в информационной сфере. Существует множество разновидностей кибератак — различные компьютерные вирусы, трояны, несанкционированное копирование данных с кредитных карт, DDoS-атаки и многое другое. Возможность деанонимизации авторов вредоносного программного обеспечения может внести существенный вклад в развитие компьютерной криминалистики.

Считается, что у каждого программиста есть свои специфические профессиональные приемы, привычки, методы написания программного кода, свой так называемый «стиль программирования» и иные признаки, идентифицирующие автора. При этом, как и в случае с естественными языками, на индивидуальный «почерк» программиста может оказывать влияние множество факторов, таких как образование, географическое место проживания, уровень квалификации и другие. «Почерк» также может изменяться с течением времени, развитием технологий и общепринятых норм «хорошего» стиля написания программ. Под «хорошим» стилем обычно понимается набор правил, позволяющих писать код, удобный для чтения, понимания, внедрения дальнейших изменений и рефакторинга. Крупные IT-компании и корпорации обычно вводят свои собственные стандарты кодирования, которые зачастую используются сторонними организациями и индивидуальными программистами в своей работе. Примером могут служить руководства по стилю программирования на языке C++ компаний Google [1] и Geosoft [2].

Определение авторства исходного кода представляет собой актуальную задачу в сфере информационной безопасности, лицензирования в области разработки программного обеспечения, а также может оказать существенную помощь во время

судебных разбирательств, при решении вопросов об интеллектуальной собственности и плагиате.

Целью настоящей дипломной работы является разработка программного обеспечения для определения авторства исходного кода программ на языке C/C++, основанного на методах стилометрического анализа текста.

1 Обзор исследований в области стилометрического анализа исходного кода

На первом этапе выполнения дипломной работы необходимо было провести подробный аналитический обзор информационных источников, рассмотреть существующие методы определения авторства исходного кода и различные подходы к решению такого рода задачи.

В работе [3] представлен набор инструментов и техник, используемых для решения задач анализа авторства исходного кода, а также обзор некоторых наработок в данной предметной области. Кроме того, авторы приводят собственную классификацию проблем и подходов к их решению в рамках задачи деанонимизации авторов программного обеспечения.

Frantzeskou [3] выделяет следующие проблемы (задачи) анализа авторства исходного кода:

– идентификация автора — направлена на определение, принадлежит ли определенный фрагмент кода конкретному автору;

– характеристика автора — базируется на анализе стиля программирования;

– определение плагиата — нахождение схожестей среди множества фрагментов файлов исходного кода;

– определение намерений автора — был ли код изначально вредоносным или стал таковым в следствие программной ошибки;

– дискриминация авторов — определение, был ли код написан одним автором или несколькими.

Подходы к решению вышеперечисленных проблем (задач):

– анализ «вручную» — данный подход включает в себя исследование и анализ фрагмента исходного кода экспертом;

– вычисление схожести — базируется на измерении и сравнении различных метрик или токенов для набора файлов исходного кода;

– статистический анализ — в таком подходе используются статистические

техники, такие как дискриминантный анализ и стилометрия, позволяющие определить различия между авторами;

– машинное обучение — используются методы рассуждения на основе прецедентов и нейронные сети для классификации автора на базе некоторого набора метрик.

В работе [4] предложен способ определения авторства программного обеспечения. в основе которого лежит статистический подсчет метрик, отражающих «почерк создателя» программного обеспечения. На основе метрик составлен «профиль почерка» программистов и вычисляется отклонение от данного профиля для каждого автора. Преимуществом данного метода является его независимость от языков программирования. Метод получил название SCAP (Source Code Author Profiles).

В [5] исходный код транслировался в абстрактные синтаксические деревья, после чего разбивался на функции. Дерево каждой функции принималось за отдельный документ с известным автором. Выборка, состоящая из такого рода деревьев подавалась на вход SVM-классификатору, оперирующему данными типа «дерево». Классификатор обучался на файлах исходного кода двух авторов, в результате чего удалось достичь точности около 67-88%.

В работах [6], [7] и [8] рассматривался способ атрибуции исходного кода с использованием метода N-грамм. Вопрос определения авторства программ в данной работе рассматривался с точки зрения определения плагиата. В качестве выборки использовался набор из 1640 файлов исходного кода, написанных 100 авторами. Позднее удалось улучшить точность классификации данной модели до 77% за счет применения рейтинговых схем.

В [9] применялся алгоритм классификации Random Forest [10] и построение абстрактных синтаксических деревьев. Обучение и тестирование производилось для количества авторов от 250 до 1600. При этом удалось добиться высокой точности — 94-98%. Кроме того, авторы статьи выяснили в ходе работы, что сложнее определить авторов более простых примеров, нежели сложных программ, а также

значительно выделяются авторы с большим опытом программирования на C/C++. В своей дальнейшей работе [11] авторы предложили применение данного подхода для анализа неполных, некомпилируемых образцов кода.

На основании проведенного исследования было решено опробовать подход, основанный на вычислении статистических метрик, характеризующих авторский стиль написания программ, и методов машинного обучения.

Сравнительная таблица с подробным описанием данных информационных источников и используемых в них методов приведена в приложении А.

2 Выбор набора признаков, характеризующих автора программы

Задача стилометрического анализа исходного кода состоит в выделении и статистическом подсчете лексических, синтаксических, структурных и или каких-либо иных признаков на основании обработки текста программы.

Перечисленные в данном разделе признаки, по которым идентифицировались авторы, являются характерными для языков C и C++, однако могут быть использованы для исследования C-подобных языков, например, D, Java, Objective C, C#, PHP, perl и другие.

2.1 Лексические признаки

Главная особенность данной группы признаков состоит в том, что они могут быть вычислены при непосредственном анализе исходного кода программы в виде текстового файла. При этом код программы может быть некомпилируемым, неполным, содержащим синтаксические или программные ошибки.

Лексические признаки, улучшают читаемость кода и включают в себя:

– стиль комментирования (табл. 2.1) — преобладающий в тексте вид комментариев (однострочные или многострочные), а также общее их количество;

– стиль расстановки фигурных скобок (табл. 2.2) [12] — к наиболее известным относят «K&R», «Whitesmith», «One True Bracing Style», стиль Алмена и другие;

– стиль разметки (табл. 2.3) — расстановка пробелов, табуляций, число переносов строки к общей длине файла.

Таблица 2.1 – Признаки, определяющие стиль комментирования

Стиль комментирования		
Признак	Обозначение	Определение
Число однострочных комментариев	ln_inline_ comments	Натуральный логарифм отношения числа однострочных комментариев к длине файла всимволах
Число многострочных комментариев	ln_multiline_ comments	Натуральный логарифм отношения числа многострочных комментариев к длине файла в символах
Число комментариев	ln_comments	Натуральный логарифм отношения числа комментариев к длине файла в символах

Таблица 2.2 – Признаки, определяющие стиль расстановки фигурных скобок

Стиль расстановки фигурных скобок		
Признак	Обозначение	Определение
Число одиночных раскрывающихся скобок	ln_open_ brace_alone	Натуральный логарифм отношения числа раскрывающихся скобок, одиночных в строке, к длине файла в символах
Число раскрывающихся скобок, первых в строке	ln_open_ brace_first	Натуральный логарифм отношения числа раскрывающихся скобок, после которых следует код, к длине файла в символах
Число раскрывающихся скобок, последних в строке	ln_open_ brace_last	Натуральный логарифм отношения числа раскрывающихся скобок, которым предшествует код, к длине файла в символах
Число закрывающихся скобок, одиночных в строке	ln_closing_ brace_alone	Натуральный логарифм отношения числа зарывающихся скобок, одиночных в строке, к длине файла в символах
Число закрывающихся скобок, первых в строке	ln_closing_ brace_first	Натуральный логарифм отношения числа закрывающихся скобок, после которых следует код, к длине файла в символах
Число закрывающихся скобок, последних в строке	ln_closing_ brace_last	Натуральный логарифм отношения числа закрывающихся скобок, которым предшествует код, к длине файла в символах

Таблица 2.3 – Признаки, определяющие стиль разметки

Стиль разметки		
Признак	Обозначение	Определение
Число пробелов	ln_spaces	Натуральный логарифм отношения числа пробелов к длине файла в символах
Число символов табуляции	ln_tabs	Натуральный логарифм отношения числа символов табуляции к длине файла в символах
Число переводов строки	ln_newlines	Натуральный логарифм отношения числа переводов строки к длине файла в символах
Коэффициент пробельных символов	whitespace_ ratio	Натуральный логарифм отношения суммы всех пробельных символов (пробелов, символов табуляции, переводов строки) к длине файла в символах

Дополнительно вычисляются (табл. 2.4):

– число макросов [13] — использует ли программист директивы препроцессора;

– средняя длина строки — позволяет также оценить читаемость кода (слишком длинные программные файлы плохо воспринимаются человеком).

Таблица 2.4 – Дополнительные признаки

Дополнительные признаки		
Признак	Обозначение	Определение
Число макросов	ln_macros	Натуральный логарифм отношения числа макросов к длине файла в символах
Число строк кода	lines_of_code	Число строк кода, не включающее пустые строки

2.2 Ключевые слова C++

Ключевые слова C++ представляют собой список зарезервированных последовательностей символов, используемых языком, недоступных для переопределения.

Для ключевых слов языка C++ вычислялась статистическая мера TF (term frequency), отображающая число вхождения некоторого ключевого слова к общему количеству слов в документе. Подсчет частот ключевых слов может дать представление о предпочтениях автора в определенного рода конструкциях, например, циклов «for» относитльно «while» или «do while», а также об уровне его профессиональной квалификации (определенные конструкции языка C/C++ используются крайне редко, сложны для понимания и предназначены для решения узкоспециализированных задач).

Словарь из 84 ключевых слов C++ (стандарт 11) был взят на сайте с официальной документацией [14] и представлен в таблице 2.5.

Таблица 2.5 – Ключевые слова языка C++ (стандарт 11)

Ключевые слова языка C++					
alignas	char32_t	enum	namespace	return	try
alignof	class	explicit	new	short	typedef
and	compl	export	noexcept	signed	typeid
and_eq	const	extern	not	sizeof	typename
asm	constexpr	false	not_eq	static	union
auto	const_cast	float	nullptr	static_assert	unsigned
bitand	continue	for	operator	static_cast	using
bitor	decltype	friend	or	struct	virtual
bool	default	goto	or_eq	switch	void
break	delete	if	private	template	volatile
case	do	inline	protected	this	wchar_t
catch	double	int	public	thread_local	while
char	dynamic_cast	long	register	throw	xor
char16_t	else	mutable	reinterpret	true	xor_eq

3 Моделирование процесса определения авторства исходного кода

Описание процесса определения авторства исходного кода программ в виде модели «черного ящика» согласно методологии IDEF0 представлено на рисунке 3.1, его декомпозиция — на рисунке 3.2.

Рисунок 3.1 – Модель «черного ящика» процесса определения авторства исходного кода по методологии IDEF0

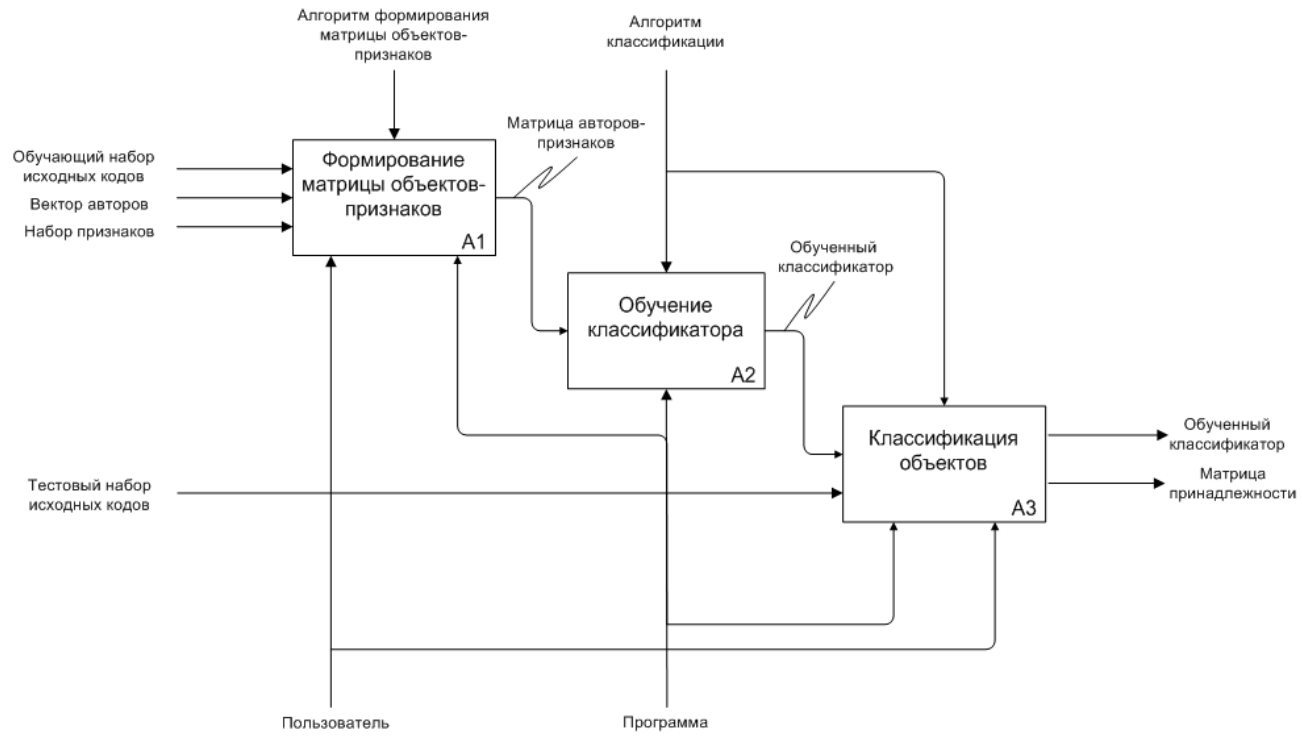

Рисунок 3.2 – Декомпозиция «черного ящика» процесса определения авторства исходного кода по методологии IDEF0

4 Классификация авторов программ

В данной работе в качестве базового алгоритма для всех классификаторов (см. разделы 4.1, 4.2, 4.3) были выбраны деревья решений (Decision Trees), тестирование и оценка модели производилась на основе 10-фолдовой кросс-валидации (см. раздел 5).

Деревья решений (Decision Trees) [15] или деревья принятия решений являются одним из наиболее популярных методов решения задач классификации, регрессии и прогнозирования. Впервые деревья решений были предложены Ховилендом и Хантом (Hoveland, Hunt) в конце 50-х годов прошлого века и в наиболее простом виде представляют собой совокупность правил в иерархической структуре. Основа такой структуры — это ветвление при проверке условий («Да» — «Нет»).

4.1 Алгоритм классификации Random Forest

Алгоритм классификации Random Forest Classifier [10] строится на двух базовых принципах:

– bagging — мета-алгоритм в машинном обучении, при котором на основе большого числа «слабых» классификаторов (в данном случае деревьев решений) строится один «сильный» классификатор (рис. 4.1);

– метод случайных подпространств.

Преимущества данного алгоритма классификации:

– способность эффективно обрабатывать данные с большим числом признаков и классов;

– нечувствительность к масштабированию (к любым монотонным преобразованиям) значений признаков;

– существует методы оценивания значимости отдельных признаков в модели;

– внутренняя оценка способности модели к обобщению (тест out-of-bag);

– высокая параллелизуемость и масштабируемость.

Недостатки алгоритма Random Forest Classifier:

– алгоритм склонен к переобучению на некоторых задачах, особенно на зашумленных, однако для избежания переобучения используется энтропия Шеннона или коэффициент прироста информации (англ. Gain);

– большой размер получаемых моделей приводит к существенным затратам памяти на хранение деревьев, однако данный недостаток решается повышением вычислительных мощностей и распараллеливанием вычислений.

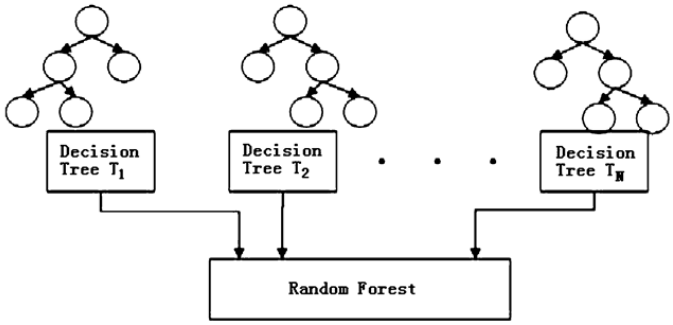

Рисунок 4.1 – Random Forest Classifier

4.2 Алгоритм классификации AdaBoost

Алгоритм AdaBoost (сокр. от adaptive boosting) [16] является мета-алгоритмом, в процессе обучения строит композицию из базовых алгоритмов обучения для улучшения их эффективности.

Достоинства:

– хорошая обобщающая способность — в реальных задачах (не всегда, но часто) удаётся строить композиции, превосходящие по качеству базовые алгоритмы, при этом обобщающая способность может улучшаться (в некоторых задачах) по мере увеличения числа базовых алгоритмов;

– простота реализации;

– время построения композиции практически полностью определяется временем обучения базовых алгоритмов.

Недостатки алгоритма классификаций AdaBoost:

– склонен к переобучению при наличии значительного уровня шума в данных;

– требует достаточно длинных обучающих выборок;

– бустинг может приводить к построению громоздких композиций, состоящих из сотен алгоритмов, такие композиции исключают возможность содержательной интерпретации, требуют больших объёмов памяти для хранения базовых алгоритмов и существенных затрат времени на вычисление классификаций.

4.3 Алгоритм классификации ExtraTrees

Алгоритм ExtraTrees (Extremly Randomized Trees) [17] является модификацией алгоритма Random Forest Classifier (см. раздел 4.1), но отличается еще более рандомизированным разделением входного набора данных на подвыборки. Как правило, результаты работы данного алгоритма схожи с результатами Random Forest Classifier, однако в определенных случаях могут давать улучшение точности классификации.

5 Тестирование инструмента построения аналитической модели

Для тестирования аналитической модели в машинном обучении применяется процедура скользящего контроля, получившая название кросс-валидации (cross-validation) или перекрестной проверки.

Процедура кросс-валидации [18] включает в себя случайное разбиение на k подгрупп (или фолдов) примерно одинакового размера. Первый фолд служит для тестирования модели, остальные используются для обучения классификатора. Для тестовой подвыборки вычисляется среднеквадратичное отклонение. Процедура повторяется k-1 раз, при этом каждая из подгрупп выступает в роли тестовой выборки.

В данной работе тестирование производилось с применением 10-фолдовой кросс-валидации. Всего было произведено 10 вычислительных экспериментов.

