МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка веб-приложения для бинарной классификации вредоносных команд по метрике MITRE с использованием алгоритмов машинного обучения

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.04.2023.308-338.ВКР

Научный руководитель,
ст. преподаватель кафедры СП
_____ К.Ю. Никольская

Автор работы,
студент группы КЭ-403
_____ М.Д. Григорьев

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-403
Григорьеву Максиму Дмитриевичу,
обучающемуся по направлению
09.03.04 «Программная инженерия»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12) Разработка веб-приложения для бинарной классификации вредоносных команд по метрике MITRE с использованием алгоритмов машинного обучения.
2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
3. **Исходные данные к работе**

3.1. Ahn G., Kim K., Park W., Shin D. Malicious File Detection Method Using Machine Learning and Interworking with MITRE ATT&CK Framework. // Appl. Sci., 2022. – 21 с.

3.2. MITRE ATT&CK. [Электронный ресурс] URL: https://attack.mitre.org/ (дата обращения: 07.02.2024 г.).

3.3. Введение в Data Science и машинное обучение. [Электронный ресурс] URL: https://stepik.org/course/4852/syllabus (дата обращения: 07.02.2024 г.).

4. **Перечень подлежащих разработке вопросов**

4.1. Провести обзор научной литературы.

4.2. Подготовить обучающий набор данных.

4.3. Реализовать выбранные методы машинного обучения.

4.4. Разработать веб-приложение классификации вредоносных команд по метрике MITRE.

4.5. Провести тестирование разработанного веб-приложения.

5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
ст. преподаватель кафедры СП К.Ю. Никольская

Задание принял к исполнению М.Д. Григорьев

ГЛОССАРИЙ

1. *Машинное обучение* – это раздел информатики, посвященный созданию алгоритмов, опирающихся на набор данных о каком-либо явлении [1].

2. *MITRE ATT&CK (Adversarial Tactics, Techniques and Common Knowledge)* – база знаний, которая содержит описание различных тактик и методов, которые используются хакерами и злоумышленниками для выполнения кибератак [2].

3. *Модель (модель машинного обучения)* – это алгоритмическая конструкция или математический объект, созданный на основе алгоритма обучения, который позволяет компьютерной системе обрабатывать данные и делать предсказания или принимать решения без явного программирования [10].

4. *Набор данных* – это коллекция структурированных или неструктурированных данных, используемых для анализа, исследования и обучения моделей машинного обучения, статистики или других аналитических целей [10].

5. *Регрессия* – это задача прогнозирования метки с действительным значением для образца без метки [10].

6. *Гиперпараметр* – это свойство алгоритма обучения, обычно (но не всегда) имеющее числовое значение. Это значение влияет на работу алгоритма. Гиперпараметры не вычисляются алгоритмом. Они должны задаваться аналитиком перед запуском алгоритма [10].

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

Информационная безопасность по-прежнему остается важной проблемой для организаций и государственных учреждений в наше время. С развитием технологий и распространением интернета, угрозы кибербезопасности стали все более распространенными. Киберпреступники могут использовать различные методы для получения доступа к конфиденциальным данным или повреждения информационных систем.

Машинное обучение является одной из наиболее актуальных и быстро развивающихся областей информационных технологий в настоящее время.

Одно из главных преимуществ машинного обучения заключается в том, что оно может использоваться для анализа большого объема данных, которые не могут быть обработаны человеком без помощи вычислительной техники.

В рамках выполнения выпускной квалификационной работы была поставлена задача исследовать возможность применения методов машинного обучения для решения задачи классификации вредоносных команд.

Постановка задачи

Целью выпускной квалификационной работы является разработка веб-приложения для бинарной классификации вредоносных команд по метрике MITRE с использованием алгоритмов машинного обучения. Для достижения поставленной цели необходимо решить следующие задачи:

1) провести обзор научной литературы;
2) подготовить обучающий набор данных;
3) реализовать выбранные методы машинного обучения;
4) разработать веб-приложение классификации вредоносных команд по метрике MITRE;
5) провести тестирование разработанного приложения.

Структура и содержание работы

Работа состоит из введения, четырех глав, заключения и списка литературы. Объем работы составляет 67 страниц, объем списка литературы – 26 источников.

В первой главе производится анализ научной литературы и описывается предметная область.

Вторая глава посвящена проектированию веб-приложения и моделей машинного обучения. Также, в этой главе определены функциональные и нефункциональные требования к приложению, а также приведены диаграмма компонентов и вариантов использования системы.

В третьей главе рассказывается о способах и средствах реализации веб-приложения и моделей машинного обучения. Также, в данной главе рассказывается, как проводилось обучение моделей машинного обучения. Также рассказывается, как используется Docker контейнер.

В четвертой главе описываются способы и результаты тестирования веб-приложения, докер контейнера и моделей машинного обучения. Также, показаны результаты функционального тестирования для некоторых страниц приложения.

В приложении содержится спецификация вариантов использования системы.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1. Обзор научной литературы

«Data Mining Applied to Intrusion Detection: MITRE Experiences» [1]

В данной работе для классификации команд по классификации MITRE использовался алгоритм инкрементного классификатора с использованием domain knowledge rules (DK). Алгоритм был разработан авторами статьи, который использует обучение с учителем.

Набор данных представляет из себя отчеты, которые были собраны с различных сетевых устройств. Набор данных состоял из 8 653 ложных команд и 42 312 вредоносных команд. Каждая команда добавлялась в таблицу, в который был указано устройство, с которого снята команда, характеристики устройства, сама команда и время, когда эта команда была снята с точностью до секунды (рисунок 1).

```
<theory>                    ::= <pragma> <rule list> EOF
<pragma>                    ::= rank | percent
<rule list>                 ::= { <rule> }*
<rule>                      ::= <rating> : <simple rule>;
<rating>                    ::= { - | + } <number>
<simple rule>               ::=
    <attribute value pair> { , <attribute value pair> }*
<attribute value pair>      ::=
    <optional name> <attribute value operator> <optional name>
<optional name>             ::= <name> | *
<attribute value operator> ::= = | > | >= | < | <=
```

Рисунок 1 – Таблица построения правил

Алгоритм должен был определить: является ли команда «ложной тревогой», аномалией или же действительной атакой. Результаты представлены в таблице 1.

Таблица 1 – Результат первой итерации алгоритма

Labeled Type		Detected Type			
	Total Number	**False Alarm Number**	**False Alarm Percent**	**Anomaly Number**	**Anomaly Percent**
False Alarm	8 653	7 876	91,02%	777	8,98%
Attack	42 312	5 336	12,61%	36 976	87,39%

Эти результаты использовали для последующий итераций алгоритма с использованием DK. Результаты представлены в таблице 2.