6 Описание тестового набора данных

Burrows в работе [7] выделяет следующие ключевые параметры тестовых данных, которые могут влиять на точность классификации:

– число авторов — с увеличением числа авторов сложность классификации увеличивается, точность — снижается;

– число экземпляров выборки для каждого автора — желательно соблюдать одинаковым для всех авторов во избежание отклонения в сторону наиболее точно описанных авторов, а также иметь больше экземпляров для увеличения размера тестовой выборки;

– средняя длина образца кода (количество непустых строк кода) — чем длиннее, тем выше точность классификации, изменение длины экземпляров выборки может влиять на отклонение в сторону наиболее точно описанных авторов, однако не представляется возможным соблюдать длину экземпляра выборки постоянной;

– «стилистическая зрелость» (stylistic maturity) авторов — уровень квалификации, личные и профессиональные предпочтения в стиле написания программ;

– временные метки образцов кода — подразумевается, что со временем программы устаревают, технологии и методы программирования меняются и, как следствие, изменяется стиль программирования;

– репрезентативность выборки — демографические, социальные и другие факторы;

– типы авторов — студент, фрилансер, профессиональный разработчик, в идеале система должна включать в себя разные типы;

– языки программирования — если тестировать несколько языков одновременно, результат будет зависеть от характерных признаков языка;

– авторство в одном лице — большинство проектов выполняются в сотрудничестве с другими разработчиками;

– корректное авторство — без плагиата, копирования и т.п.

Burrows упоминает также от том, что характерный стиль программирования нестабилен в начале карьеры программиста, что может существенно отличать начинающего специалиста и профессионала разработки.

Программное обеспечение «WhoseCppCode» тестировалось на трех наборах данных:

1) «students» — выборка представляет собой работы студентов первого курса обучения по предмету «Основы программирования»; все программы реализуют решения однотипных задач в рамках учебной дисциплины, что исключет их разделение при классификации по функциональному назначению вместо стилистических особенностей и снижение точности классификации;

2) «Google Code Jam» — общедоступные данные ежегодной международной олимпиады по программированию Google Code Jam 2016 [19]; так же, как и в первой выборке, авторы решали схожие задачи, используя различные подходы и алгоритмы;

3) «GitHub» — данные, собранные с сайта GitHub [20], крупнейшего [21] веб-сервиса для хостинга IT-проектов и их совместной разработки.

Сбор данных с веб-хостинга GitHub производился по следующему принципу:

1) выбирались крупные open-source репозитории, посвященные разработке проектов на C/C++;

2) просматривался список контрибьюторов;

3) в качестве авторов выбирались те контрибьюторы, у которых имеются личные проекты, написанные на C/C++;

4) на основе списка пользователей автоматически, средствами программы «WhoseCppCode», производился сбор и сохранение файлов исходного кода для каждого автора.

В таблице 6.1 приводится описание некоторых характеристик каждого набора данных. В данном случае под смешанным типом авторов подразумевается, что разработчики могли быть совершенно разного уровня квалификации и рода деятельности (студенты, фрилансеры, начинающие и профессиональные разработчики,

программисты-любители и т.д.).

Таблица 6.1 – Тестовые данные

Набор данных	«Students»	«Google Code Jam»	«GitHub»
Число авторов	3	30	30
исло файлов исходного кода на одного автора	14	9	78
Всего файлов исходного кода	42	278	2334
Минимальное число строк кода	33	36	26
Максимальное число строк кода	160	461	16348
Среднее число строк кода на один файл исходного кода	45	87	234
Тип авторов	Студенты	Смешанный	Смешанный

Каждая выборка представляет собой совокупность файлов исходного кода программ на языке C/C++ с расширениями *.cpp, *.c, *.h, *.hpp, *.cxx, *.cc, *.ii, *.ixx, *.ipp, *.inl, *.txx, *.tpp, *.tpl.

7 Критерии оценки эффективности инструмента классификации

Критерии оценки работы классификатора [22] представлены в таблице 7.1, где:

– tp — истинно-положительное решение;

– tn — истинно-отрицательное решение;

– fp — ложно-положительное решение;

– fn — ложно-отрицательное решение;

– accuracy (точность) — отношение количества документов, по которым классификатор принял правильное решение, к общему числу документов (примеров файлов исходного кода);

– precision (правильность) — доля документов, действительно принадлежащих данному классу, относительно всех документов, которые система отнесла к этому классу;

– recall (полнота) — доля найденных классфикатором документов, принадлежащих классу, относительно всех документов этого класса в тестовой выборке;

– f1-score (f1-мера) — гармоническое среднее между правильностью и полнотой.

Таблица 7.1 – Критерии оценки работы классификатора

Критерий	Формула	Луч. знач.	Худ. знач.
Accuracy (точность)	(tp + tn) / число примеров * 100 %	100 %	0 %
Precision (правильность)	tp / (tp + fp)	1	0
Recall (полнота)	tp / (tp + fn)	1	0
F1-score (F1-мера)	2 * (precision * recall) / (precision + recall)	1	0

8 Результаты классификации авторов программ

При тестировании классификатора использовались критерии оценки, описанные в разделе 7, а также время работы программы. Результаты работы классификатора представлены в таблице 8.1, а также на рисунках 8.1, 8.2 и 8.3.

Таблица 8.1 – Результаты работы

Набор данных «Students»					
Классификатор	Accuracy, %	Precision	Recall	F1-score	Время работы, сек.
Random Forest	89,55	0,90	0,93	0,90	93,61
AdaBoost	70,45	0,70	0,74	0,70	53,22
ExtraTrees	91,85	0,92	0,95	0,92	53,70
Набор данных «Google Code Jam»					
Классификатор	Accuracy, %	Precision	Recall	F1-score	Время работы, сек.
Random Forest	86,66	0,86	0,88	0,87	110,65
AdaBoost	19,43	0,16	0,16	0,19	103,53
ExtraTrees	88,09	0,88	0,90	0,88	60,34
Набор данных «GitHub»					
Классификатор	Accuracy, %	Precision	Recall	F1-score	Время работы, сек.
Random Forest	69,92	0,69	0,71	0,70	223,77
AdaBoost	16,44	0,09	0,11	0,16	451,63
ExtraTrees	70,99	0,70	0,72	0,71	201,13

Наихудшие результаты показал алгоритм AdaBoost, в то время как наиболее точным и быстрым из трех представленных алгоритмов оказался ExtraTrees (см. раздел 4.3).

С использованием метода, основанного на извлечении лексических признаков

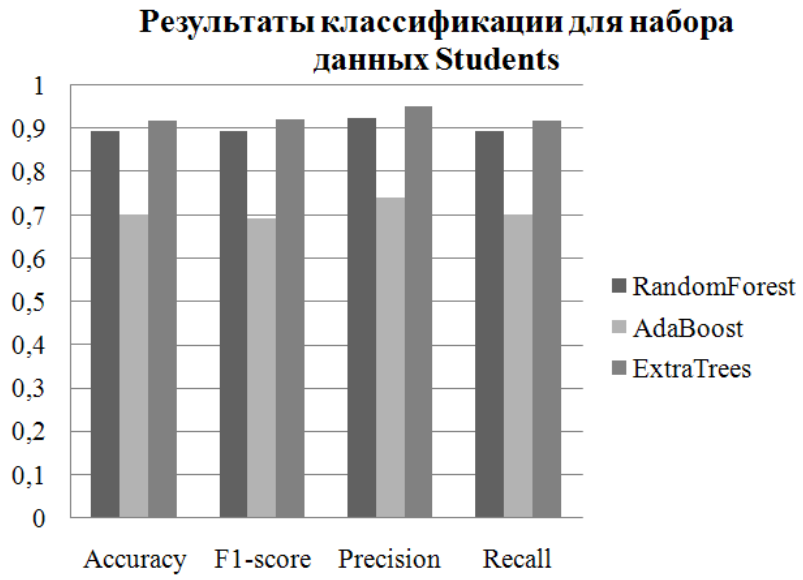

Рисунок 8.1 – «Students»

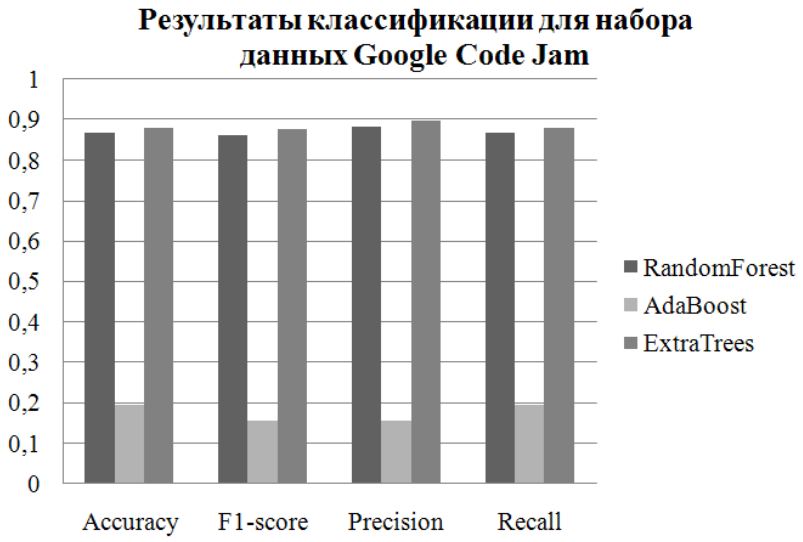

Рисунок 8.2 – «Google Code Jam»

и классификации с помощью алгоритма ExtraTrees (Extremly Randomized Trees) точность классификации составила 70-71% на выборке данных из 30 авторов и 2334 неполных, немопилируемых файлов с веб-хостинга GitHub.

По результатам классификации можно сделать следующие выводы:

1) заменив алгоритм классификации RandomForest на его модифицированную версию, ExtraTrees (Extremly Randomized Trees), можно повысить точность класси-

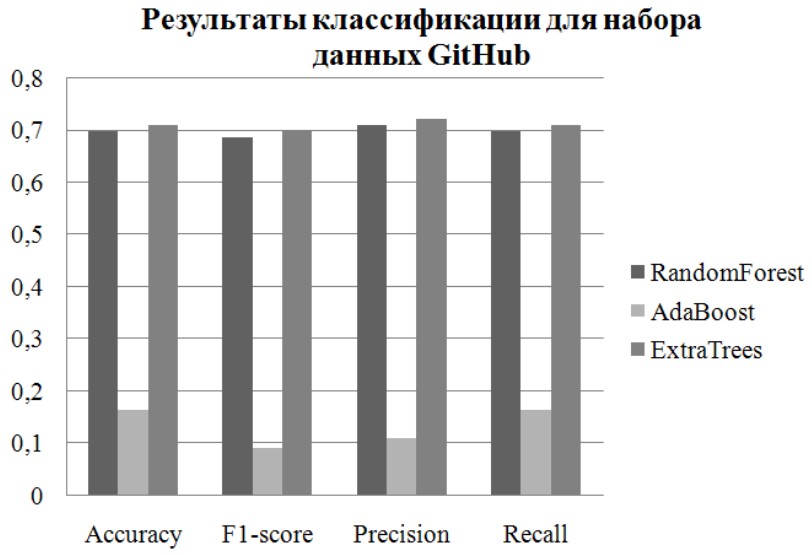

Рисунок 8.3 – «GitHub»

фикации, достигнутую в работе [11], что в итоге, при воссоздании эксперимента коллег, даст наилучший результат классификации авторов исходного кода на языке C/C++ на сегодняшний день;

2) исследованные методы могут применяться не только в «лабораторных» условиях, когда тестовая выборка генерируется на основе студенческих работ или результатов олимпиад по программированию, где решаются схожие задачи, ограниченные по времени и объему кода, но и при решении реальных практических задач.

9 Описание программного обеспечения «WhoseCppCode»

Программное обеспечение «WhoseCppCode» состоит из двух основных частей:

– программного модуля, реализующего все необходимые функции для сбора, анализа и обработки данных, а также построения модели классификации авторов исходного кода, описанной в разделе 3;

– программного интерфеса на основе технологии Jupyter Notebook [23], предназначенного для визуализации полученных в ходе классификации результатов, сбора необходимых данных с ресурса GitHub [20], построения матрицы объектов-признаков на основе входных данных, проведения вычислительных экспериментов.

Программный модуль реализован на языке прораммирования высокого уровня Python с использованием следующих программных библиотек:

– Scikit-Learn [24] — open-source библиотека для машинного обучения: классификации, регрессии, кластеризации и т.д.

– Plotly [25] — графическая Python-библиотека для построения интерактивных графиков, таблиц, диаграмм.

– Numpy [26] — библиотека для научных вычислений, предоставляющая методы работы с большими массивами данных.

– Scipy [27] — предоставляет среду для проведения математических и научных вычислений.

– Pandas [28] — open-source библиотека, предназначенная для анализа данных.

– Ipywidgets [29] — интерактивные HTML виджеты для Jupyter Notebook.

Интерфейс основан на веб-технологиях, может использоваться для демонстрации возможностей программ на языке Python. Библиотека Jupyter Notebook, с помощью которой был реализован данный интерфейс, была выбрана за счет ряда преимуществ:

– является свободным ПО;

– поддерживает множество языков программирования;

– позволяет хранить вместе код, изображения, комментарии, формулы и графики;

– не требует знаний и применения веб-технологий, таких как CSS, HTML, JavaScript;

– может быть запущен на любом сервере, необходим только доступ по ssh/http;

– позволяет экспортировать код и сам блокнот в любом формате;

– предназначена для демонстрации разработок на языке Python (в основном в машинном обучении).

Основной модуль программы «WhoseCppCode» может быть использован отдельно от Jupyter Notebook при разработке различного рода программ, систем и интерфейсов лицами, заинтересованными в задаче классификации программистов.

Диаграмма действий в нотации UML, описывающая основной алгоритм работы программы «WhoseCppCode» представлена на рисунке 9.1, вид интерфейса программы — на рисунках 9.2, 9.3, 9.4 примеры ввода и вывода данных в интерфейсе Jupyter Notebook — на рисунках 9.5 и 9.6.

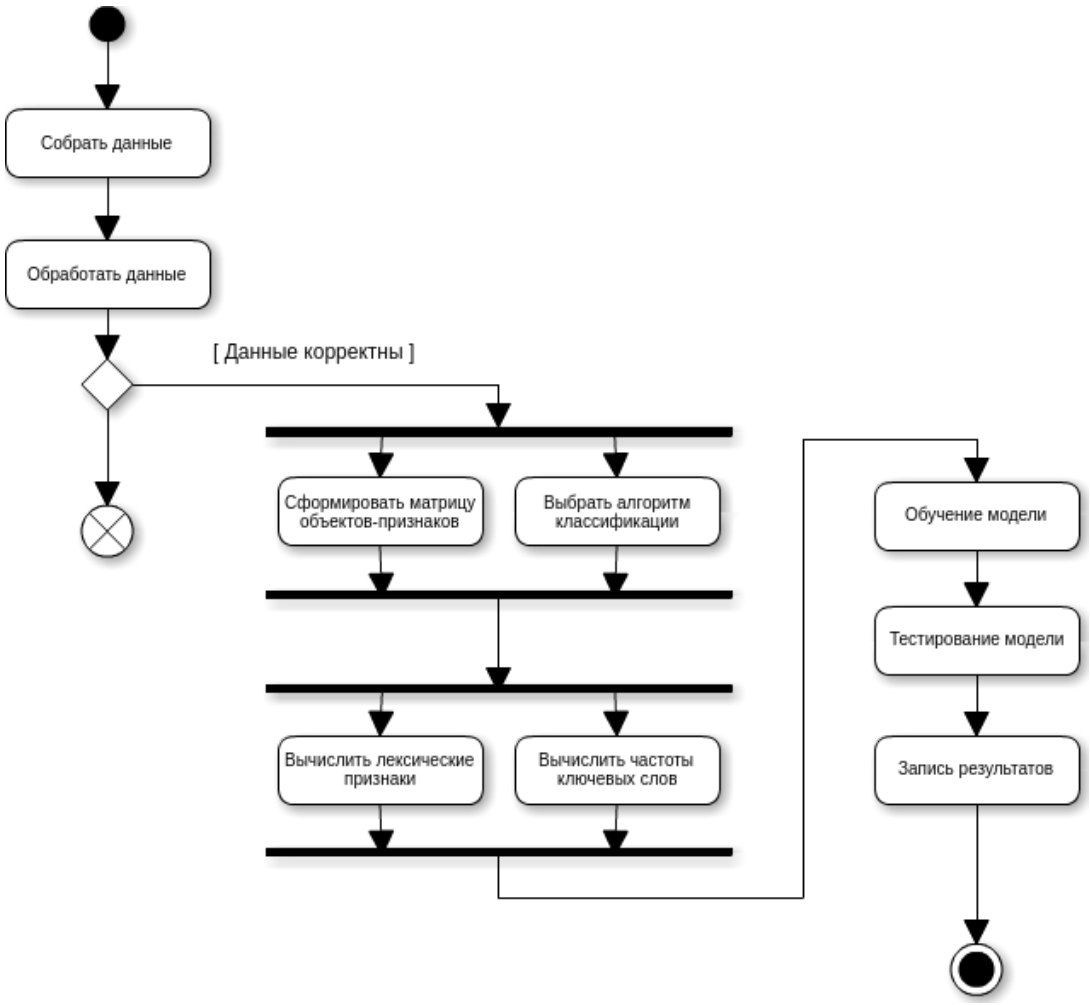

Рисунок 9.1 – Диаграмма действий алгоритма работы программы «WhoseCppCode»