Таблица 2 – Сравнение результатов алгоритма

Run	**With DK**			**No DK**		
	Error	**Time**	**№**	**Error**	**Time**	**№**
1	6,5	0,9	6	6,5	0,8	6
2	0,3	1,0	8	24,2	3,6	11
3	1,3	0,8	5	11,3	3,7	13
4	0,5	1,1	4	0,3	4,6	15
5	1,1	1,1	4	3,7	6,9	17
6	17,6	0,6	0	9,2	7,9	18
7	12,8	2,0	8	51,9	11,5	19
8	17,6	3,0	5	1,4	44,4	37
9	3,4	0,8	5	24,6	26,4	23
10	1,6	1,3	5	17,3	51,9	40
11	0,9	10,0	13	2,3	78,9	42
12	2,9	2,2	5	6,7	59,4	33
13	41,4	1,5	9	50,7	117,7	53
14	0	0,1	0	60,0	127,9	49
15	0	0	0	0,1	131,5	51
16	0	0,2	0	2,5	137,2	51
17	0,5	0,4	0	0,8	134,1	43
18	22,0	0,6	2	20,6	177,0	46
19	33,5	0,8	12	11,2	163,3	62
20	0	0,3	7	0	213,8	61
21	0,2	0,5	0	21,1	218,4	63
22	0	0,5	3	1,1	253,3	73
23	0,8	1,3	0	1,2	286,1	69
24	11,1	1,9	5	7,6	365,8	74
25	0	2,3	10	0,1	244,0	72
Avg.	**7,0**	**1,4**	**4,6**	**13,5**	**114,8**	**41,6**

В ходе работы удалось получить точность в 91%. Результаты показывают, что этот подход имеет потенциал для точной классификации команд.

«Cyber security threat modeling based on the MITRE Enterprise ATT&CK Matrix» [2]

В данной статье рассматривается вариант противодействия злоумышленникам с помощью симуляции атак. Для этого был создан язык под названием enterpriseLang. EnterpriseLang – это доменный язык, основанный на Meta Attack Language. Meta Attack Language – это языковой фреймворк моделирования угроз состоящий из вероятностных графов атак и защиты с объектно-ориентированным моделированием, в котором описаны шаги и техники, используемые злоумышленником, по классификации MITRE. На рисунке 2 показано, из чего состоит enterpriseLang.

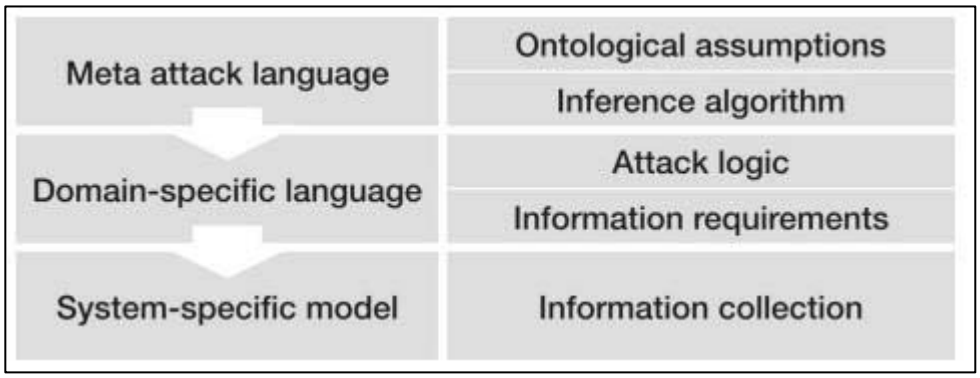

Рисунок 2 – Иерархия Meta Attack Language

Самые часто используемые символы в enterpriseLang языка Meta Attack показаны в таблице 3.

Язык enterpriseLang позволяет:

1) анализировать слабые места, связанные с известными методами атаки;

2) предоставлять предложения по смягчению последствий этих атак.

Заинтересованные в безопасности предприятия стороны могут оценить угрозы для своей корпоративной ИТ-среды и проанализировать, какие параметры безопасности можно реализовать для более эффективной защиты системы.

Таблица 3 – Символы Meta Attack Language

Symbol	Meaning	Description
–>	Leads to	Successful compromise by this attack step allows adversaries to execute further attack steps, or this defense can defend against one or more further attack steps
|	OR	An OR attack in which step A can be reached if any of the attack steps that refer to A are reached
&	AND	An AND attack in which step A can be reached only when all of the attack steps that refer to A are reached
#	Defense	Unlike attack steps, defenses are Boolean. A represents the countermeasure to an attack step. A defense can be enabled or disabled by setting the defense value to TRUE of FALSE, respectively
E	Existence	This operation is used when the existence of a connected/associated asset is checked. It acts as a defense, but the Boolean value is automatically assigned according to the existence of the asset
+>	Append	When parent and child assets have overlapping attack steps or defenses, the expressions defined for the child attack steps/defenses are appended to those of the parent attack steps/defenses
X.A	Collect operator	Attack step A on asset X is referenced
TTC	Time to compromise	TTC is a probability distribution that reflects the effort needed for an adversary to complete the related attack step. Each attack step have a local TTC, and the MAL simulations calculate/aggregate the global TTC over model/scenarios

В язык вошли 266 техник из классификации ATT&CK, описание каждой из этих техник, описание платформы, на которой будет применяться техника атаки, минимальные необходимые права, которые требуются злоумышленнику для применения техники, уровень доступа, включающий в себя права пользователя или права администратора.

После описания перечисленных выше данных, они конвертируются в Meta Attack Language. Пример этого процесса показан на рисунке 3.

Угрозы моделируются с помощью деревьев атак или графов атак. Эти методы направлены на то, чтобы показать все пути через систему, которые заканчиваются состоянием, в котором противник успешно достиг своей цели. Пример метамодели, состоящей из объектов предприятия показан на рисунке 4.

```
category Account {
  asset UserAccount {
    | userRights
        ->  windows.userAccessTokenManipulation
    # userAccountManagement
        ->  windows.userAccessTokenManipulation
  }
  asset AdminAccount {
    | adminRights
    # privilegedAccountManagement
  }
  asset WindowsAdmin extends AdminAccount {
    | adminRights
        +>  windows.adminAccessTokenManipulation
    # privilegedAccountManagement
        +>  windows.adminAccessTokenManipulation
  }
}
category Software {
  asset Windows {
    & userAccessTokenManipulation
    info: "Adversaries may use access tokens to operate
    under a different user or system security context
    to perform actions and evade detection."
        ->  service.exploitationForPrivilegeEscalation
    & adminAccessTokenManipulation
        ->  service.exploitationForPrivilegeEscalation
  }
  asset Service {
    | exploitationForPrivilegeEscalation
  }
}

associations {
UserAccount [userAccount] * <--Accesses-->
  1 [windows] Windows
AdminAccount [adminAccount] * <--Accesses-->
  1 [windows] Windows
Windows [windows] 1 <--Runs-->
  * [service] Service
}
```

Рисунок 3 – Преобразование в Meta Attack Language

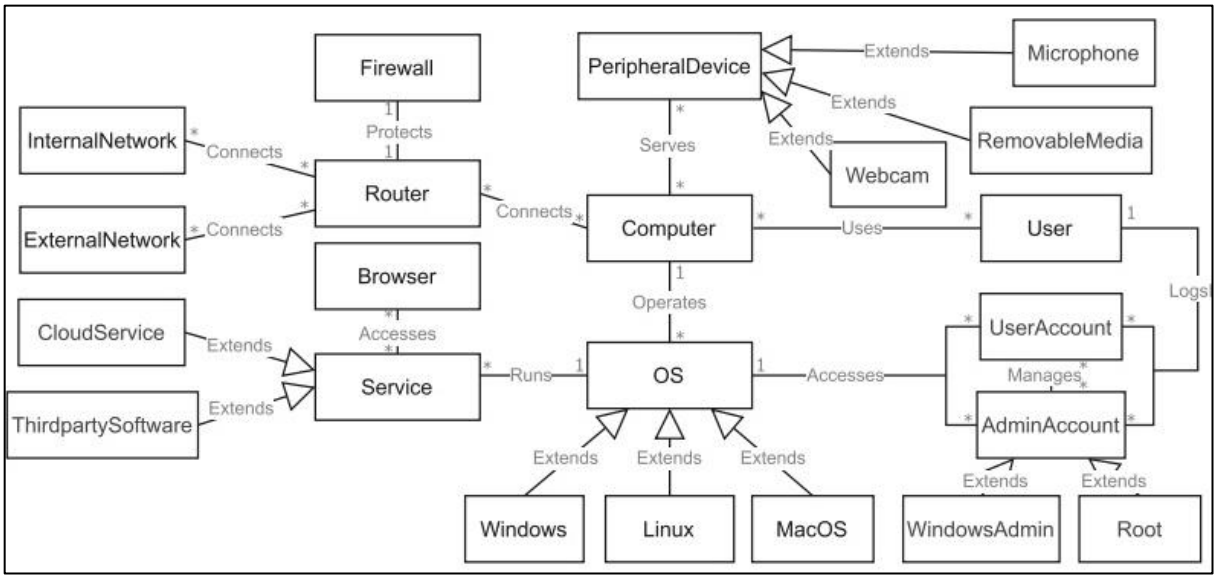

Рисунок 4 – Метамодель из объектов предприятия

В качестве данных была использована база знаний MITRE ATT&CK, которая служила основой для создания конкретных моделей угроз, методологий и инструментов.

«A Machine Learning Approach to Dataset Imputation for Software Vulnerabilities» [3]

В данной работе были созданы 12 моделей, которые способны предсказать 9 из 12 Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) категорий тактик используя только Common Attack Pattern Enumeration and Classification (CAPEC). Данные модели должны были восстановить набор данных, в котором большинство данных неполные, используя метод для замены отсутствующих данных некоторым замещающим значением, чтобы сохранить большую часть данных/информации.

Работа заключалась в следующем: модели должны были восстановить признак tactics в наборе данных «ENISA».

Набор данных «ENISA» – это набор данных уязвимостей, который содержит пропущенные значения. В декабре 2019 European Union Agency for Cybersecurity (ENISA) опубликовала доклад под названием «State of Vulnerabilities 2018/2019: Analysis of Events in the life of Vulnerabilities». Этот набор данных охватывает период опубликованных уязвимостей с 1 января 2018 г. по 31 августа 2019 г. Уязвимости были собраны и размещены в открытом наборе данных. Данные организованы в двумерную табличную структуру размером 27 471 столбцов на 59 колонок (27 471 объектов, соответственно 19 404 из которых не имели данных в признаке tactics). Из 59 столбцов те, которые содержат идентификатор уязвимости, оценки CVSS (обе версии), Common Weakness Enumeration (CWE) и количество использований, имели заполненные значения, хотя количество использований имело значение 0 в более чем 90% случаев.

Каждый объект из набора данных помечался несколькими уникальными категориями tactic из списка в таблице 4.

Таблица 4 – Характеристика исходных данных

Source type	Data type	Description
NVD database	CVE data	The NVD is the U.S. government repository of standards-based vulnerability management data. The NVD includes databases of security checklist references, security-related software flaws, misconfigurations, product names, and impact metrics®
ATT&CK	Attacker's patterns (techniques & tactics)	MITRE ATT&CK™ is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations
Shodan	Number of exploits	Database of internet-connected devices (e.g. webcams, routers, servers, etc.) acquiring data from various HTTP/HTTPS - port 80, 8080, 443, 8443)
Exploit database	Non-CVE data	Contains information on public exploits and corresponding vulnerable software. The collection of exploits is acquired from direct submissions, mailing lists and other public sources
Zero-Day Initiative	CVE and non-CVE	Encourages reporting of zero-day vulnerabilities privately to affected vendors by financially rewarding researchers (a vendor-agnostic bug bounty program). No technical details on individual vulnerabilities are made public until after vendor released patches. ZDI do not resell or redistribute the vulnerabilities’
ThreatConnect	Number of incidents related to CVE	Automated threat intelligence for Intel systems
VulDB	Exploit prices and software categories	Vulnerability database documenting and explaining security vulnerabilities and exploits
US CERT	Industry sector	The US Department for Homeland Security's Cybersecurity and Infrastructure Security Agency (CISA) aims to enhance the security, resiliency, and reliability of the USA's cybersecurity and communications infrastructure
Zerodium	Bug bounty exploit prices	A zero-day acquisition platform. Founded by cybersecurity experts with experience in advanced vulnerability research

Далее были созданы индивидуальные для каждой tactic модели, используя машинное обучение с учителем. Каждая модель была спроектирована с одинаковой архитектурой, показанной на рисунке 5.