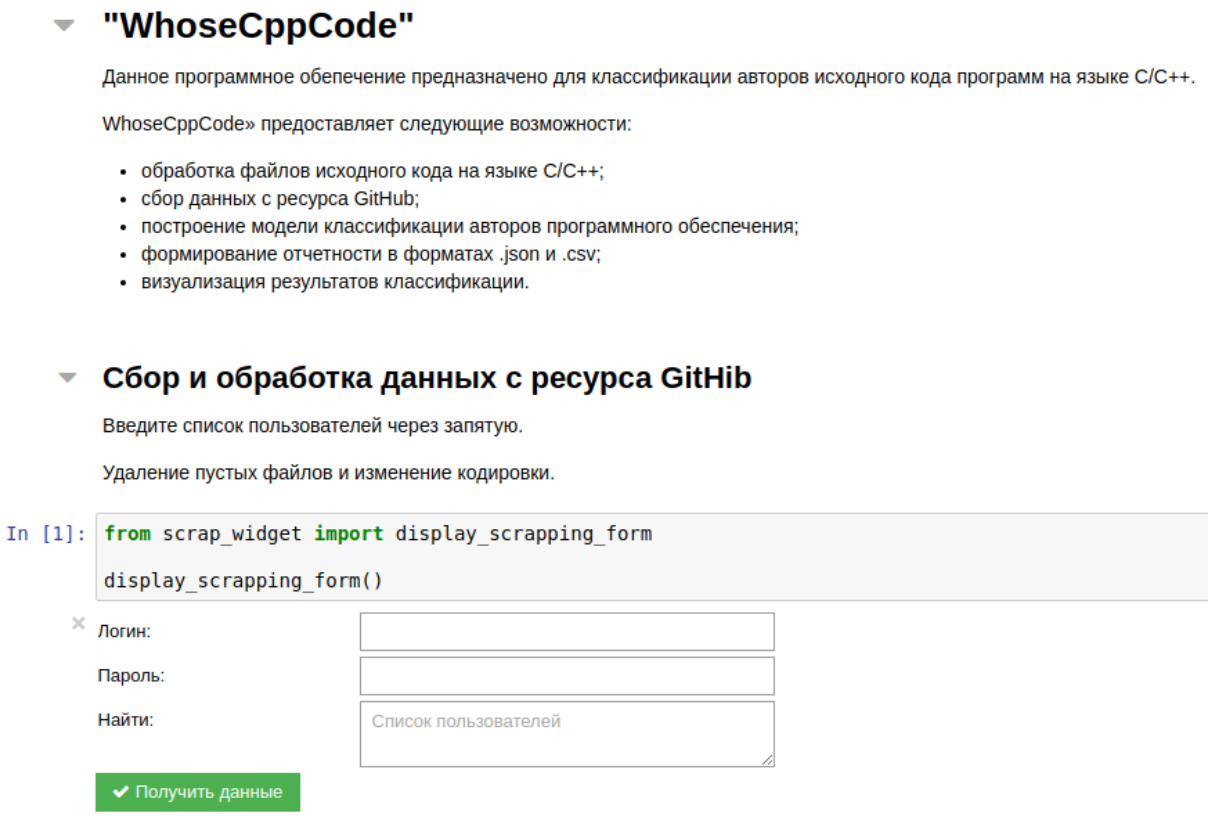

Рисунок 9.2 – Вид программного интерфейса: сбор и обработка данных

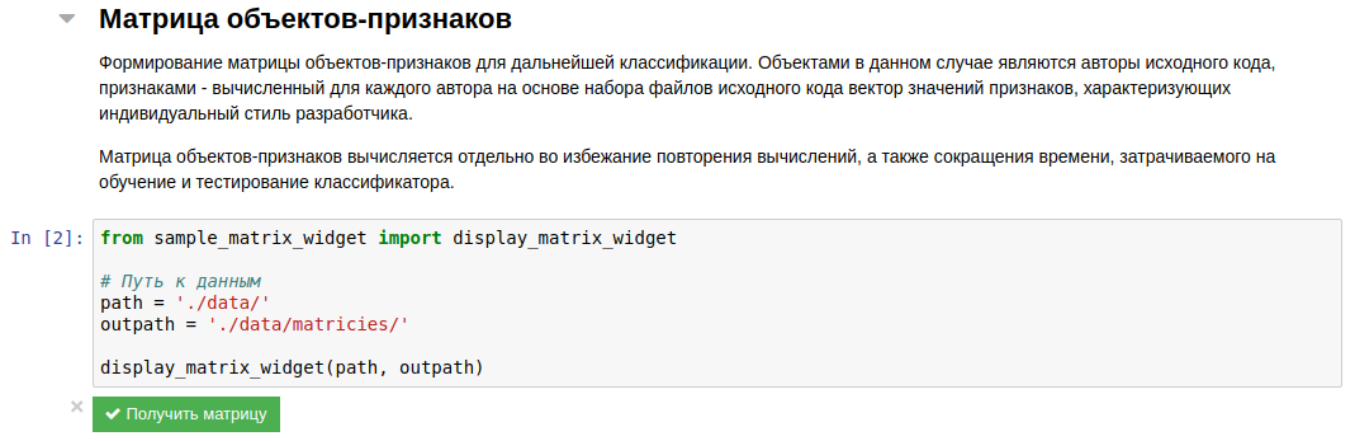

Рисунок 9.3 – Вид программного интерфейса: формирование матрицы объектов-признаков

Классификация

Построение модели классификации, ее обучение на выбранном наборе данных, визуализация результатов.

Отчеты по результатам работы программы в форматах .json и .csv располагаются в директории results в корне проекта.

Ввод:

- **Циклов** - количество итераций эксперимента
- **Данные** - данные для классификации:
  - students - лабораторные работы студентов каф. КИБЭВС по дисциплине "Основы программирования"
  - Google Code Jam 2016 - работы участников ежегодной олимпиады по программированию от компании Google
  - GitHub - данные с веб-хостинга GitHub
- **Алгоритм** - алгоритм классификации

```
In [3]: from main_widget import display_main_form

        display_main_form()
```

Циклов: 1

Данные: students

Алгоритм: RandomForest

✔ Классифицировать

Рисунок 9.4 – Вид программного интерфейса: классификация

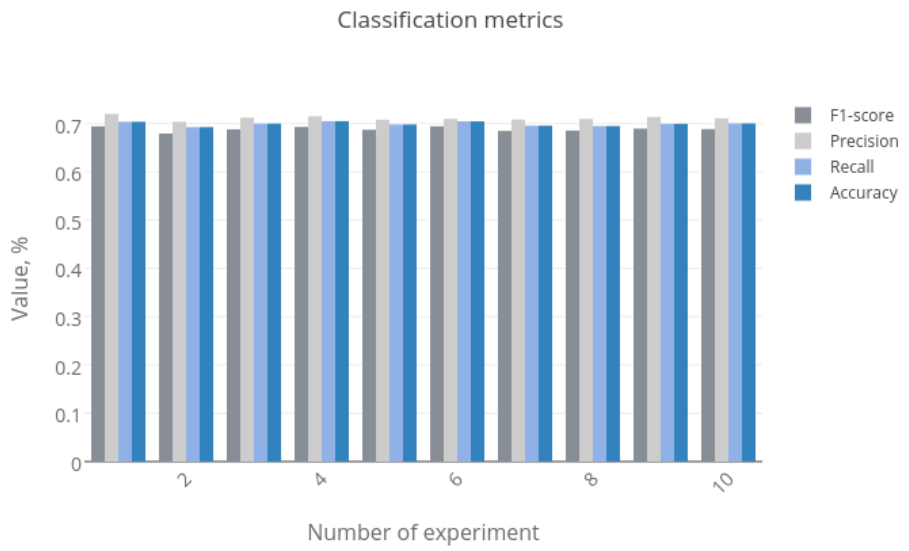

Рисунок 9.5 – Вывод диаграммы результатов классификации

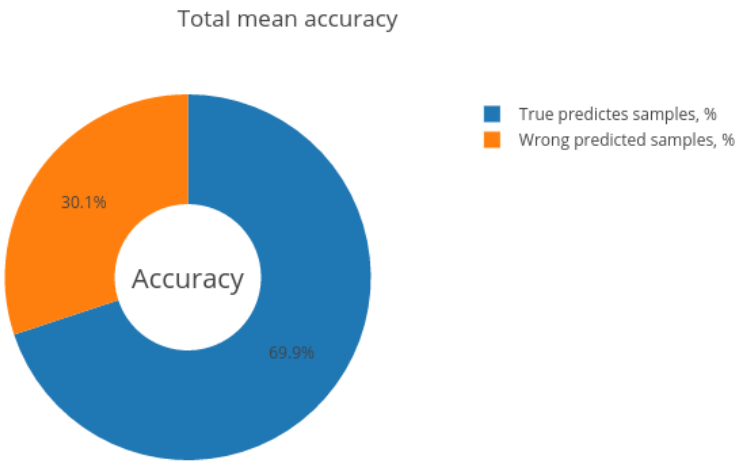

Рисунок 9.6 – Пример вывода диаграммы для средней точности классификации

10 Руководство программиста

10.1 Аннотация

Настоящий раздел содержит назначение, условия применения, характерные особенности и описание возможностей программного обеспечения «WhoseCppCode». Предъявлены требования к квалификации программиста. Указана последовательность действий программиста, обеспечивающих установку, настройку, загрузку, запуск, выполнение и завершение программы, приведено описание входных и выходных данных, а также сообщения, выводимые программой, и соответствующие им действия.

10.2 Назначение программы

Программное обеспечение (ПО) «WhoseCppCode» предназначено для определения авторства программ на языке C/C++ по исходному коду и может быть использовано организациями, занимающимися решением вопросов информационной безопасности, лицензирования ПО, интеллектуальной собственности и расследования инцидентов, связанных с применением вредоносного ПО. Программа состоит из программного модуля, реализующего заявленный функционал, и интерфейса, предназначенного для визуализации ввода и вывода данных, а также удобной работы с возможностями основного модуля. При этом интерфейс не является обязательным, программа может быть использована в качестве модуля при разработке иных автоматизированных систем.

ПО «WhoseCppCode» предоставляет следующие возможности:

- обработка файлов исходного кода на языке C/C++;
- сбор данных с ресурса GitHub;
- построение модели классификации авторов программного обеспечения;
- формирование отчетности в форматах *.json и *.csv;
- визуализация результатов классификации.

Работа с ПО «WhoseCppCode» доступна всем пользователям с доступом к предварительно установленной и настроенной рабочей программной среде, реализованной на ПЭВМ, специально предназначенном сервере или с помощью средств виртуализации.

10.3 Уровень подготовки программиста

Программист, использующий ПО «WhoseCppCode», должен иметь опыт работы с ОС Linux, базовые навыки программирования, а также обладать следующими знаниями:

– знать соответствующую предметную область;

– понимать основы машинного обучения, построения и оценки моделей классификации.

Квалификация программиста должна позволять осуществлять установку и настройку программной среды для работы системы, а также сбор и анализ данных.

10.4 Условия применения программы

Для работы с ПО «WhoseCppCode» необходимо следующее программное обеспечение:

– ОС Linux (тестирование программы производилось на ОС Ubuntu 17.04) с доступом к глобальной сети Интернет;

– программная среда с установленными зависимостями (библиотеками);

– веб-обозреватель.

Инструкция по установке и настройке программной среды приведена в разделе 10.6 настоящего документа.

Требования к характеристикам автоматизированного рабочего места:

– 2,00 Гб ОЗУ или выше;

– CD-ROM;

– 1 Гб на жестком диске.

10.5 Характеристика программы

ПО «WhoseCppCode» состоит из двух основных частей:

– программного модуля, реализующего все необходимые функции для сбора, анализа и обработки данных, а также построения модели классификации авторов исходного кода

– программного интерфеса на основе технологии Jupyter Notebook, предназначенного для визуализации полученных в ходе классификации результатов, сбора необходимых данных с ресурса GitHub, построения матрицы объектов-признаков на основе входных данных, проведения вычислительных экспериментов.

Интерфейс основан на веб-технологиях, может использоваться для демонстрации возможностей программ на языке Python. ПО «WhoseCppCode» использует ряд сторонних библиотек, реализующих алгоритмы классификации, используемых в процессе классификации авторов исходного кода, обеспечивающих работу интерфейса, а также визуализацию полученных результатов. Подробное описание особенностей программы приведено в пояснительной записке к ПО «WhoseCppCode».

Основной модуль программы «WhoseCppCode» может быть использован отдельно от Jupyter Notebook при разработке различного рода программ, систем и интерфейсов лицами, заинтересованными в задаче классификации программистов.

Время выполнения программы зависит от вычислительных мощностей автоматизированного рабочего места, количества данных, подвергаемых обработке, а также количества циклов работы классификатора (число итераций вычислительных экспериментов), задаваемых пользователем.

10.6 Инструкция по установке и настройке программной среды

Ниже приведена последовательность терминальных команд по установке и настройке ПО «WhoseCppCode» в ОС Ubuntu:

1) Перейти в директорию проекта:

$ cd whose_cpp_code

2) Установить Python версии 3.6:

$ sudo apt-get install python3.6

3) Создать виртуальную оболочку, в которую будут установлены необходимые зависимости проекта:

$ virtualenv --python=python3.6

4) Активировать виртуальную оболочку:

$ source whose_cpp_code/bin/activate

5) Установить необходимые зависимости:

$ pip install -r requirements.txt

6) Запустить локальный сервер Jupyter Notebook:

$ jupyter notebook

После запуска локального сервера в веб-обозревателе автоматически открывается страница проекта. Необходимо запустить файл user_notebook.ipynb (рис. 10.1).

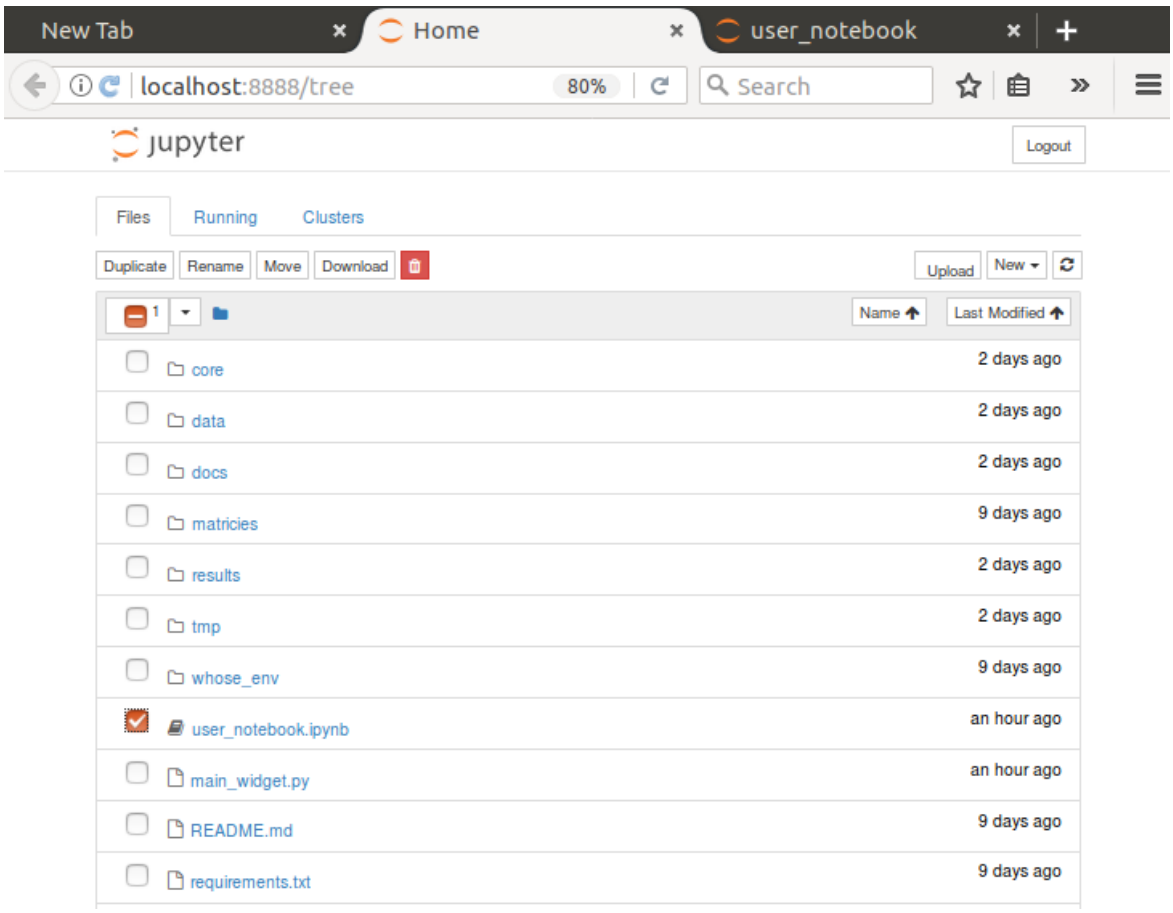

Рисунок 10.1 — Главное окно Jupyter Notebook

10.7 Обращение к программе

Для начала работы с программой необходимо открыть в веб-обозревателе адрес сервера JupyterNotebook, перейти в меню «Cell» и запустить выполнение ячеек командой «Run All», после чего дождаться загрузки всех ячеек (рис. 10.2).

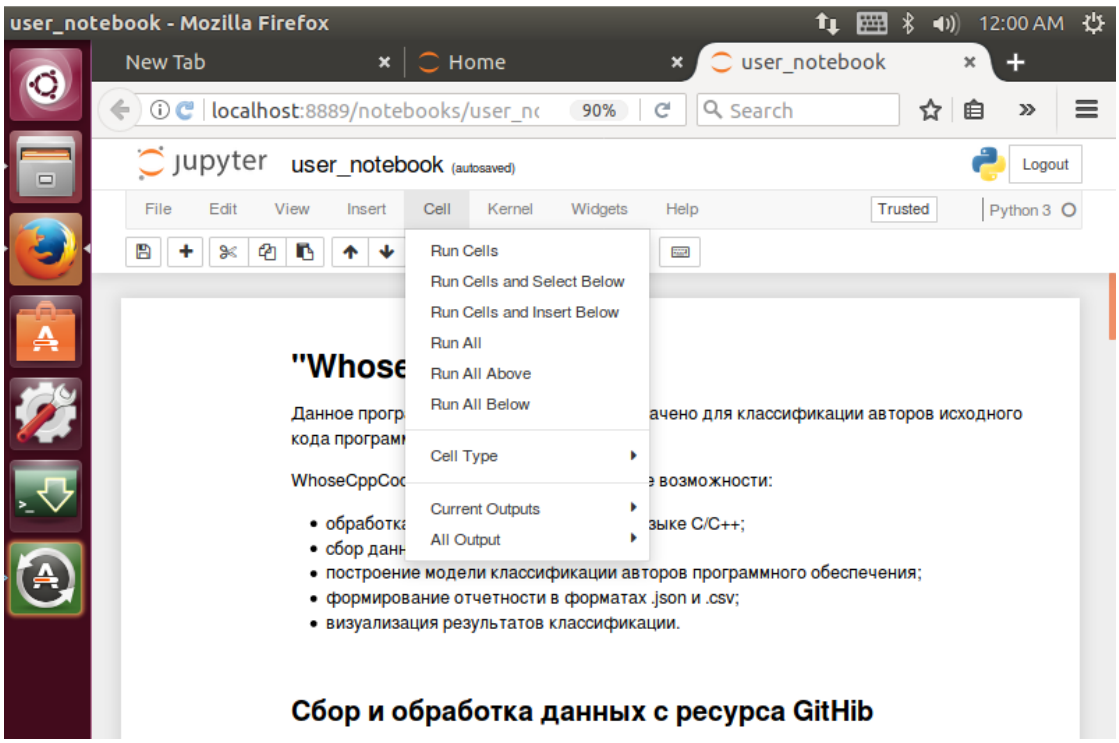

Рисунок 10.2 — Запуск программы

10.7.1 Сбор и обработка данных с ресурса GitHub

Форма, приведенная на рисунке 10.3, предназначена для сбора файлов исходного кода на языке C/C++ с веб-хостинга GitHub. При этом необходимо иметь аутентификационные данные, соответственно, пользователь должен быть зарегистрирован на сайте www.github.com. Помимо логина и пароля в форму через запятую вводится список пользователей, чьи файлы будут загружены и обработаны.

В процессе работы данной формы файлы указанных пользователей загружаются в корень проекта в директорию «data», все файлы конвертируются в кодировку UTF-8, удаляются пустые. Подобная обработка необходима для корректной дальнейшей работы с загруженными файлами.