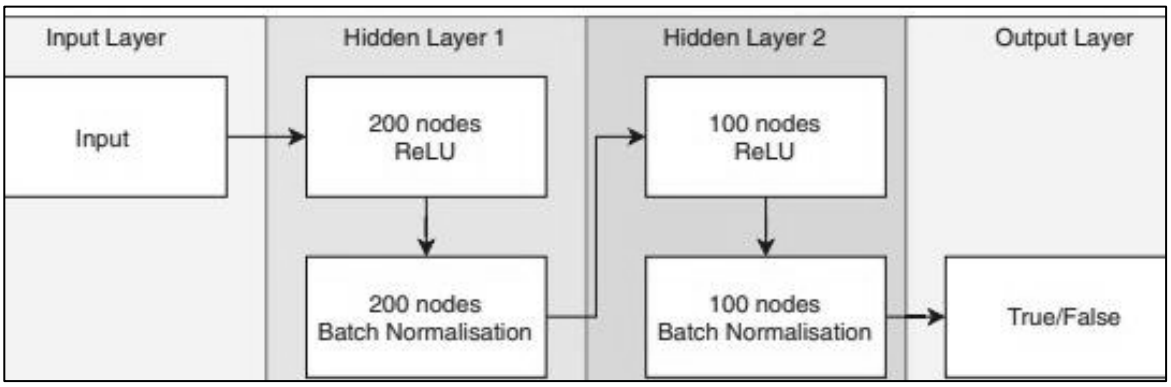

Рисунок 5 – Конфигурация архитектуры модели для каждой категории

В процессе обучения с учителем использовался оптимизатор Адам бета-коэффициенты которого были равны 0,9 и 0,99. Сокращение веса был $1 \cdot 10^{-2}$ и темп обучения $1 \cdot 10^{-3}$.

Каждая модель обучалась 5 эпох, пока не достигла 99,88% точности.

На данных для теста, модели, предсказывающие Persistence, Privilege Escalation, Defence Evasion, Credential Access, Collection достигли точности в 99,88%, а остальные – 100%. На рисунке 6 показано распределение tactic в наборе данных.

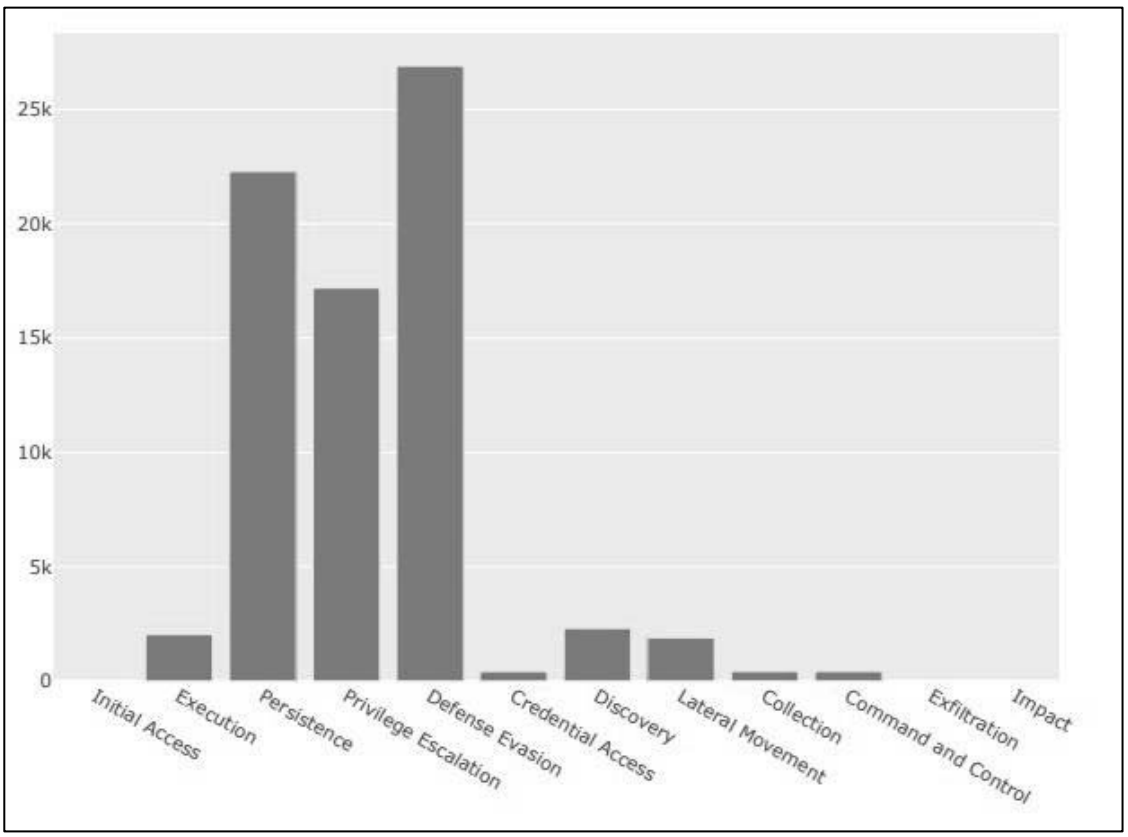

Рисунок 6 – Заполнение набора данных «ENISA»

«A Novel Enhanced Naïve Bayes Posterior Probability (ENBPP) Using Machine Learning: Cyber Threat Analysis» [4]

В данной статье рассматривался вопрос повышения точности и эффективности прогнозирования кибератак, для чего был разработан алгоритм Enhanced Naïve Bayes Posterior Probability (ENBPP). Данный алгоритм сочетает в себе 2 функции:

1) модифицированную версию Naïve Bayes posterior probability function;

2) модифицированная функция оценки риска.

Предполагалось, что сочетание этих 2 функций улучшит точность предсказания и уменьшит время.

В качестве данных использовалось 5 разных наборов данных, общее количество объектов в которых равнялось 328 814.

В итоге, алгоритм имел точность 92–96% и среднее время от 0,043 до 0,028 секунд. Средняя точность алгоритма для каждого из 5 наборов данных представлена на рисунке 7.