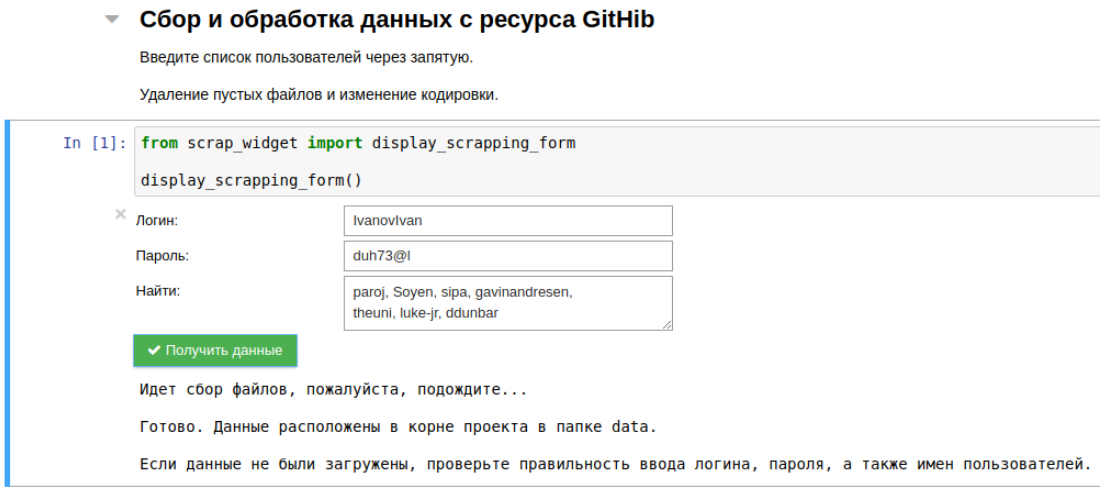

Рисунок 10.3 — Пример работы с формой «Сбор и обработка данных»

10.7.2 Вычисление матрицы объектов-признаков

На рисунке 10.4 приведен пример работы с формой, предназначенной для формирования на основе загруженных данных матрицы-объектов признаков, подаваемой в последствии на вход алгоритму классификации. В данном случае под объектами подразумеваются авторы программ на языке C/C++, каждому из которых соответствует вектор стилистических признаков, вычисленных на основе представленных файлов исходного кода.

Матрица объектов-признаков вычисляется отдельно во избежание повторения вычислений и для экономии времени, затрачиваемого на работу программы, поскольку процесс классификации предполагает несколько циклов вычислительных экспериментов, включающих обучение и тестирование классификатора.

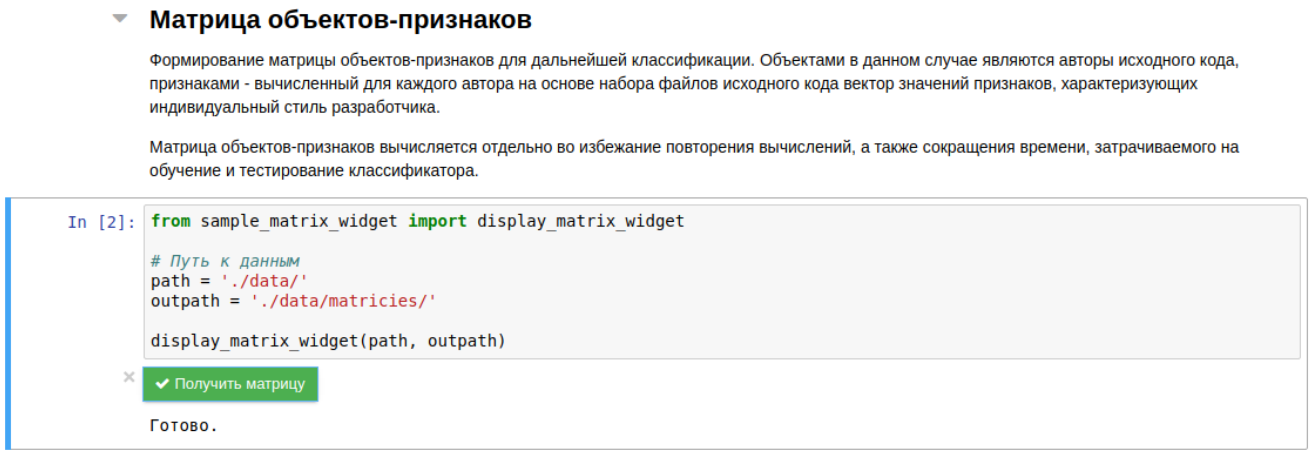

Рисунок 10.4 — Пример работы с формой «Матрица объектов-признаков»

10.7.3 Классификация, входные данные и вывод программы

Основной функцией пользовательского интерфейса ПО «WhoseCppCode» является демонстрация процесса классификации авторов исходного кода программ на языке C/C++ (рис. 6.4) на заранее сформированных наборах данных:

– «students» — лабораторные работы студентов ТУСУР кафедры КИБЭВС 1-го курса обучения по дисциплине «Основы программирования»;

– «Google Code Jam 2016» — работы участников ежегодной олимпиады по программи-рованию от компании Google;

– «GitHub» — данные с веб-хостинга www.github.com.

Существует возможность классификации пользовательского набора данных. Для этого необходимо сформировать и обработать данные при помощи форм, описанных в разделах 10.7.1 и 10.7.2, после чего в выпадающем меню «Данные» выбрать опцию «user_data» (рис. 10.5). При этом на вход классификатору будет подаваться сформированная пользователем матрица объектов-признаков, расположенная в корне проекта в директории data/matrices.

Все наборы данных, а также список GitHub-пользователей, чьи программы подвергались классификации, прилагаются к ПО «WhoseCppCode».

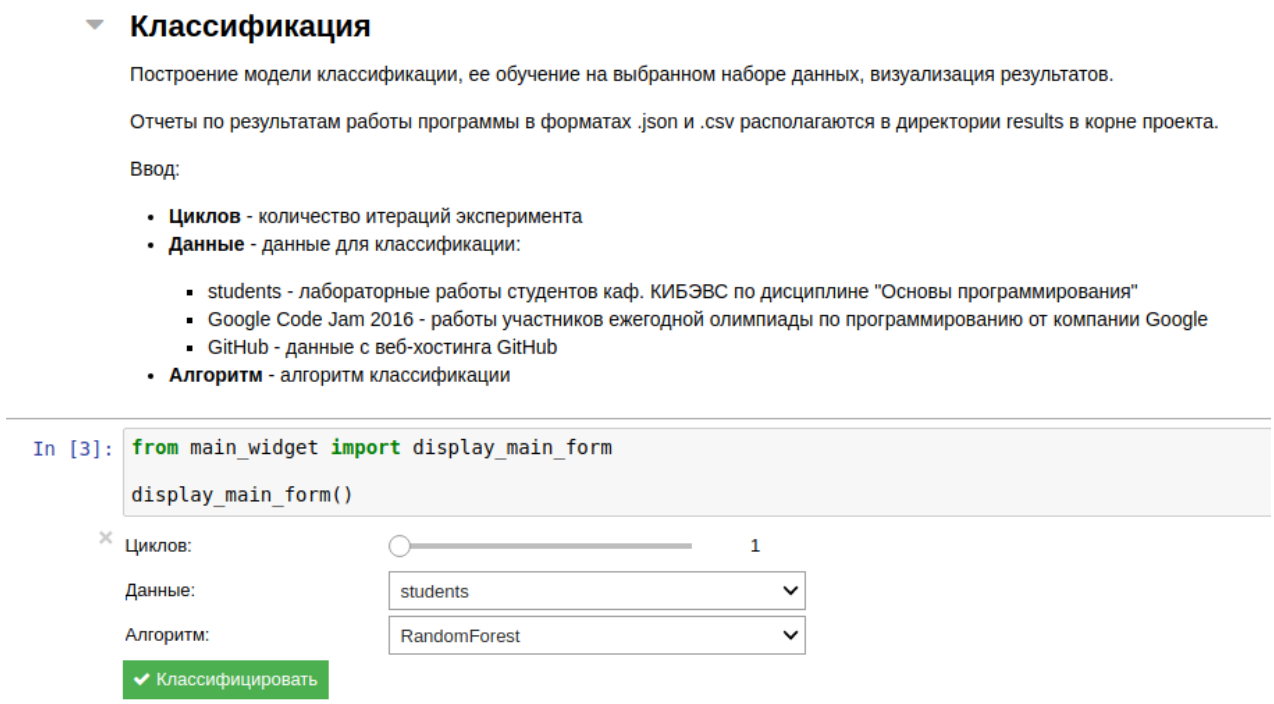

Рисунок 10.5 — Пример работы с формой «Классификация»

В форме «Классификация» существует возможность выбора числа циклов (количество итераций вычислительных экспериментов), набора данных, на которых будет производиться процесс классификации (рис. 10.6), алгоритма классификации (рис. 10.7).

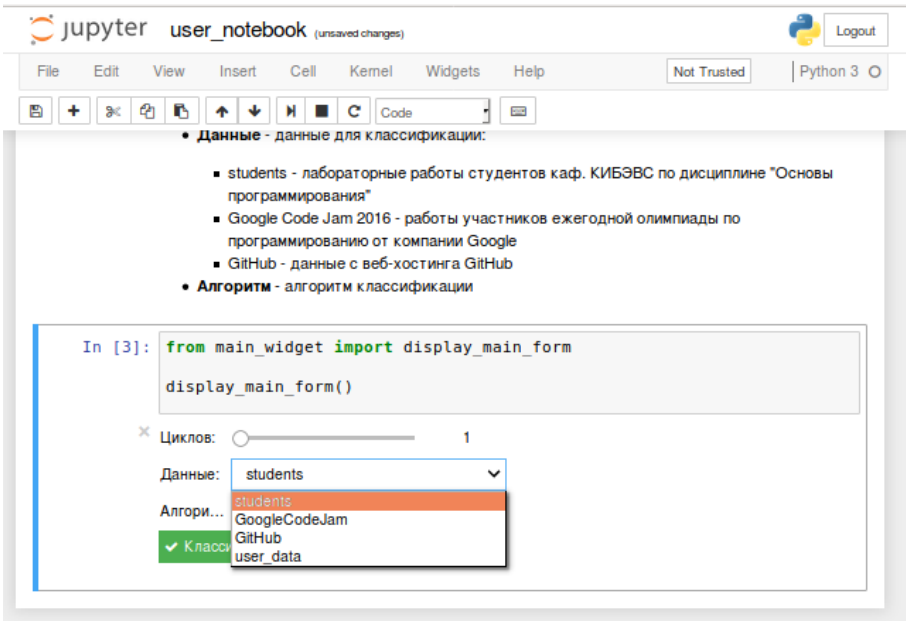

Рисунок 10.6 — Выбор набора данных для дальнейшей классификации

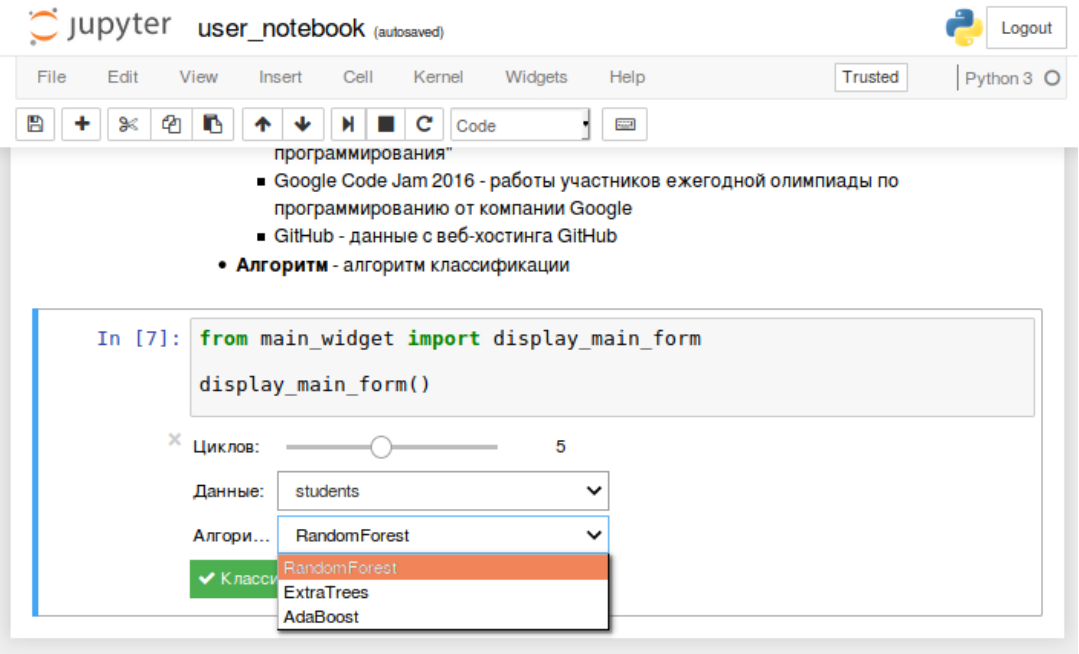

Рисунок 10.7 — Выбор алгоритма классификации

На рисунках 10.8, 10.9 и 10.10 представлен вывод результатов работы программы. Полученные диаграммы можно экспортировать в формате *.png с помощью команды «Download plot as png», как показано на рисунке 10.11.

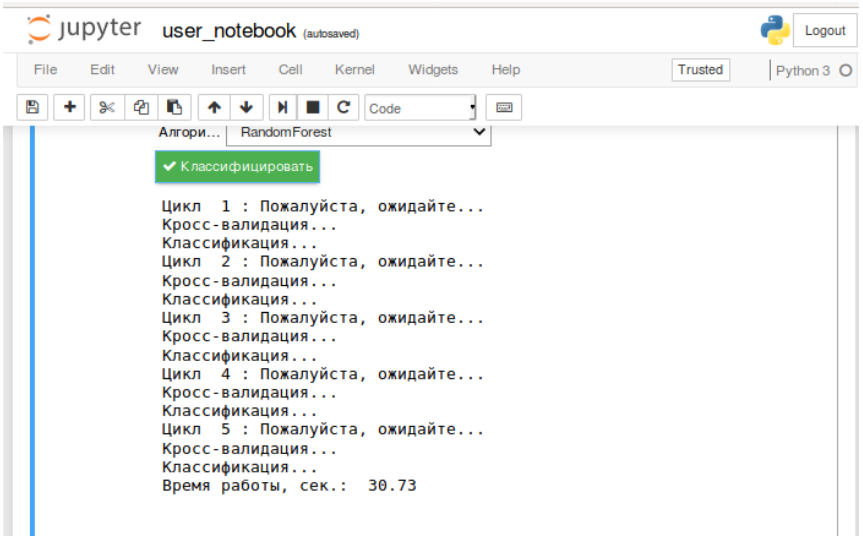

Рисунок 10.8 — Вывод программы в процессе классификации

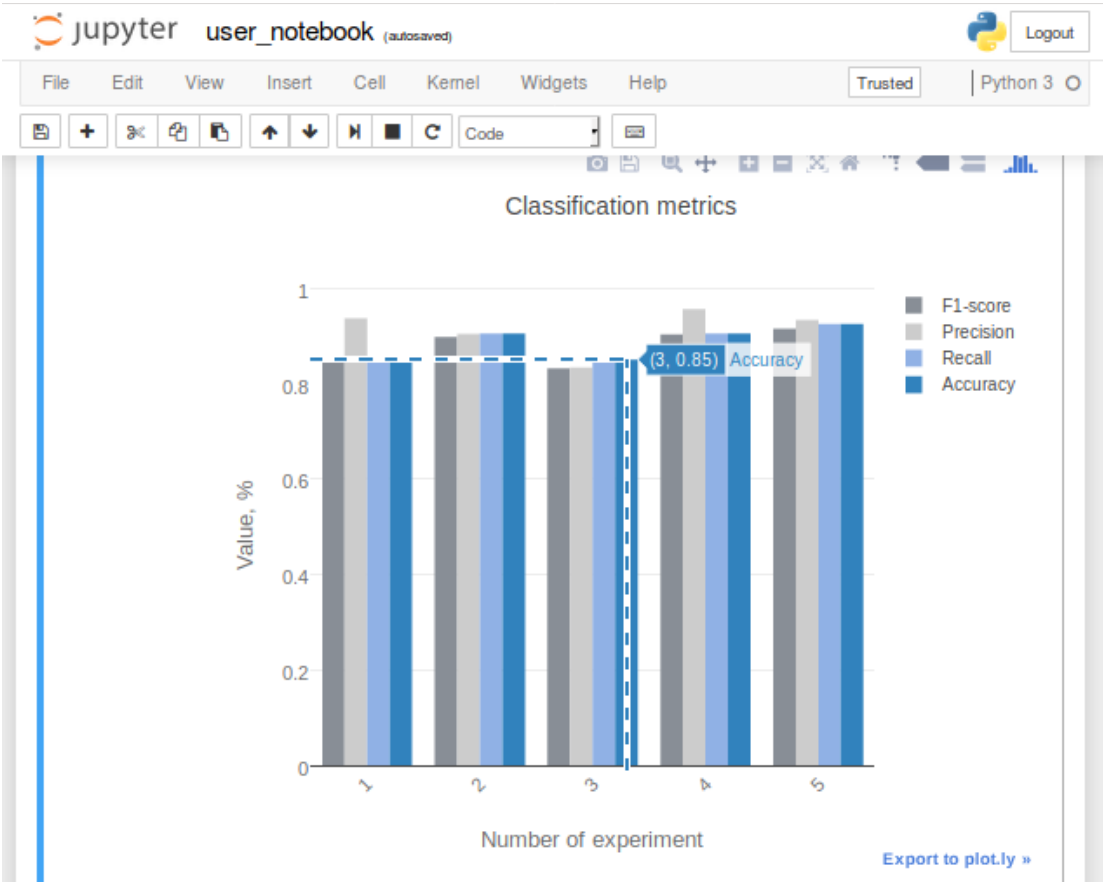

Рисунок 10.9 — Диаграмма значений метрик классификации

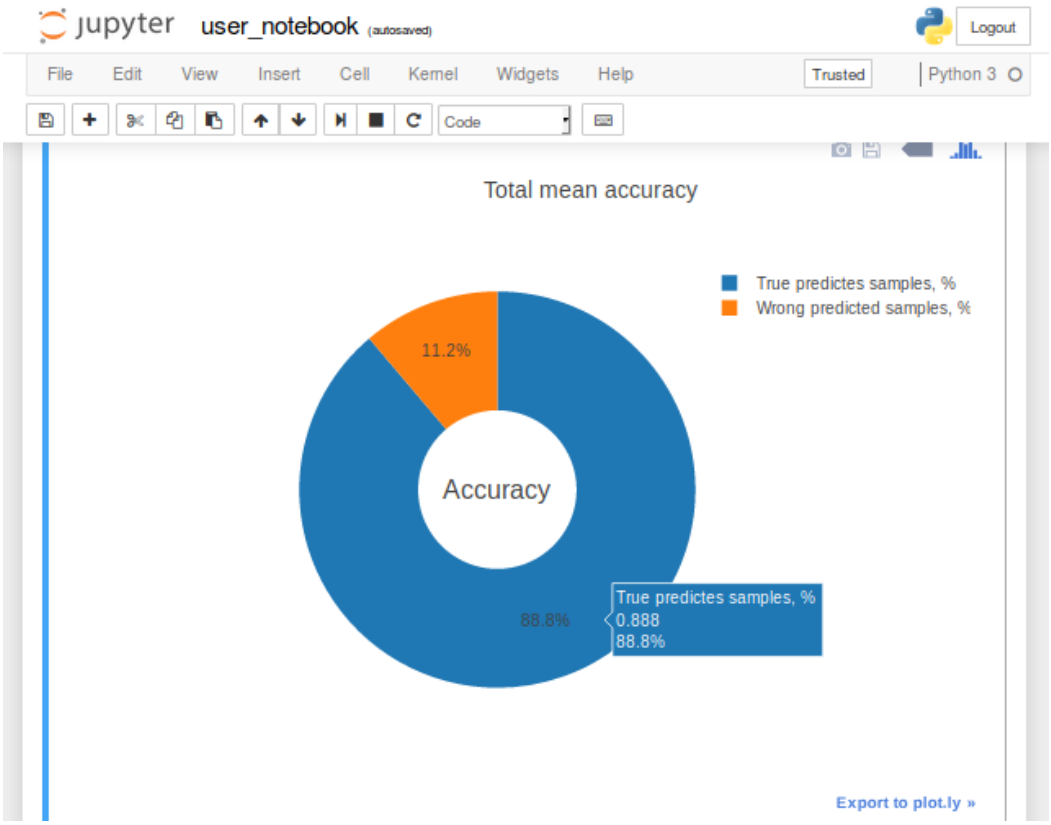

Рисунок 10.10 — Диаграмма значения средней точности классификации

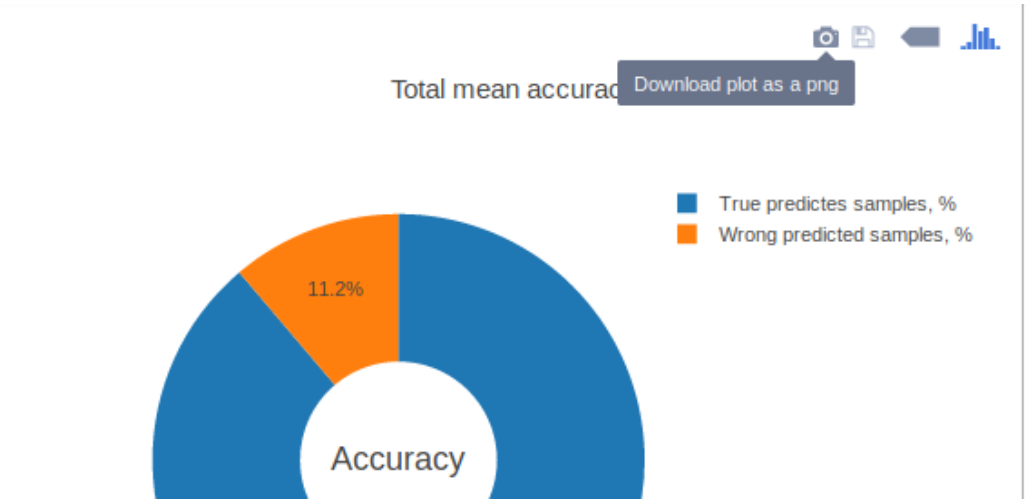

Рисунок 10.11 — Сохранение диаграммы в формате *.png

10.8 Сообщения

Выводимые сообщения и соответствующие им действия:

– «Пожалуйста, введите данные для аутентификации на сайте www.github.com». Заполнены не все поля формы для сбора данных, необходимо ввести логин и пароль GitHub-пользователя.

– «Идет сбор файлов, пожалуйста, подождите…». Сообщение о запуске процесса сбора и обработки данных с веб-хостинга www.github.com.

– «Готово. Данные расположены в корне проекта в папке data. Если данные не были загружены, проверьте правильность ввода логина, пароля, а также имен пользователей.». Завершение сбора и обработки данных с веб-хостинга www.github.com. При отсутствии результатов работы программы в указанных директориях, следует проверить корректность ввода логина и пароля для аутентификации на сайте www.github.com, а также имен GitHub-пользователей, чьи файлы будут подвержены сбору и обработке.

– «Готово». Завершение процесса преобразования данных в матрицу объектов-призна-ков.

– «Цикл 1: Пожалуйста, ожидайте… Кросс-валидация…

Классификация...». Вывод во время процесса классификации для контроля за выполнением программы.

– «Время работы, сек.: 9.42». Классификация успешно завершена, вывод суммарного времени работы в секундах.

Для повторного запуска выполнения любой из форм программ, необходимо нажать на нужную ячейку щелчком мыши, после чего — комбинацию клавиш «Ctrl + Enter» и продолжить работу с формой. При сбое в работе программы необходимо перезапустить все ячейки, перейдя в меню «Cell Run — All», дождаться загрузки ячеек.

11 Руководство пользователя

11.1 Аннотация

Настоящий раздел содержит назначение, условия применения и описание возможностей программного обеспечения «WhoseCppCode». Предъявлены требования к квалификации пользователя. Указана последовательность действий пользователя, обеспечивающих загрузку, запуск, выполнение и завершение программы, приведены сообщения, выводимые программой.

11.2 Назначение программы

Программное обеспечение (ПО) «WhoseCppCode» предназначено для определения авторства программ на языке C/C++ по исходному коду и может быть использовано организациями, занимающимися решением вопросов информационной безопасности, лицензирования ПО, интеллектуальной собственности и расследования инцидентов, связанных с применением вредоносного ПО. Программа состоит из программного модуля, реализующего заявленный функционал, и интерфейса, предназначенного для визуализации ввода и вывода данных, а также удобной работы с возможностями основного модуля. При этом интерфейс не является обязательным, программа может быть использована в качестве модуля при разработке иных автоматизированных систем.

ПО «WhoseCppCode» предоставляет следующие возможности:

- обработка файлов исходного кода на языке C/C++;
- сбор данных с ресурса GitHub;
- построение модели классификации авторов программного обеспечения;
- формирование отчетности в форматах *.json и *.csv;
- визуализация результатов классификации.

Работа с ПО «WhoseCppCode» доступна всем пользователям с доступом к предварительно установленной и настроенной рабочей программной среде,

реализованной на ПЭВМ, специально предназначенном сервере или с помощью средств виртуализации.

11.3 Уровень подготовки пользователя

Пользователь ПО «WhoseCppCode» должен иметь пользовательский опыт работы с ОС Linux, а также обладать следующими знаниями:

– знать соответствующую предметную область;

– понимать основы машинного обучения, построения и оценки моделей классификации.

Квалификация пользователя должна позволять осуществлять сбор и анализ данных.

11.4 Условия выполнения программы

Для работы с ПО «WhoseCppCode» необходимо следующее программное обеспечение:

– 64-разрядная ОС Linux (тестирование программы производилось на ОС Ubuntu 17.04) с доступом к глобальной сети Интернет;

– программная среда с установленными зависимостями (библиотеками);

– веб-обозреватель.

Требования к характеристикам автоматизированного рабочего места:

– 2,00 Гб ОЗУ или выше;

– CD-ROM;

– 1 Гб на жестком диске.

11.5 Выполнение программы

Для начала работы с программой необходимо открыть в веб-обозревателе адрес сервера JupyterNotebook, перейти в меню «Cell» и запустить выполнение ячеек командой «Run All», после чего дождаться загрузки всех ячеек (рис. 11.1).

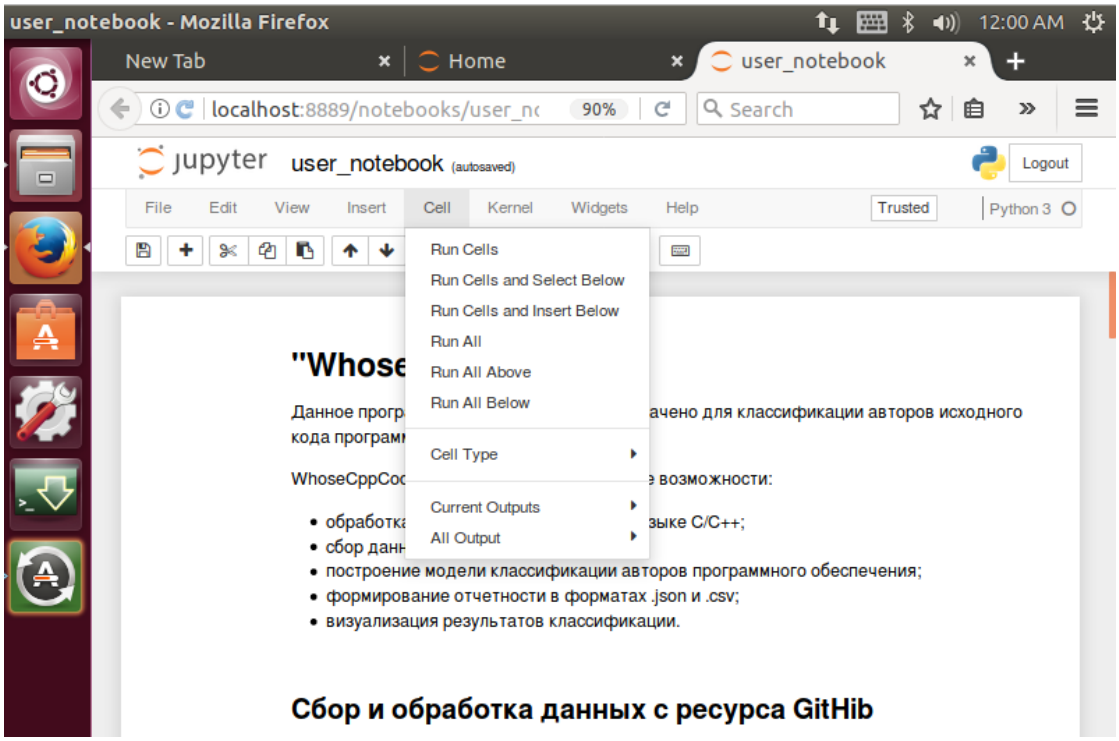

Рисунок 11.1 — Запуск программы

11.5.1 Сбор и обработка данных с ресурса GitHub

Форма, приведенная на рисунке 11.2, предназначена для сбора файлов исходного кода на языке C/C++ с веб-хостинга GitHub. При этом необходимо иметь аутентификационные данные, соответственно, пользователь должен быть зарегистрирован на сайте www.github.com. Помимо логина и пароля в форму через запятую вводится список пользователей, чьи файлы будут загружены и обработаны.

В процессе работы данной формы файлы указанных пользователей загружаются в корень проекта в директорию «data», все файлы конвертируются в кодировку UTF-8, удаляются пустые. Подобная обработка необходима для корректной дальнейшей работы с загруженными файлами.

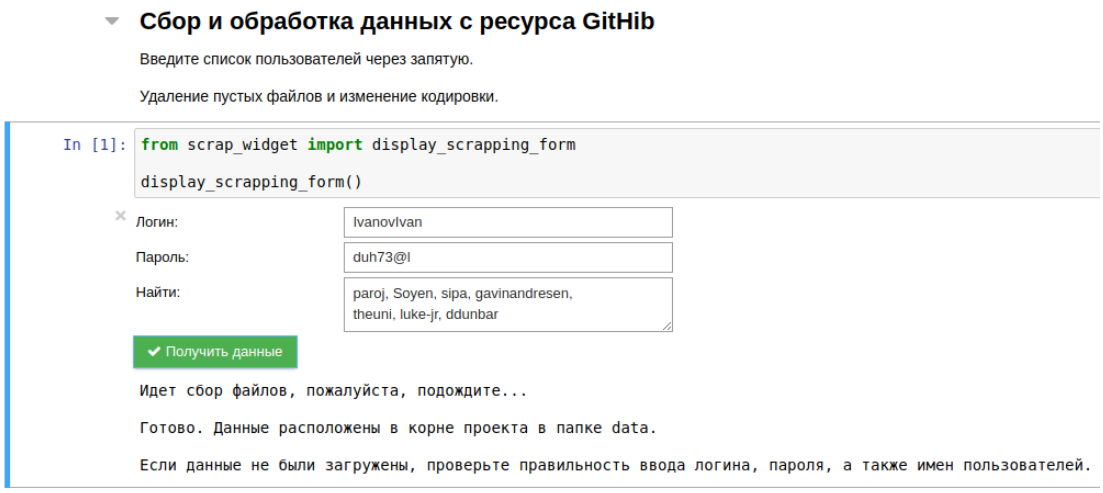

Рисунок 11.2 — Пример работы с формой «Сбор и обработка данных»

11.5..2 Вычисление матрицы объектов-признаков

На рисунке 11.3 приведен пример работы с формой, предназначенной для формирования на основе загруженных данных матрицы-объектов признаков, подаваемой в последствии на вход алгоритму классификации. В данном случае под объектами подразумеваются авторы программ на языке C/C++, каждому из которых соответствует вектор стилистических признаков, вычисленных на основе представленных файлов исходного кода.

Матрица объектов-признаков вычисляется отдельно во избежание повторения вычислений и для экономии времени, затрачиваемого на работу программы, поскольку процесс классификации предполагает несколько циклов вычислительных экспериментов, включающих обучение и тестирование классификатора.

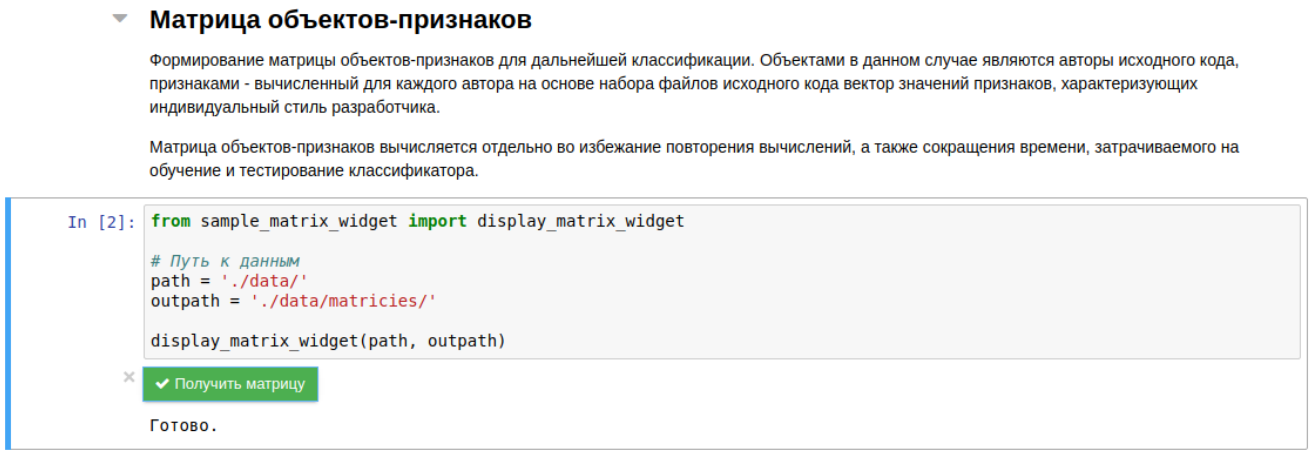

Рисунок 11.3 — Пример работы с формой «Матрица объектов-признаков»

11.5.3 Классификация

Основной функцией пользовательского интерфейса ПО «WhoseCppCode» является демонстрация процесса классификации авторов исходного кода программ на языке C/C++ (рис. 11.4) на заранее сформированных наборах данных:

– «students» — лабораторные работы студентов ТУСУР кафедры КИБЭВС 1-го курса обучения по дисциплине «Основы программирования»;

– «Google Code Jam 2016» — работы участников ежегодной олимпиады по программи-рованию от компании Google;

– «GitHub» — данные с веб-хостинга www.github.com.

Существует возможность классификации пользовательского набора данных. Для этого необходимо сформировать и обработать данные при помощи форм, описанных в разделах 11.5.1 и 11.5.2, после чего в выпадающем меню «Данные» выбрать опцию «user_data» (рис. 11.5). При этом на вход классификатору будет подаваться сформированная пользователем матрица объектов-признаков, расположенная в корне проекта в директории data/matrices.

Все наборы данных, а также список GitHub-пользователей, чьи программы подвергались классификации, прилагаются к ПО «WhoseCppCode».

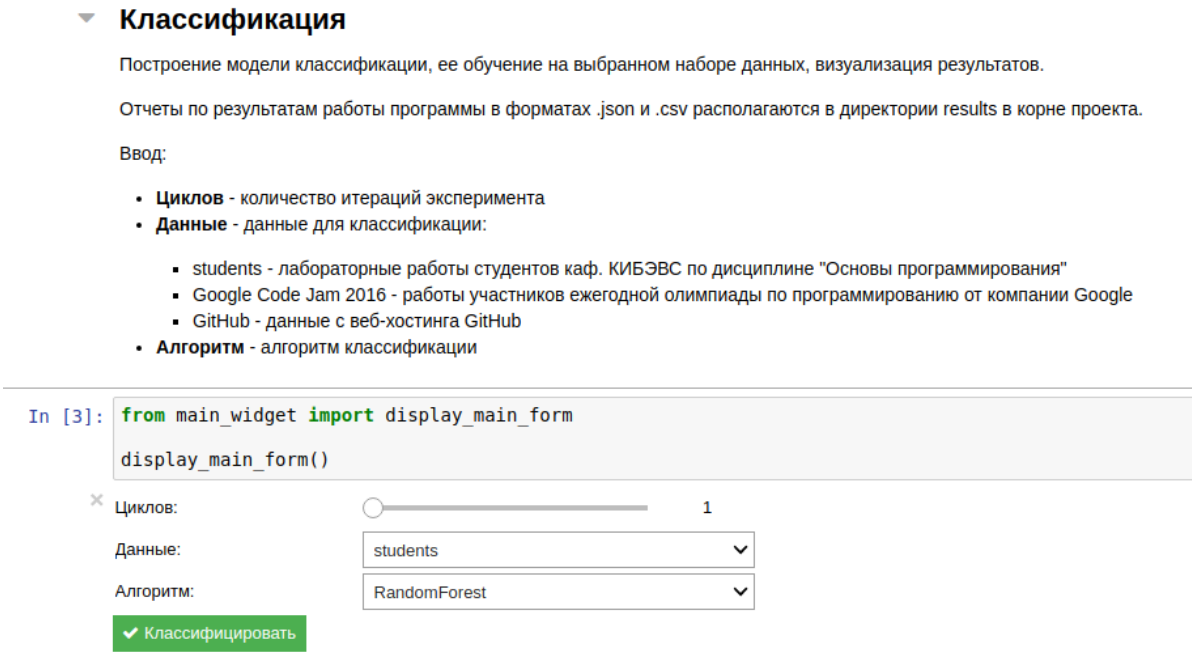

Рисунок 11.4 — Пример работы с формой «Классификация»

В форме «Классификация» существует возможность выбора числа циклов (количество итераций вычислительных экспериментов), набора данных, на которых будет производиться процесс классификации (рис. 11.5), алгоритма классификации (рис. 11.6).