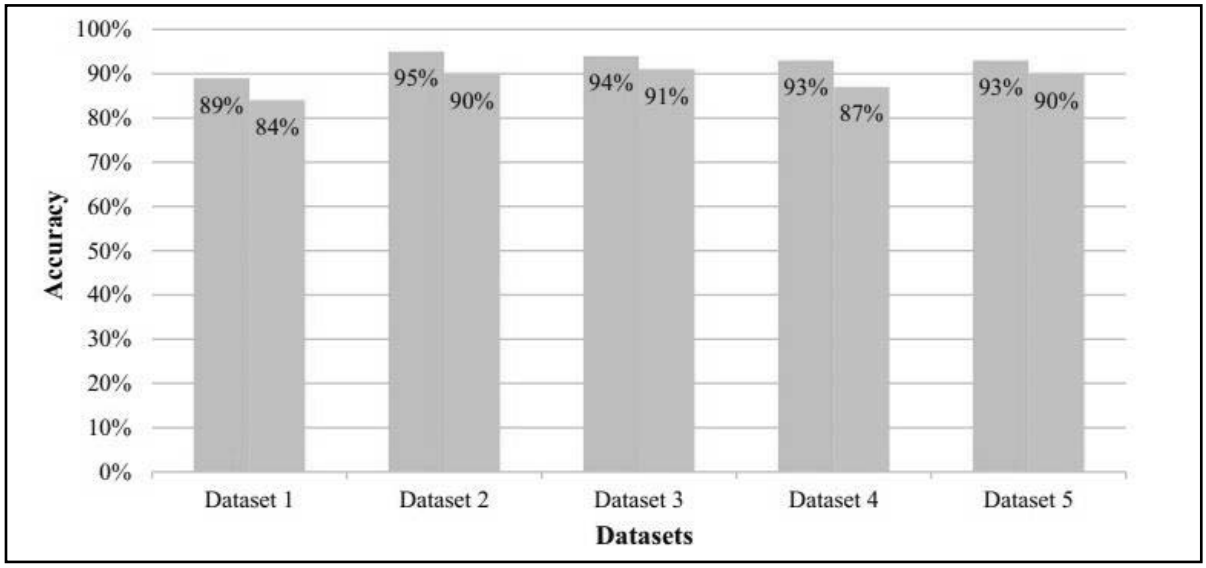

Рисунок 7 – Средняя точность алгоритма

Архитектура данной модели представлена на рисунке 8. Среднее время алгоритма для каждого из 5 наборов данных представлена на рисунке 9.

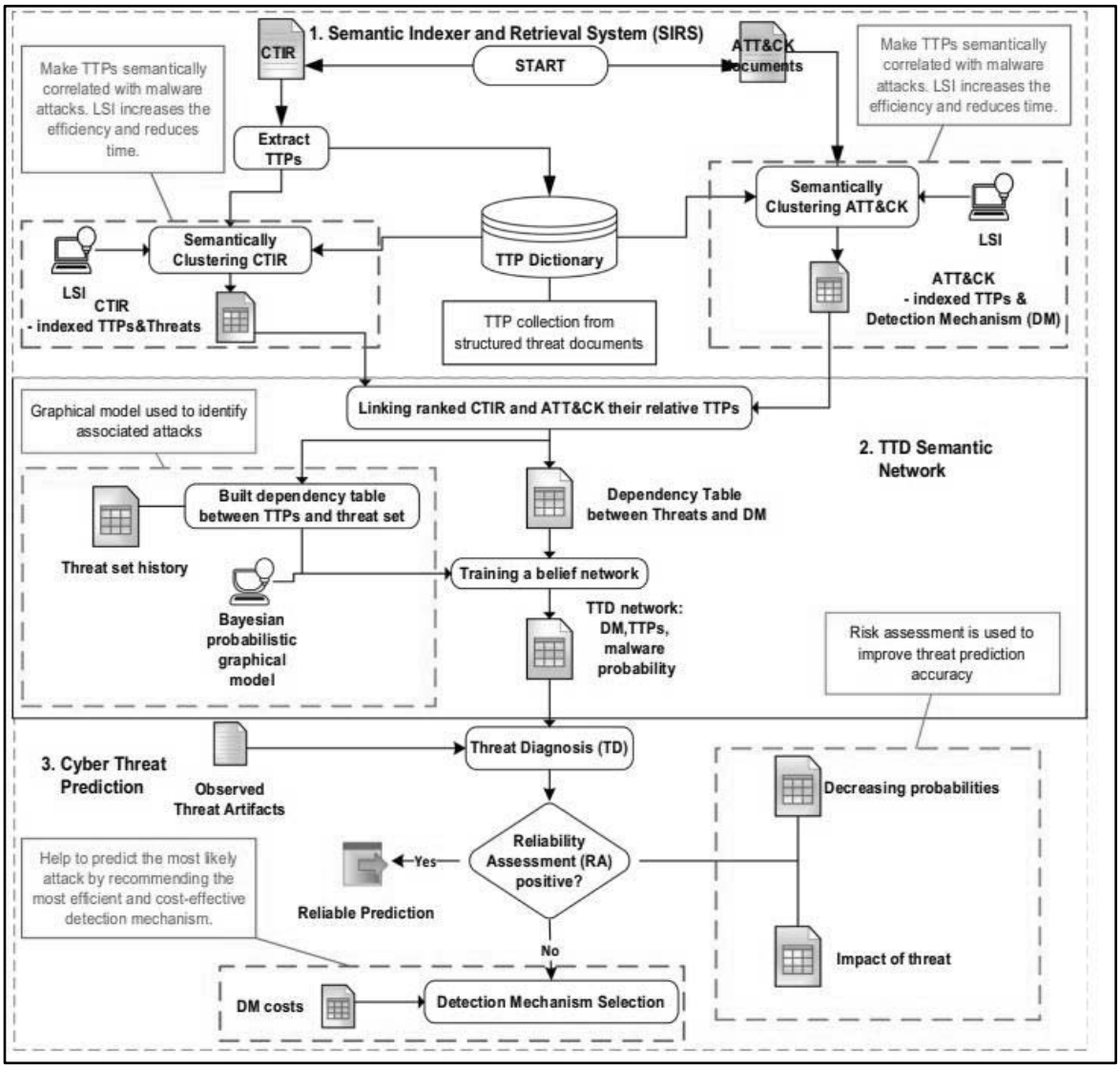

Рисунок 8 – Блок-схема предлагаемого метода прогнозирования угроз с использованием байесовского алгоритма

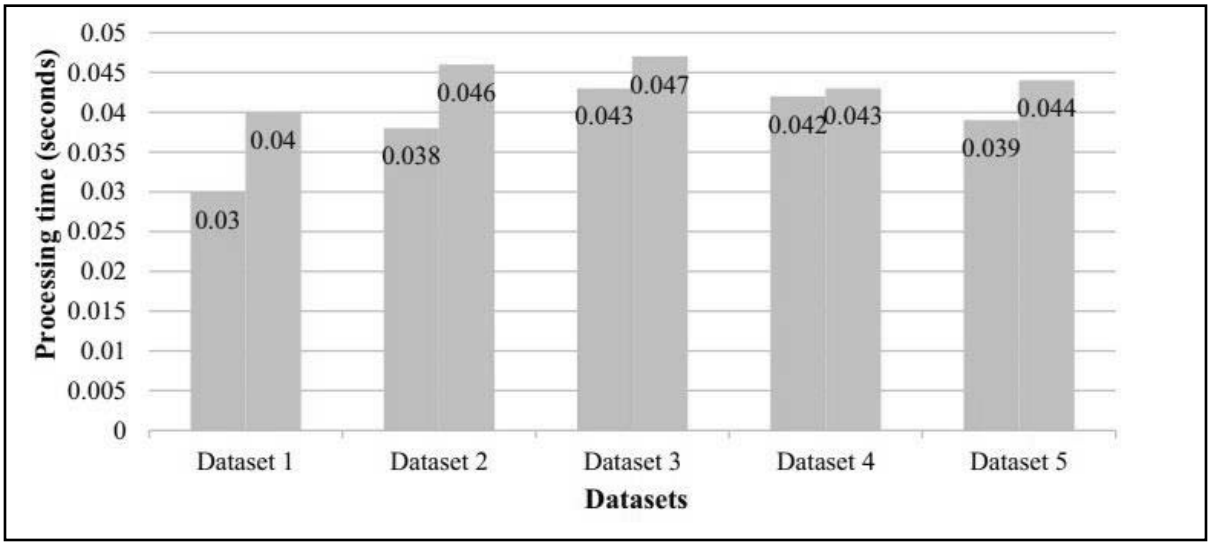

Рисунок 9 – Среднее время алгоритма

Для достижения поставленной цели был разработан новый метод, основанный на машинном обучении, который семантически извлекает угрозы и тактики, методы и процедуры атак из известных источников угроз для создания семантической сети.

Семантическая сеть устанавливает вероятностные отношения между угрозами, тактиками, методами, и процедурами, используя Naïve Bayes алгоритм машинного обучения для выявления и прогнозирования угроз. Алгоритм Naïve Bayes нормализует условную вероятность появления угроз, тактик, методов, и процедур и находит наилучший набор данных для предсказания угроз.

Стандартный байесовский алгоритм претерпел некоторые изменения, показанные на рисунке 10.

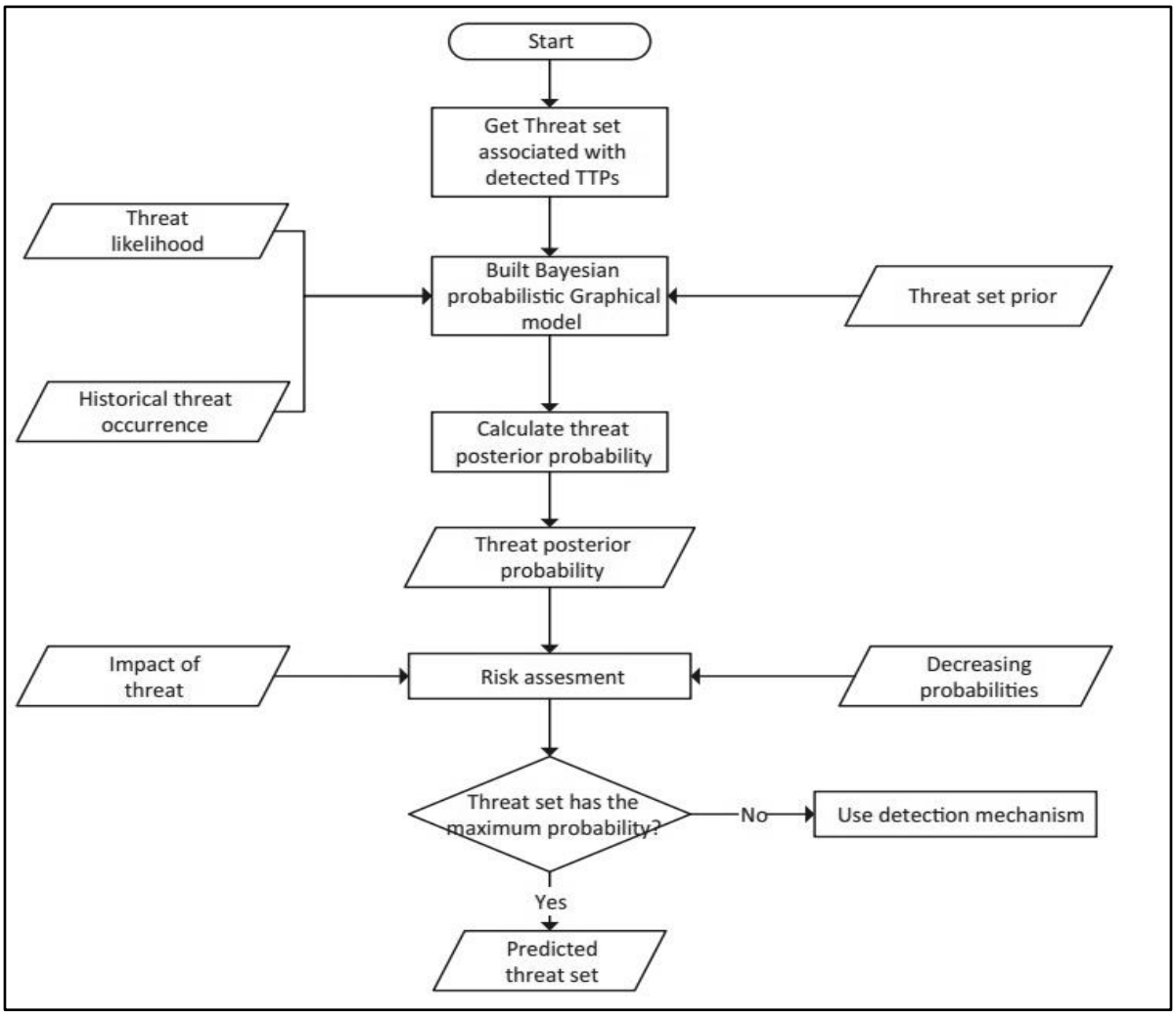

Рисунок 10 – Блок-схема предлагаемой модифицированной байесовской вероятности модели прогнозирования угроз

1.2. Задача бинарной классификации вредоносных команд по метрике MITRE

Задача классификации заключается в предсказании категории или метки класса для новых наблюдений на основе обучающих данных. Другими словами, задача классификации состоит в том, чтобы установить соответсвие между входными данными и фиксированным набором категорий, называемыми классами. Это тип задачи, который пытается ответить на вопрос «в каком классе находится данное наблюдение?».