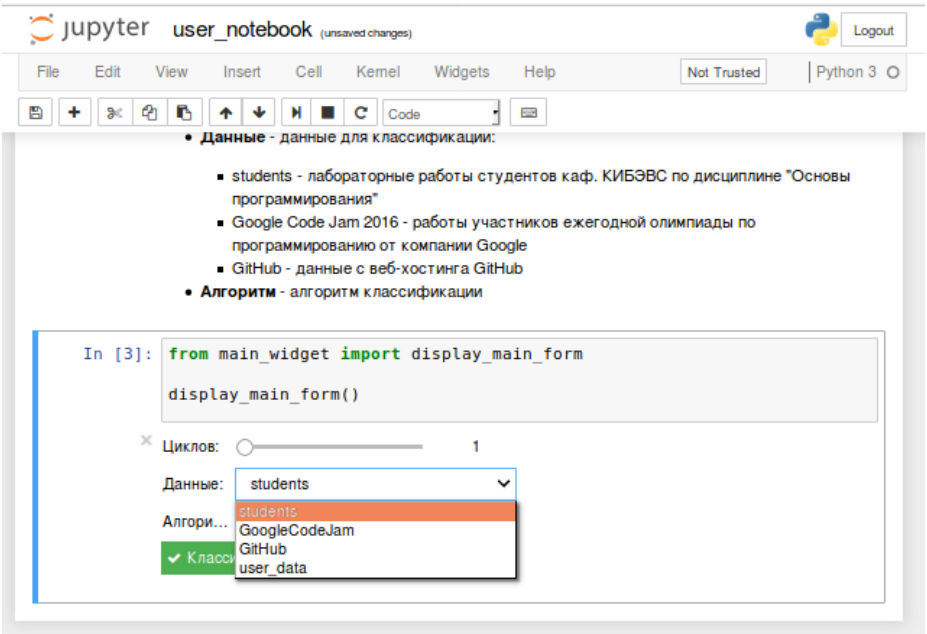

Рисунок 11.5 — Выбор набора данных для дальнейшей классификации

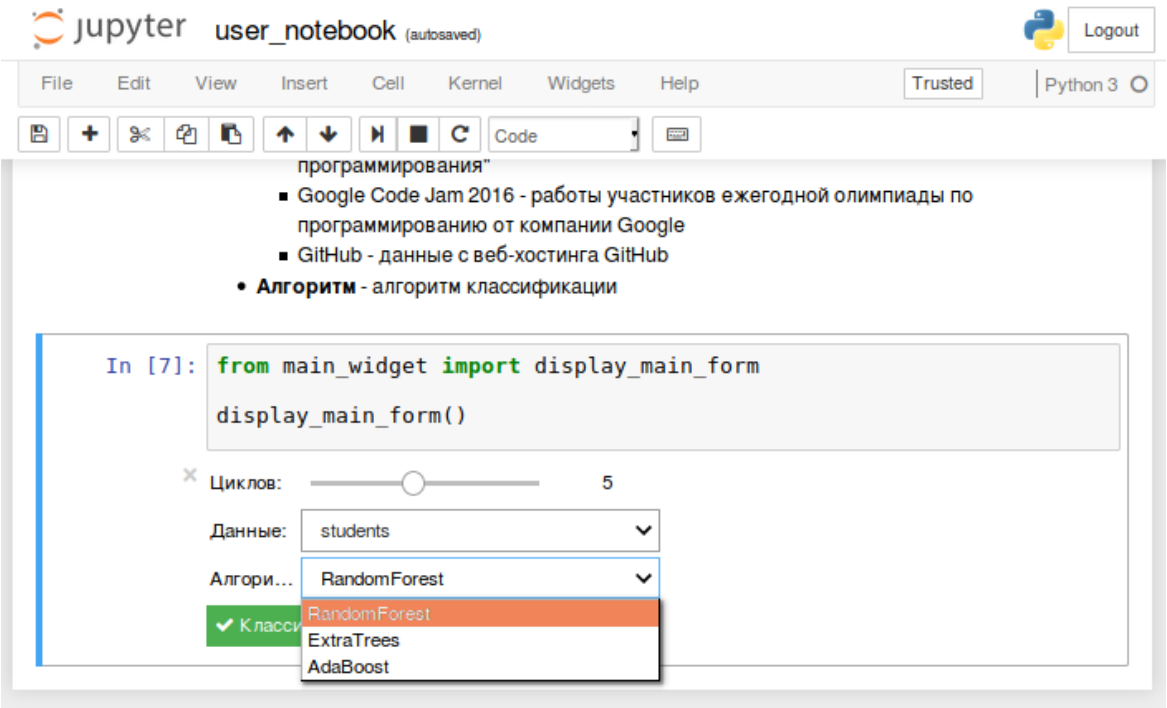

Рисунок 11.6 — Выбор алгоритма классификации

На рисунках 11.7, 11.8 и 11.9 представлен вывод результатов работы программы. Полученные диаграммы можно экспортировать в формате *.png с помощью команды «Download plot as png», как показано на рисунке 4.10.

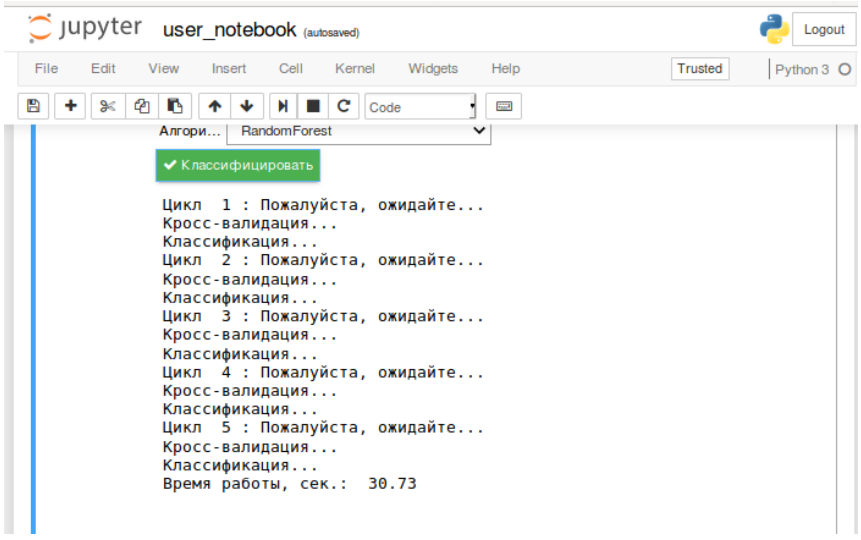

Рисунок 11.7 — Вывод программы в процессе классификации

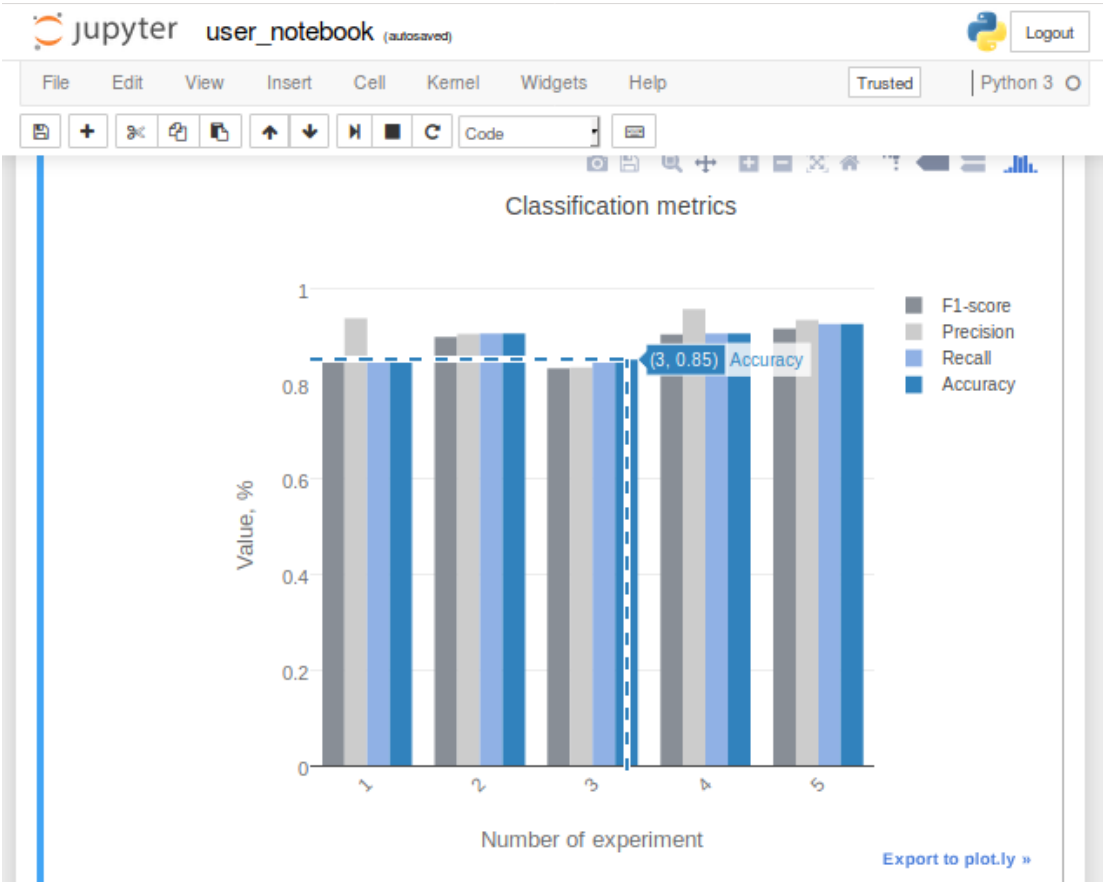

Рисунок 11.8 — Диаграмма значений метрик классификации

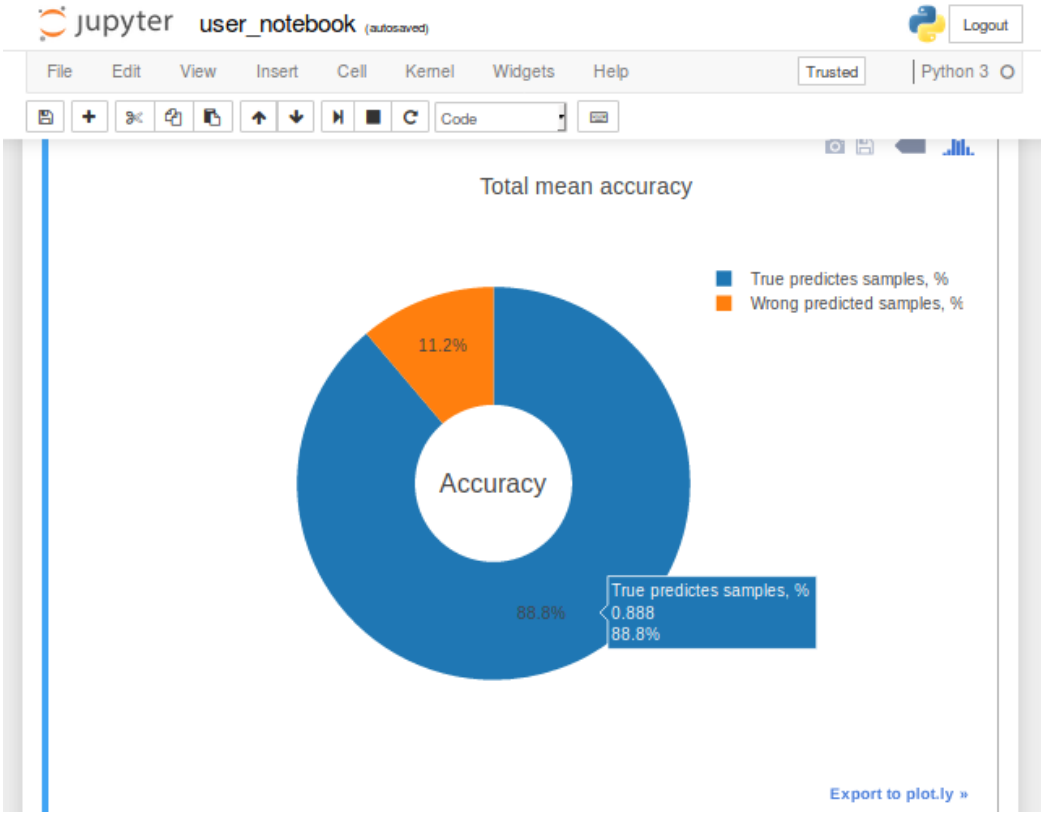

Рисунок 11.9 — Диаграмма значения средней точности классификации

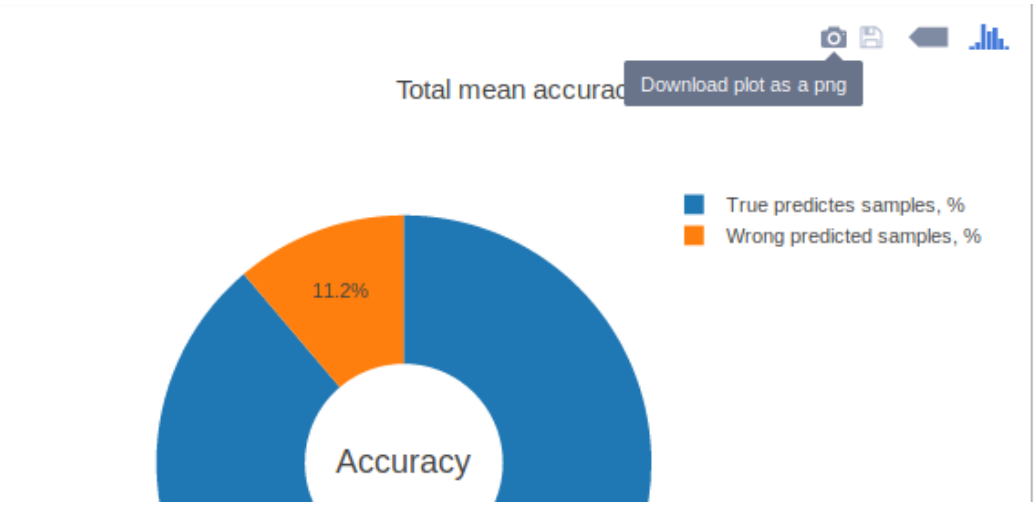

Рисунок 11.10 — Сохранение диаграммы в формате *.png

11.6 Сообщения пользователю

Выводимые сообщения и соответствующие действия пользователя:

– «Пожалуйста, введите данные для аутентификации на сайте www.github.com». Заполнены не все поля формы для сбора данных, необходимо ввести логин и пароль GitHub-пользователя.

– «Идет сбор файлов, пожалуйста, подождите…». Сообщение о запуске процесса сбора и обработки данных с веб-хостинга www.github.com.

– «Готово. Данные расположены в корне проекта в папке data. Если данные не были загружены, проверьте правильность ввода логина, пароля, а также имен пользователей.». Завершение сбора и обработки данных с веб-хостинга www.github.com. При отсутствии результатов работы программы в указанных директориях, следует проверить корректность ввода логина и пароля для аутентификации на сайте www.github.com, а также имен GitHub-пользователей, чьи файлы будут подвержены сбору и обработке.

– «Готово». Завершение процесса преобразования данных в матрицу объектов-призна-ков.

– «Цикл 1: Пожалуйста, ожидайте… Кросс-валидация…

Классификация...». Вывод во время процесса классификации для контроля за выполнением программы.

– «Время работы, сек.: 9.42». Классификация успешно завершена, вывод суммарного времени работы в секундах.

Для повторного запуска выполнения любой из форм программ, необходимо нажать на нужную ячейку щелчком мыши, после чего — комбинацию клавиш «Ctrl + Enter» и продолжить работу с формой. При сбое в работе программы необходимо перезапустить все ячейки, перейдя в меню «Cell Run — All», дождаться загрузки ячеек. Если программа по-прежнему работает некорректно, необходимо обратиться к программисту, ответственному за установку и настройку программы.

12 Технико-экономическое обоснование дипломной работы

12.1 Обоснование актуальности дипломной работы

Вследствие активного развития информационных технологий возрастает и количество преступлений в информационной сфере. Зачастую злоумышленники, как внешние, так и внутренние, используют самостоятельно разработанные программы для осуществления атак на информационные ресурсы. Системы, способные идентифицировать разработчиков вредоносного ПО, могут внести существенный вклад в развитие компьютерной криминалистики, а также оказывать помощь в исследовании вопросов интеллектуальной собственности среди разработчиков программного обеспечения.

Цель настроящей дипломной работы — разработать ПО, способное идентифицировать автора программы по исходному коду, с перспективой его дальнейшего применения в борьбе с киберпреступностью, в области лицензионных, патентных, и иных судебных разбирательств.

12.2 Организация и планирование работы

Судент может быть принят на работу техником. Должностной оклад техника составляет 4400 рублей в месяц. Стоимость одного часа работ руководителя с ученой степенью кандидата наук и должностью доцента согласно коллективному договору ТУСУР на 2016-2019 годы равна 230 рублей.

Согласно выделенным этапам работы расчитывется трудоемкость выполненных работ (табл. 12.1), а также строится диаграмма Ганта (рис. 12.1), отображающего затраты времени на выполнение дипломной работы всеми участниками, представленная в приложении Ж.