Для решения задач классификации используются различные алгоритмы и модели машинного обучения, такие как метод опорных векторов, бустинг, K-NN, нейронные сети, байесовский классификатор, логистическая регрессия, бэггинг и дерево решений [8].

MITRE ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) – это модель, разработанная MITRE Corporation, которая описывает стандартные тактики, техники и процедуры, используемые киберпреступниками и злонамеренными акторами вовремя кибератак. База была создана в 2013 году с целью составления структурированной матрицы используемых киберпреступниками приёмов для упрощения задачи реагирования на киберинциденты. Модель MITRE ATT&CK содержит матрицы, которые описывают различные типы атак и фазы кибератаки, а также конкретные методы и техники, которые злоумышленники могут использовать.

Каждая тактика и техника имеет уникальный идентификатор и описание, что позволяет исследователям и организациям понять, какие уязвимости и слабости могут быть использованы в их сетях для обнаружения и предотвращения атак. Пример матрицы ATT&CK представлен на рисунке 11.

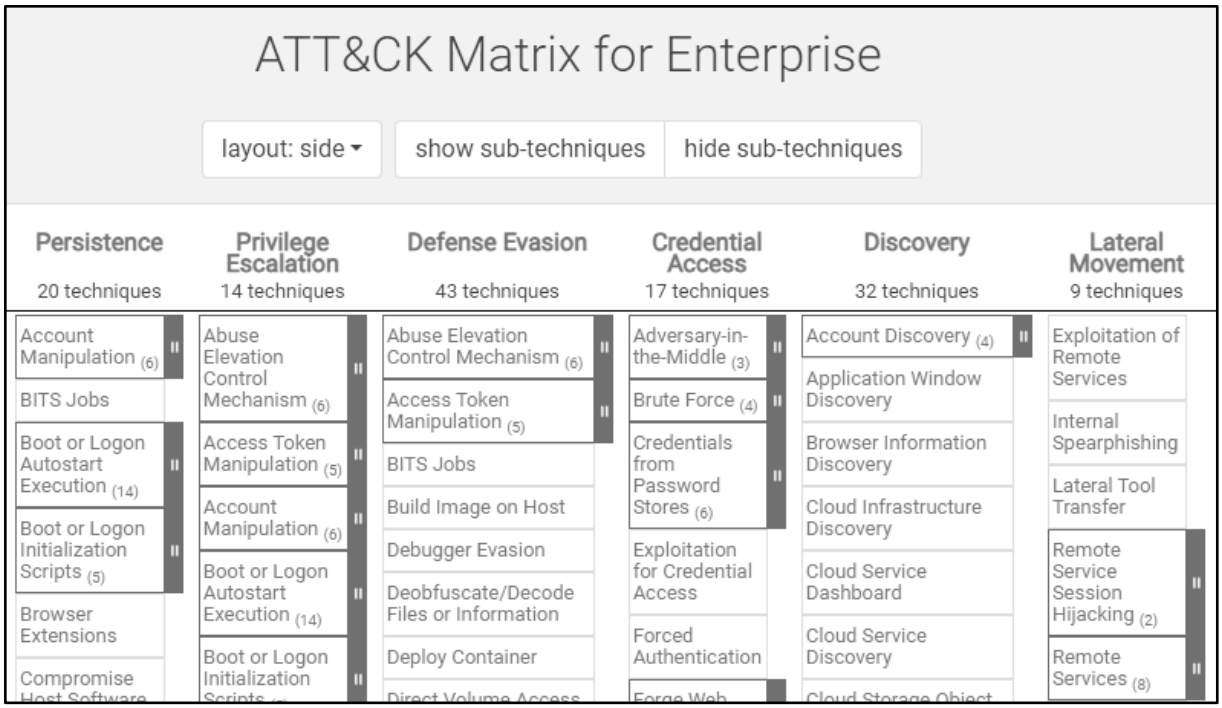

Рисунок 11 – Матрица ATT&CK

В качестве примера вредоносной команды возьмем команду «route print». Команда «route» – основная, а «print» – вспомогательная. Поэтому в матрице ATT&CK ищем информацию про команду «route» (рисунок 12).

Domain	ID	Name
Enterprise	T1016	System Network Configuration Discovery

Рисунок 12 – Команда «route» в матрице ATT&CK

В матрице «ATT&CK» написано, что данная команда может быть использована для получения конфигурации системной сети, следовательно, команда «route print» является вредоносной.

Пример еще одной вредоносной команды – «netsh wlan show interface». Основную нагрузку несет в себе команда «netsh», информация о которой представлена на рисунке 13.

Domain	ID		Name
Enterprise	T1546	.007	Event Triggered Execution: Netsh Helper DLL
Enterprise	T1562	.004	Impair Defenses: Disable or Modify System Firewall
Enterprise	T1090		Proxy
Enterprise	T1518	.001	Software Discovery: Security Software Discovery

Рисунок 13 – Команда «netsh» в матрице MITRE

Как видно из матрицы, команда «netsh» может быть использована в четырех разных техниках, например, чтобы отключить локальный файервол или для установки прокси-туннеля для удаленного доступа. В примере команда «netsh wlan show interface» покажет конфигурацию интерфейса «wlan», поэтому она считается вредоносной.

В качестве третьей вредоносной команды будет «tasklist», информация о которой представлена в матрице MITRE (рисунок 14).

Domain	ID		Name
Enterprise	T1057		Process Discovery
Enterprise	T1518	.001	Software Discovery: Security Software Discovery
Enterprise	T1007		System Service Discovery

Рисунок 14 – Команда «tasklist» в матрице MITRE

Из матрицы узнаем, что команда может быть использована с целью получения списка запущенных процессов и сервисов в системе, поэтому команда «tasklist» является вредоносной.

В качестве безопасной команды будет команда «git.exe branch –D master». При обращении к матрице MITRE с целью найти информацию про эту команду результат поиска будет «No results» (рисунок 15), поэтому команда «git.exe branch –D master» не является вредоносной.

Рисунок 15 – Команда «git.exe» в матрице MITRE

1.3. Методы машинного обучения

1.3.1. Naïve Bayes

Наивный байесовский классификатор – классификатор, основанный на теореме Томаса Байеса. Является главным методом для понимания вероятности некоторого события $P(A|B)$ при наличии некой новой информации, $P(B|A)$ и априорной субъективной оценки вероятности события $P(A)$ (формула 1):

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}. \quad (1)$$

Наивные байесовы классификаторы объединяют в общий классификатор ряд желательных в практическом машинном самообучении качеств. К ним относятся:

1) интуитивно понятный подход;
2) возможность работы с малыми данными;
3) низкие затраты на тренировку и предсказание;
4) часто надежные результаты в разнообразных условиях.