Таблица 12.1 – Трудоемкость выполненных работ

Наименование этапов и содержание работ	Исполнитель (должность)	Трудоемкость		Количество исполнителей, чел.
		Нормо-часы, н-ч	Процент от общей трудоемкости, %	
1 Обзор информационных источников	Исп.	6	3	1
2 Моделирование	Исп.	12	7	1
3 Программная реализация разработанной модели	Исп.	60	36	1
4 Разработка программного интерфейса	Исп.	15	9	1
5 Сбор и обработка тестовых данных	Исп.	11	6	1
6 Тестирование и отладка полученной модели	Исп.	6	3	1
7 Анализ результатов	Рук.	2	10	1
	Исп.	12	6	1
8 Проведение расчетов по технико-экономическому обоснованию	Исп.	6	3	1
9 Проведение расчетов по безопасности жизнедеятельности	Исп.	6	3	1
10 Оформление, проверка и исправление пояснительной записки	Рук.	8	40	1
	Исп.	24	14	1
11 Подготовка необходимой документации	Рук.	10	50	1
	Исп.	12	7	1
12 Защита ВКР	Исп.	6	3	1
Всего: 12	Рук.	20	100	1
	Исп.	176	100	1

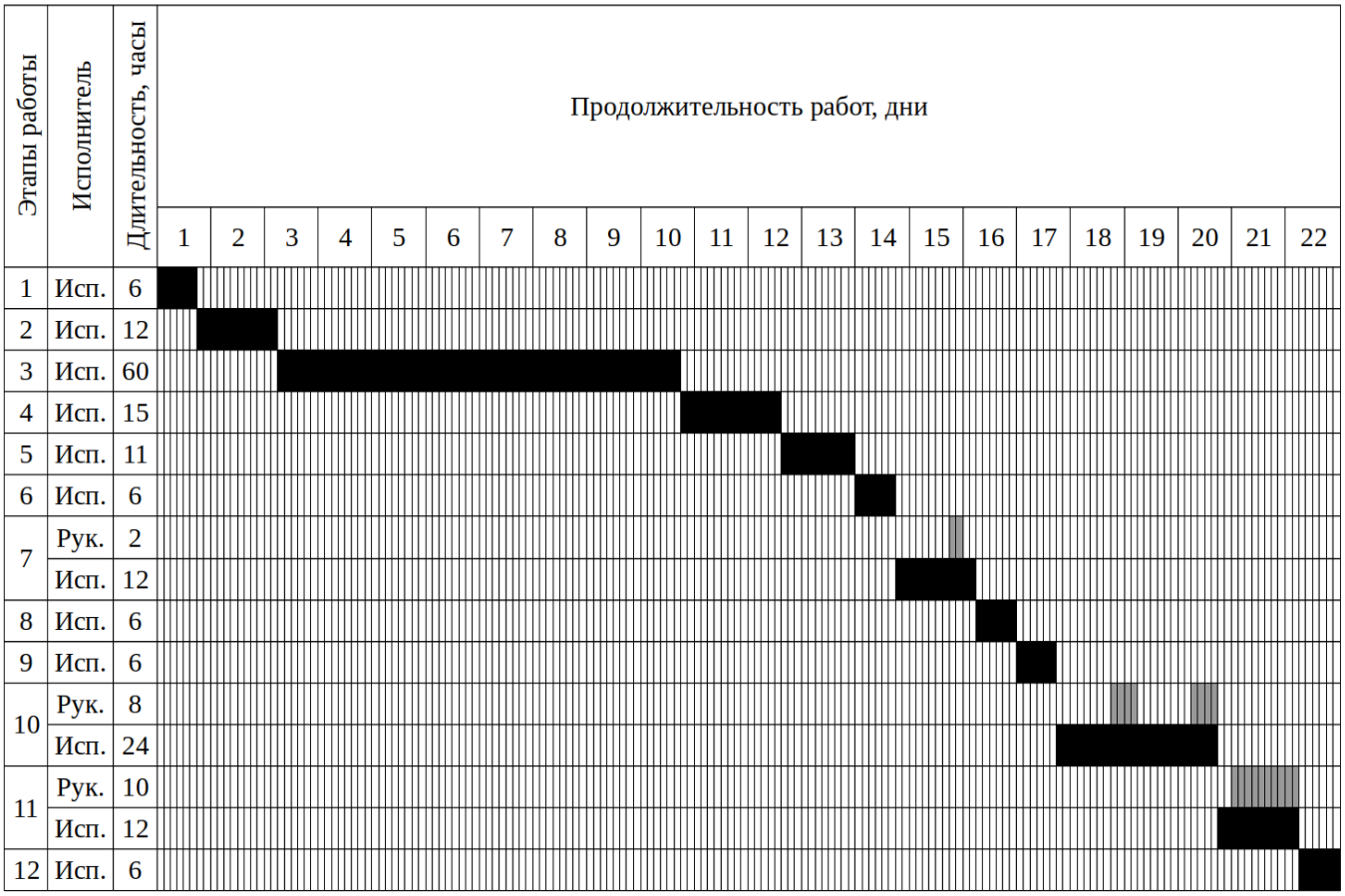

Рисунок 12.1 – Диаграмма Ганта

На работу руководителя отводится 20 часов, исполнителя — 22 рабочих дня по 8 часов.

12.3 Определение сметной стоимости работы

12.3.1 Затраты на оборудование и амортизацию

Основным оборудованием при проведении работы являются компьютер и принтер, которые постановлением Правительства Российской Федерации от 1.01.02 г. № 1 отнесены ко второй амортизационной группе – «имущество со сроком полезного использования свыше 2 лет до 3 лет включительно» [30].

Определим норму амортизации линейным методом [31]. Расчет производится по формулам:

$$N_a = \frac{1}{T_n} \times 100\%,$$

$$A = F_n \times N_a,$$

где N_a — годовая норма амортизации;

T_n — срок полезного использования объекта основных средств;

F_n — первоначальная стоимость объекта основных средств;

A — сумма амортизации за год.

Годовая норма амортизации и для ноутбука, и для принтера с учетом трехлетнего срока использования объектов основных средств составит:

$$N_a = \frac{1}{3} \times 100\% \approx 33,33\%.$$

Результаты расчётов амортизационных отчислений приведены в таблице 12.2.

Таблица 12.2 – Смета затрат на амортизацию

Наименование прибора, оборудования	Потребленное количество, шт.	Цена, руб.		Время использования по теме	Норма амортизации, %	Сумма амортизации, руб.
		Единицы	Всего			
Ноутбук Samsung NP530U3B	1	25000	25000	1 мес.	33,33	700
Итого: 700 руб.						

Затраты на амортизацию составили 700 рублей.

12.3.2 Расходы на оплату труда и социальные взносы

Статья затрат учитывает выплаты по заработной плате работников, вычисленные на основании тарифных ставок и должностных окладов в соответствии с приня-

той в организации-разработчике системой оплаты труда. В этой статье также отражаются расчеты премий, дополнительной заработной платы, страховых взносов. В соответствии со статьей 426 Налогового Кодекса РФ страховые взносы составляют 30,2% [32] от фактической оплаты труда.

Согласно статье 151 Трудового Кодекса РФ «Оплата труда при совмещении профессий (должностей), расширении зон обслуживания, увеличении объема работы или исполнении обязанностей временно отсутствующего работника без освобождения от работы, определенной трудовым договором» [33] доплата работнику производится в случае совмещения профессий (должностей), расширения зон обслуживания, увеличения объема работы или исполнения обязанностей временно отсутствующего работника без освобождения от работы, определенной трудовым договором.

Поскольку исполнителем осуществлялась исключительно запланированная работа, дополнительная заработная плата и премия не учитываются.

Результаты расчёта расходов на оплату труда участников работы представлены в таблице 12.3.

Таблица 12.3 – Расчет расходов на оплату труда участников работы

Участники работы	$ЗП_{пр}$	РК, руб. 30%	$О_{зп}$	ФОТ	Страховые взносы, руб. 30,2%	Всего
Рук.	4600	1380	5980	5980	1085,96	7785,96
Исп.	4840	1452	6292	6292	1900,184	8192,184
Итого: 15978,14 руб.						

Общие затраты на оплату труда участников работы составили 15978 рублей

14 копеек.

12.3.3 Затраты на основные и вспомогательные материалы

Статья включает расходы по приобретению и доставке основных и вспомогательных материалов, необходимых для опытно-экспериментальной проработки решения, для изготовления макета или опытного оборудования. Сюда включаются и стоимость необходимых материалов для изготовления образцов и макетов, и материалов необходимых для оформления требуемой документации.

Результаты расчёта стоимости материалов представлены в 12.4.

Таблица 12.4 – Расчёт затрат на основные и вспомогательные материалы

Наименование материала	Единицы измерения	Потребленное количество	Цена за единицу, руб.	Сумма, руб.
Пачка бумаги	Шт.	1	254	254
DVD-R диск TDK	Шт.	1	40	40
Конверт для CD-диска	Шт.	1	6	6
Итого: 300 руб.				

Затраты на основные и вспомогательные нужды составили 300 рублей.

12.3.4 Расходы на электроэнергию

Статья включает затраты по электроэнергии на технологические нужды. В настоящее время одноставочный тариф на электроэнергию для населения, проживающего в городских населенных пунктах Томской области в домах, оборудованных в установленном порядке электрическими плитами и (или) электроотопительными приборами, составляет 2,17 руб./ кВт ч. Тариф введен Приказом от 23.12.2016 г. № 6-840 «О тарифах на электрическую энергию для населения и потребителей, приравненных к категории население по Томской области на 2017 год» [34], принятый департаментом тарифного регулирования Томской области.

Результаты расчётов приведены в 12.5.

Таблица 12.5 – Затраты на электроэнергию

Наименование прибора или оборудования	Количество, шт.	Потребляемая мощность, кВт.	Часы работы	Тариф за 1 кВт- час, руб.	Стоимость электроэнергии, руб.
Ноутбук Samsung NP530U3B	1	0,05	176	2,17	19,1
Освещение	1	0,6	176	2,17	229,152
Итого: 248,25 руб.					

Затраты на электроэнергию составили 248 рублей 25 копеек.

12.3.5 Накладные расходы

Результаты расчёта накладных расходов приведены в таблице 12.6.

Таблица 12.6 – Накладные расходы

Услуга	Количество	Стоимость одной единицы, руб.	Сумма затрат, руб.
Брошюрирование	1 шт.	50	50
Интернет	1 мес. использования	500	500
Итого: 550 руб.			

Накладные расходы составляют 550 рублей.

12.3.6 Сводная смета затрат

На основании всех произведённых расчётов составим сводную смету затрат на выполнение работы в виде таблицы 12.7.

Таблица 12.7 – Сводная смета затрат

Наименование статей затрат	Всего, руб.
ФОТ со страховыми взносами	15978,14
Основные и вспомогательные материалы	300
Амортизационные отчисления	700
Затраты на электроэнергию	248,25
Накладные расходы	550
Итого: 17776,39 руб.	

Итого общая сумма затрат на выполнение дипломной работы составила 17776,39 рублей. Около 90% всех расходов приходится на оплату труда руководителя и исполнителя, остальные 10% — на прочие расходы, включающие в себя амортизацию, расходы на электроэнергию, основные и вспомогательные материалы, а также накладные расходы.

13 Вопросы охраны труда и безопасности жизнедеятельности

13.1 Анализ опасных и вредных производственных факторов на рабочем месте

В ходе трудового процесса организм человека может подвергаться различным воздействиям, оказывающим влияние на его здоровье и работоспособность. Подобное воздействие может приводить к различным результирующим последствиям, которые зависят от характера воздействия (прямого или опосредованного), наличия тех или иных факторов производственной среды, а также условий их проявления.

Принято разграничивать производственные факторы на две основные группы — опасные производственные факторы (ОПФ) и вредные производственные факторы (ВПФ). При этом однозначно отнести тот или иной фактор к подмножеству опасных или вредных не всегда представляется возможным, поскольку даже нейтральные производственные факторы при наличии определенных условий и обстоятельств могут становиться вредными или опасными для человека, приводить к травмам и заболеваниям, связанным с трудовой деятельностью.

При работе за ПЭВМ воздействие на организм человека носит физическую природу. Кроме того работники подвергаются нервно-психическим перегрузкам.

Можно выделить следующие физические факторы, связанные с работой за ПЭВМ:

– факторы шума;
– температура, влажность и подвижность воздуха рабочей зоны;
– значение напряжения в электрической цепи;
– уровень статического электричества и электромагнитных излучений;
– отсутствие или недостаток естественного и искусственного освещения;
– повышенная яркость, пульсация света.

К нервно-психическим перегрузкам относят:

– умственное перенапряжение, в том числе вызванное информационной перегрузкой;

– монотонность труда;

– эмоциональные перегрузки [35].

Работа за персональной электронно-вычислительной машиной (ПЭВМ) относится к категории Iа: с интенсивностью энерготрат до 120 ккал/ч (до 139 Вт), производимая сидя и сопровождающаяся незначительным физическим напряжением [36]. Требования к такого рода рабочим местам устанавливает СанПин 2.2.2/2.4.1340-031 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы» [37].

Перечень продукции и контролируемых гигиенических параметров огласно [37] приведен в таблице 13.1.

Таблица 13.1 – Перечень продукции, контролируемых гигиенических параметров

Вид продукции	Контролируемые гигиенические параметры
Машина вычислительная электронная цифровая персональная (ПЭВМ)	– уровни электромагнитных полей (ЭМП); – уровни акустического шума; – концентрация вредных веществ в воздухе; – визуальные показатели видеодисплейного терминала
Устройства периферийные: модем, клавиатура, устройства хранения информации	– уровни ЭМП; – уровни акустического шума; – концентрация вредных веществ в воздухе

13.3 Требования к уровням электромагнитных полей на рабочих местах, оборудованных ПЭВМ

Временные допустимые уровни ЭМП, создаваемых ПЭВМ на рабочих местах пользователей представлены в таблице 13.2 [37].

Таблица 13.2 – Временные допустимые уровни ЭМП, создаваемых ПЭВМ на рабочих местах

Наименование параметров		ВДУ
Напряженность электрического поля	в диапазоне частот 5 Гц – 2 кГц	25 В/м
	в диапазоне частот 2 кГц – 400 кГц	2,5 В/м
Плотность магнитного потока	в диапазоне частот 5 Гц - 2 кГц	250 нТл
	в диапазоне частот 2 кГц – 400 кГц	25 нТл
Напряженность электростатического поля		15 кВ/м

13.4 Требования к визуальным параметрам устройств отображения информации

Предельно допустимые значения визуальных параметров визуального дисплейного терминала, контролируемые на рабочих местах, представлены в таблице 13.3 [37].

Для дисплеев на плоских дискретных экранах (жидкокристаллических, плазменных и т.п.) и не менее 60 Гц для дисплеев частота обновления изображения должна быть не менее 75 Гц при всех режимах разрешения экрана, гарантируемых нормативной документацией на конкретный тип дисплея [37].

13.5 Требования к микроклимату. Концентрации вредных веществ, выделяемых ПЭВМ в воздух помещения

Показатели микроклимата должны обеспечивать сохранение теплового баланса человека с окружающей средой и поддержание оптимального или допустимого теплового состояния организма. Несоблюдение оптимальных микроклиматических

Таблица 13.3 – Визуальные параметры устройств отображения информации

Параметры	Допустимые значения
Яркость белого поля	Не менее 35 кд/кв.м
Неравномерность яркости рабочего поля	Не более $\pm$ 20%
Контрасность (для монохромного режима)	Не менее 3:1
Временная нестабильность изображений (непреднамеренное изменение во времени яркости изображения на экране дисплея)	Не должна фиксироваться
Пространственная нестабильность изображения (непреднамеренные изменения положения фрагментов изображения на экране)	Не более $2 \times 1E(-4L)$, где L — проектное расстояние наблюдения, мм

условий может привести к ухудшению состояния здоровья, снижению работоспособности, ощущению дискомфорта, напряжению механизмов терморегуляции.

Работа за ПЭВМ относится к категории Iа: с интенсивностью энерготрат до 120 ккал/ч (до 139 Вт), производимая сидя и сопровождающаяся незначительным физическим напряжением [36].

Согласно [36] допустимые величины показателей микроклимата на рабочих местах производственных помещений категории Iа должны соответствовать значениям, приведенным в таблице 13.4.

Амплитуда колебания температуры воздуха в течение смены не должна превышать 4° С.

Таблица 13.4 – Оптимальные величины показателей микроклимата на рабочих местах производственных помещений категории Iа

Период года	Температура воздуха, °С	Температура поверхностей, °С	Относительная влажность воздуха	Скорость движения воздуха, м/с
Холодный	22-24	21-25	60-40	0,1
Теплый	23-25	22-26	60-40	0,1

13.6 Требования к освещению на рабочих местах, оборудованных ПЭВМ

Согласно [37], к освещению рабочих мест, оборудованных ПЭВМ, представляются следующие требования:

1) Рабочие столы следует размещать таким образом, чтобы видеодисплейные терминалы были ориентированы боковой стороной к световым проемам, чтобы естественный свет падал преимущественно слева.

2) Искусственное освещение в помещениях для эксплуатации ПЭВМ должно осуществляться системой общего равномерного освещения. В производственных и административно-общественных помещениях, в случаях преимущественной работы с документами, следует применять системы комбинированного освещения (к общему освещению дополнительно устанавливаются светильники местного освещения, предназначенные для освещения зоны расположения документов).

3) Освещенность на поверхности стола в зоне размещения рабочего документа должна быть 300-500 лк. Освещение не должно создавать бликов на поверхности экрана. Освещенность поверхности экрана не должна быть более 300 лк.

4) Следует ограничивать прямую блесткость от источников освещения, при этом яркость светящихся поверхностей (окна, светильники и др.), находящихся в поле зрения, должна быть не более 200 кд/м^2.

5) Следует ограничивать отраженную блесткость на рабочих поверхностях (экран, стол, клавиатура и др.) за счет правильного выбора типов светильников и расположения рабочих мест по отношению к источникам естественного и искусственного освещения, при этом яркость бликов на экране ПЭВМ не должна превышать 40 кд/м^2 и яркость потолка не должна превышать 200 кд/м^2.

6) Показатель ослепленности для источников общего искусственного освещения в производственных помещениях должен быть не более 20. Показатель дискомфорта в административно-общественных помещениях не более 40, в дошкольных и учебных помещениях не более 15.

7) Яркость светильников общего освещения в зоне углов излучения от 50 до 90° с вертикалью в продольной и поперечной плоскостях должна составлять не более 200 кд/м^2, защитный угол светильников должен быть не менее 40°.

8) Светильники местного освещения должны иметь непросвечивающий отражатель с защитным углом не менее 40°.

9) Следует ограничивать неравномерность распределения яркости в поле зрения пользователя ПЭВМ, при этом соотношение яркости между рабочими поверхностями не должно превышать 3:1-5:1, а между рабочими поверхностями и поверхностями стен и оборудования 10:1.

10) Общее освещение при использовании люминесцентных светильников следует выполнять в виде сплошных или прерывистых линий светильников, расположенных сбоку от рабочих мест, параллельно линии зрения пользователя при рядном расположении видеодисплейных терминалов. При периметральном расположении компьютеров линии светильников должны располагаться локализованно над рабочим столом ближе к его переднему краю, обращенному к оператору.

11) Коэффициент запаса (Кз) для осветительных установок общего освещения должен приниматься равным 1,4.

12) Коэффициент пульсации не должен превышать 5%.

13) Для обеспечения нормируемых значений освещенности в помещениях для использования ПЭВМ следует проводить чистку стекол оконных рам и светильников не реже двух раз в год и проводить своевременную замену перегоревших ламп.

13.7 Требования к организации рабочих мест пользователей ПЭВМ

Организация и оборудование рабочих мест с ПЭВМ для взрослых пользователей согласно [37] включает в себя следующие требования:

1) Высота рабочей поверхности стола для взрослых пользователей должна регулироваться в пределах 680-800 мм; при отсутствии такой возможности высота рабочей поверхности стола должна составлять 725 мм.

2) Модульными размерами рабочей поверхности стола для ПЭВМ, на основании которых должны рассчитываться конструктивные размеры, следует считать: ширину 800, 1000, 1200 и 1400 мм, глубину 800 и 1000 мм при нерегулируемой его высоте, равной 725 мм.

3) Рабочий стол должен иметь пространство для ног высотой не менее 600 мм, шириной — не менее 500 мм, глубиной на уровне колен — не менее 450 мм и на уровне вытянутых ног - не менее 650 мм.

4) Конструкция рабочего стула должна обеспечивать:

– ширину и глубину поверхности сиденья не менее 400 мм;

– поверхность сиденья с закругленным передним краем;

– регулировку высоты поверхности сиденья в пределах 400-550 мм и углам наклона вперед до 15° и назад до 5°;

– высоту опорной поверхности спинки 300±20 мм, ширину — не менее 380 мм и радиус кривизны горизонтальной плоскости — 400 мм;

– угол наклона спинки в вертикальной плоскости в пределах ±30°;

– регулировку расстояния спинки от переднего края сиденья в пределах 260-400 мм;

– стационарные или съемные подлокотники длиной не менее 250 мм и шириной — 50-70 мм;

– регулировку подлокотников по высоте над сиденьем в пределах 230±30 мм и внутреннего расстояния между подлокотниками в пределах 350-500 мм.

5) Рабочее место пользователя ПЭВМ следует оборудовать подставкой для ног, имеющей ширину не менее 300 мм, глубину не менее 400 мм, регулировку по высоте в пределах до 150 мм и по углу наклона опорной поверхности подставки до 20°. Поверхность подставки должна быть рифленой и иметь по переднему краю бортик высотой 10 мм.

6) Клавиатуру следует располагать на поверхности стола на расстоянии 100-300 мм от края, обращенного к пользователю, или на специальной, регулируемой по

высоте рабочей поверхности, отделенной от основной столешницы.

13.8 Оценка соответсвия автоматизированного рабочего места требованиям СанПин 2.2.2/2.4.1340-031 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы»

Автоматизированное рабочее место исполнителя оборудовано ПЭВМ (ноутбук) и жидкокристаллическим видеодисплейным терминалом, расположено в помещении, где регулярно производятся влажная и сухая уборка, проветривание, а также поддерживается оптимальная температура и влажность воздуха. В помещении большое окно, выходящее на восток, что обеспечивает поступление большого количества естественного света в рабочие часы. Оконный проем оборудован шторами. Искусственное освещение обеспечивается лампами накаливания, а также настольной люминесцентной лампой. В помещении отсутствует шумящее оборудование.

В позволяет регулировать яркость и контрастность изображения и расположен таким образом, что естественный свет падает слева от экрана. Экран видеомонитора находится от глаз пользователя на расстоянии 500 мм.

Поверхность сиденья и спинки стула оборудована полумягким нескользящим покрытием, позволяет регулировать высоту сиденья.

Рассчитаем показатели освещенности E на рабочем месте с помощью следующих формул [38]:

$$E = \frac{I}{r^2} cos(i),$$

где I — сила света, кд;

i – угол падения лучей света относительно нормали к поверхности;

r – расстояние до источника света, м.

Формула для вычисления силы света:

$$I = \frac{F}{4\pi},$$

где F — номинальный световой поток, лм.

Помещение оборудовано 1-ой люменисцентной лампой и 3-мя лампами накаливания. Световой поток люменисцентной лампы мощностью 10 Вт составляет около 400 Лм, одной лампы накаливания мощностью 60 Вт — 710 Лм.

Сила света люменисцентной лампы:

$$I_1 = \frac{400}{4\pi} = 31,83 \text{ кд.}$$

Сила света ламп накаливания:

$$I_2 = \frac{3 \times 710}{4\pi} = 169,5 \text{ кд.}$$

Показатель освещенности на поверхности стола составляет:

$$E_1 = \frac{31,83}{(0,3)^2} cos(30^\circ) + \frac{169,5}{(1,5)^2} cos(45^\circ) = 359,56 \text{ лк.}$$

Показатель освещенности на поверхности экрана составляет:

$$E_2 = \frac{31,83}{(0,3)^2} cos(60^\circ) + \frac{169,5}{(1,6)^2} cos(60^\circ) = 209,94 \text{ лк.}$$

По результатам расчетов можно сделать вывод, что показатели освещенности соответствуют нормам, принятым в [37]: освещенность поверхности стола в диапазоне 300-500 лк, на поверхности экрана не превышает 300 лк.

«Суммарное время регламентированных перерывов при 8-часовой рабочей смене для категории работ Iа составляет 50 минут. В случаях, когда характер работы требует постоянного взаимодействия с ВДТ (набор текстов или ввод данных и т.п.) с напряжением внимания и сосредоточенности, при исключении возможности периодического переключения на другие виды трудовой деятельности, не связанные с ПЭВМ, рекомендуется организация перерывов на 10 - 15 мин. через каждые 45 - 60 мин. работы. Продолжительность непрерывной работы с ВДТ без регламентированного перерыва не должна превышать 1 ч.» [37].

В процессе работы за ПЭВМ каждые 45 минут проводились 10-15 минутные перерывы во избежание психо-эмоциональной перегрузки и для снятия физического

напряжения. Во время перерывов осуществлялись легкая физическая разминка, а также упражнения для глаз.

В результате анализа можно сделать вывод, что организация рабочего места, на котором выполнялась дипломная работа, удовлетворяет перечисленным выше требованиям правильной организации рабочего места оператора ЭВМ. Так как и остальные условия работы в помещении являются удовлетворительными (микроклимат, освещение и т.д.), данное рабочее место работника можно считать соответствующим общим эргономическим требованиям.

Заключение

В ходе дипломной работы были выполнены все поставленные задачи:

– проведен обзор актуальных информационных источников в области деанонимизации авторов программного обеспечения по его исходному коду;

– построена модель процесса определения авторства исходного кода;

– сформирован и обработан набор данных, состоящий из трех подвыборок, имеющих характерные особенности, которые оказывают влияние на точность классификации;

– произведена программная реализация разработанной модели и ее тестирование;

– разработан интерфейс на основе технологии Jupyter Notebook;

– проведены вычислительные эксперименты;

– произведен анализ полученных результатов;

– приведено технико-экономическое обоснование работы;

– рассмотрены вопросы безопасности жизнедеятельности.

Итогом дипломной работы является разработанное программное обеспечение «WhoseCppCode» на языке программирования Python, предназначенное для сбора и обработки данных, классификации авторов исходного кода на языке C/C++, визуализации полученных результатов при помощи интерфейса. Основной программный модуль может быть использован отдельно при разработке различных систем, интерфейсов и программ, а также дальнейших научных исследований задачи деанонимизации авторов исходного кода.

Перечень основных терминов и определений

Ниже преведены определения основных понятий, использованных в настоящем документе:

– исходный код — текстовый файл, содержащий текст компьютерной программы на каком- либо языке программирования и визуально понятный человеку;

– стилометрия — исследование стилистики, включающее статистический анализ текста на предмет признаков, характеризующих индивидуальный стиль автора;

– классификация — процесс отнесения исследуемой сущности к какому-либо заданному классу;

– кросс-валидация — процедура, применяемая в машинном обучении для проверки, насколько успешно построенная аналитическая модель может работать на независимом наборе данных;

– open-source — концепция свободного (открытого) программного обеспечения, т.е. программ с исходным кодом, открытым для использования и изменения сторонними разработчиками;

– репозиторий — удаленное хранилище файлов исходного кода и данных;

– контрибьютор — пользователь, вносивший изменения в open-source проект.

Список использованных источников

1 Google C++ Style Guide [Электронный ресурс]. — Режим доступа: https://google.github.io/styleguide/cppguide.html, свободный (дата обращения: 28.04.2017).

2 C++ Programming Style Guidelines [Электронный ресурс]. — Режим доступа: http://geosoft.no/development/cppstyle.html, свободный (дата обращения: 28.04.2017).

3 Source Code Authorship Analysis For Supporting The Cybercrime Investigation Process. Georgia Frantzeskou, Efstathios Stamatatos, Stefanos Gritzalis [Электронный ресурс]. — Режим доступа: http://www.icsd.aegean.gr/lecturers/stamatatos/papers/ICETE2005.pdf, свободный (дата обращения: 17.02.2017).

4 Identifying Authorship by Byte-Level N-Grams: The Source Code Author Profile (SCAP) Method. Georgia Frantzeskou, Efstathios Stamatatos, Stefanos Gritzalis [Электронный ресурс]. — Режим доступа: https://www.semanticscholar.org/paper/Identifying-Authorship-by-Byte-Level-N-Grams-The-Frantzeskou-Stamatatos/3b2531ea2685b9fb9abf071d119974ac3405874d, свободный (дата обращения: 15.02.2017).

5 Using Classification Techniques to Determine Source Code Authorship. Brian N. Pellin Computer Sciences Department University of Wisconsin [Электронный ресурс]. — Режим доступа: https://pdfs.semanticscholar.org/f9aa/790191a50bed02a877e1696c7bb71ea9f33a.pdf, свободный (дата обращения: 25.02.2017).

6 Source Code Authorship Attribution using n-grams. Steven Burrows, S.M.M. Tahaghoghi [Электронный ресурс]. — Режим доступа:

https://pdfs.semanticscholar.org/79a2/1998c2f0afe2c616c01d590d6d0f6e16e9eb.pdf, свободный (дата обращения: 23.02.2017).

7 Source Code Authorship Attribution. Steven Burrows [Электронный ресурс]. — Режим доступа: http://researchbank.rmit.edu.au/eserv/rmit:10828/Burrows.pdf, свободный (дата обращения: 23.02.2017).

8 Application of information retrieval techniques for source code authorship attribution. S. Burrows, A. Uitdenbogerd, T. Urpin [Электронный ресурс]. — Режим доступа: https://www.researchgate.net/publication/220787332_Application_of_Information_Retri ques_for_Source_Code_Authorship_Attribution, свободный (дата обращения: 23.02.2017).

9 De-anonymizing Programmers via Code Stylometry. Aylin Caliskan-Islam, Drexel University; Richard Harang, U.S. Army Research Laboratory; Andrew Liu, University of Maryland; Arvind Narayanan, Princeton University; Clare Voss, U.S. Army Research Laboratory; Fabian Yamaguchi, University of Goettingen; Rachel Greenstadt, Drexel University [Электронный ресурс]. — Режим доступа: https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-caliskan-islam.pdf, свободный (дата обращения: 18.02.2017).

10 Random Forest. Applied Multivariate Statistics — Spring 2012 [Электронный ресурс]. — Режим доступа: http://stat.ethz.ch/education/semesters/ss2012/ams/slides/v10.2.pdf, свободный (дата обращения: 17.02.2017).

11 Git Blame Who?: Stylistic Authorship Attribution of Small, Incomplete Source Code Fragments. Edwin Dauber, Aylin Caliskan, Richard Harang, Rachel Greenstadt [Электронный ресурс]. — Режим доступа: https://arxiv.org/abs/1701.05681, свободный (дата обращения: 10.03.2017).

12 Хэзфилд Р. Искусство программирования на С. Фундаментальные алгоритмы, структуры данных и примеры приложений. Энциклопедия программиста / Р. Хэзфилд, Л. Кирби. — М: . ДиаСофт, 2001. – 736 с.

13 Макросы (C/C++) [Электронный ресурс]. — Режим доступа: https://msdn.microsoft.com/ru-ru/library/503x3e3s.aspx, свободный (дата обращения: 23.02.2017).

14 Ключевые слова C++ [Электронный ресурс]. — Режим доступа: http://ru.cppreference.com/w/cpp/keyword, свободный (дата обращения: 12.03.2017).

15 Методы классификации и прогнозирования. Деревья решений. ИНТУИТ [Электронный ресурс]. — Режим доступа: http://www.intuit.ru/studies/courses/6/6/lecture/1740, свободный (дата обращения: 01.05.2017).

16 Алгоритм AdaBoost [Электронный ресурс]. — Режим доступа: http://www.machinelearning.ru/wiki/index.php?title=AdaBoost, свободный (дата обращения: 27.04.2017).

17 Extremely randomized trees. Pierre Geurts, Damien Ernst, Louis Wehenkel [Электронный ресурс]. — Режим доступа: https://pdfs.semanticscholar.org/336a/165c17c9c56160d332b9f4a2b403fccbdbfb.pdf, свободный (дата обращения: 28.04.2017).

18 Перекрёстная проверка [Электронный ресурс]. — Режим доступа: https://ru.wikipedia.org/wiki/, свободный (дата обращения: 01.05.2017).

19 Code Jam Language Stats [Электронный ресурс]. — Режим доступа: https://www.go-hero.net/jam/16, свободный (дата обращения: 10.02.2017).

20 GitHub [Электронный ресурс]. — Режим доступа: https://github.com/, свободный (дата обращения: 10.05.2017).

21 GitHub Dominates the Forges [Электронный ресурс]. — Режим доступа: https://github.com/blog/865-github-dominates-the-forges, свободный (дата обращения: 10.05.2017).

22 Оценка классификатора (точность, полнота, F-мера) [Электронный ресурс]. — Режим доступа: http://bazhenov.me/blog/2012/07/21/classification-performance-evaluation.html, свободный (дата обращения: 17.02.2017).

23 The Jupyter Notebook [Электронный ресурс]. — Режим доступа: http://jupyter.org/, свободный (дата обращения: 02.04.2017).

24 Scikit-learn — Machine Learning in Python [Электронный ресурс]. — Режим доступа: http://scikit-learn.org/stable/, свободный (дата обращения: 02.03.2017).

25 Plotly Python Library [Электронный ресурс]. — Режим доступа: https://plot.ly/python/, свободный (дата обращения: 25.02.2017).

26 NumPy [Электронный ресурс]. — Режим доступа: http://www.numpy.org/, свободный (дата обращения: 25.02.2017).

27 SciPy [Электронный ресурс]. — Режим доступа: https://www.scipy.org/, свободный (дата обращения: 25.02.2017).

28 Python Data Analysis Library [Электронный ресурс]. — Режим доступа: http://pandas.pydata.org/, свободный (дата обращения: 25.02.2017).

29 ipywidgets: Interactive HTML Widgets [Электронный ресурс]. — Режим доступа: https://github.com/jupyter-widgets/ipywidgets, свободный (дата обращения: 09.03.2017).

30 О классификации основных средств, включаемых в амортизационные группы: постановление Правительства РФ № 1 от 1 января 2002 г. [Электронный ресурс]. — Режим доступа: http://www.consultant.ru/document/cons_doc_LAW_34710/, свободный (дата обращения: 29.05.2017).

31 Т.А. Фролова. Экономика предприятия: конспект лекций [Электронный ресурс]. — Режим доступа: http://www.aup.ru/books/m203/2_5.htm, свободный (дата обращения: 6.06.2017).

32 НК РФ Статья 426. Тарифы страховых взносов в 2017 – 2019 годах [Электронный ресурс]. — Режим доступа: http://www.consultant.ru/document/cons_doc_LAW_28165/994c7ea6856fd572dd88ab72d25b2f8d9bc99dd3/, свободный (дата обращения: 6.06.2017).

33 ТК РФ, Статья 151. Оплата труда при совмещении профессий (должностей), расширении зон обслуживания, увеличении объема работы или исполнении обязанностей временно отсутствующего работника без освобождения от работы, определенной трудовым договором [Электронный ресурс]. — Режим доступа: http://www.consultant.ru/document/cons_doc_LAW_34683/2776d76376c800a4693157045be7f028798f5046/, свободный (дата обращения: 6.06.2017).

34 Приказ от 23.12.2016 г. № 6-840 «О тарифах на электрическую энергию для населения и потребителей, приравненных к категории население по Томской области на 2017 год» [Электронный ресурс]. — Режим доступа: http://energovopros.ru/spravochnik/elektrosnabzhenie/tarify-na-elektroenergiju/tomskaya_oblast/39310/, свободный (дата обращения: 29.05.2017).

35 ГОСТ 12.0.003-2015 Система стандартов безопасности труда (ССБТ). Опасные и вредные производственные факторы. Классификация [Электронный ресурс]. — Режим доступа: http://meganorm.ru/Data2/1/4293754/4293754317.pdf, свободный (дата обращения: 31.05.2017).

36 СанПиН 2.2.4.548-96 Гигиенические требования к микроклимату производственных помещений (утв. постановлением Госкомсанэпиднадзора РФ от 1 октября 1996 г. № 21) [Электронный ресурс]. — Режим доступа: http://www.vashdom.ru/sanpin/224548-96/, свободный (дата обращения: 31.05.2017).

37 СанПиН 2.2.2/2.4.1340-03 Гигиенические требования к персональным электронно-вычислительным машинам и организации работы [Электронный ресурс]. — Режим доступа: http://docs.cntd.ru/document/901865498, свободный (дата обращения: 31.05.2017).

38 ГОСТ ИСО 8895-2002 Освещение рабочих систем внутри помещений [Электронный ресурс]. — Режим доступа: http://www.internet-law.ru/gosts/gost/5955/, свободный (дата обращения: 31.05.2017).

Приложение А

(Справочное)

Сравнительный обзор информационных источников

Таблица А.1 — Обзор источников

Название работы	Авторы, год публикации	Методы, использованные в работе	Описание данных	Достигнутая точность классификации	Язык программирования
Using classification techniques to determine source code authorship [5]	B. Pellin, 2008	ACT, SVM	4 схожие программы, 2 автора	67 — 88 %	Java
Source code authorship attribution using n-grams [6]	S. Burrows, S. Tahaghoghi, 2007	N-граммы	Выборка из 1640 файлов исходного кода и 100 авторов	67 %	C
Identifying Authorship by Byte-Level N-Grams: The Source Code Author Profile (SCAP) Method [4]	G. Frantzeskou, E. Stamatatos, S. Gritzalis, 2007	Составление профиля программиста на основе статистических метрик, подсчет отклонения от профиля	Не указано	88 % для C++, 100 % для Java	Java, C++
Application of information retrieval techniques for source code authorship attribution [8]	S. Burrows, A. Uitdenbogerd, T. Urpin, 2009	N-граммы, рейтинговые схемы	100 авторов, классифицировались по 10, 1579 программных файлов	77 %	C

Продолжение таблицы А.1

Название работы	Авторы, год публикации	Методы, использованные в работе	Описание данных	Достигнутая точность классификации	Язык программи-рования
De-anonymizing Programmers via Code Stylometry [9]	A. Caliskan-Islam, R. Harang, A. Liu, F. Yamaguchi, 2015	Статистический подсчет признаков, нечеткие АСТ	250 авторов, 1600 файлов	94 — 98 %	C/C++, Python
Git Blame Who?: Stylistic Authorship Attribution of Small, Incomplete Source Code Fragments [11]	A. Caliskan-Islam, E. Dau-ber, R. Harang, R. Greenstadt, 2017	Калибровочные кривые, нечеткие АСТ, классифика-тор Random Forest	Некомпили-руемые неполные образцы кода с ресурса GitHub	70 — 100 %	C/C++