В частности, наивный байесов классификатор основан на формуле (2):

$$P\left(y\middle|x_1, \dots, x_j\right) = \frac{P\left(x_1, \dots, x_j\middle|y\right) \cdot P(y)}{P\left(x_1, \dots, x_j\right)}, \quad (2)$$

где $P(y|x_1, \dots, x_j)$ называется апостериорным значением и является вероятностью того, что наблюдение принадлежит классу y при условии, что это наблюдение имеет значения j признаков x_1,..., x_j ;

$P(x_1, \dots, x_j|y)$ называется правдоподобием и является правдоподобной оценкой вероятности, что наблюдение имеет значения признаков x_1,...,x_j при условии, что дан их класс y;

$P(y)$ называется априорной вероятностью и является нашей субъективной оценкой вероятности класса y перед рассмотрением данных;

$P(x_1, \dots, x_j)$ называется предельной вероятностью [10].

1.3.2. Logistic Regression

Несмотря на наличие слова «регрессия» в названии, логистическая регрессия на самом деле является широко используемым бинарным классификатором. В логистической регрессии линейная модель включается в логистическую функцию $\frac{1}{1+e^{-z}}$ таким образом, как показано в формуле (3):

$$P(y_i = 1|X) = \frac{1}{1 + e^{-(\beta_0+\beta_1 x)}}, \quad (3)$$

где $P(y_i = 1|X)$ – вероятность, что целевое значение i-го наблюдения y_i, является классом 1; X – тренировочные данные; β_0 и β_1 – параметры, которые необходимо заучить; e – эйлерово число. Эффект логистической функции заключается в ограничении значения результата функции диапазоном между 0 и 1, чтобы его можно было интерпретировать как вероятность. Если $P(y_i = 1|X)$ больше 0,5, то предсказывается класс 1, в противном случае – класс 0.

Для улучшения обобщающей способности получающейся модели, то есть уменьшения эффекта переобучения, на практике часто рассматривается логистическая регрессия с регуляризацией.

Регуляризация заключается в том, что вектор параметров рассматривается как случайный вектор с некоторой заданной априорной плотностью

распределения. Для обучения модели вместо метода наибольшего правдоподобия при этом используется метод максимизации апостериорной оценки [10].

1.3.3. Decision Tree

Дерево принятия решений представляет собой алгоритм рекурсивного разделения.

Дерево принятия решений работает следующим образом: берется весь обучающий набор данных, называемый «корневым узлом», и разбивается на два или более узлов так, чтобы наблюдения, попавшие в разные узлы, максимально отличались друг от друга по зависимой переменной. В роли правил разбиения, максимизирующих эти различия, выступают значения независимых переменных.

Пример дерева решений представлен на рисунке 16.

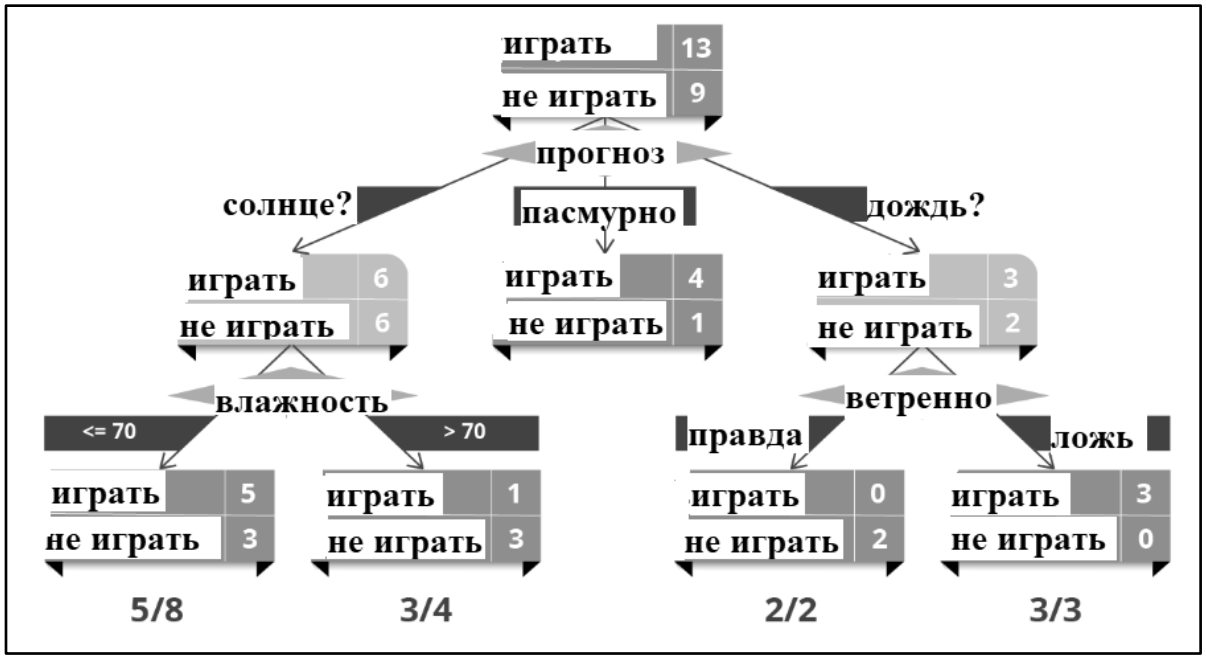

Рисунок 16 – Пример дерева решений

Качество разбиения оценивается с помощью статистических критериев. Правила и статистики отмечаются на ветвях – линиях, которые соединяют разбиваемый узел с узлами, полученными в результате разбиения. Для каждого узла вычисляются вероятности в виде процентных долей категорий

зависимой переменной (если зависимая переменная является категориальной) или средние значения зависимой переменной (если зависимая переменная является количественной). В результате выносится решение – спрогнозированная категория зависимой переменной (если зависимая переменная является категориальной) или спрогнозированное среднее значение зависимой переменной (если зависимая переменная является количественной).

Аналогичным образом каждый узел, получившийся в результате разбиения корневого узла, разбивается дальше на узлы, т. е. узлы внутри узла, и т. д. Этот процесс продолжается до тех пор, пока есть возможность разбиения на узлы. Получившаяся иерархическая структура, характеризующая взаимосвязь между значением зависимой переменной и значениями независимых переменных, называется деревом (рисунок 17) [10].

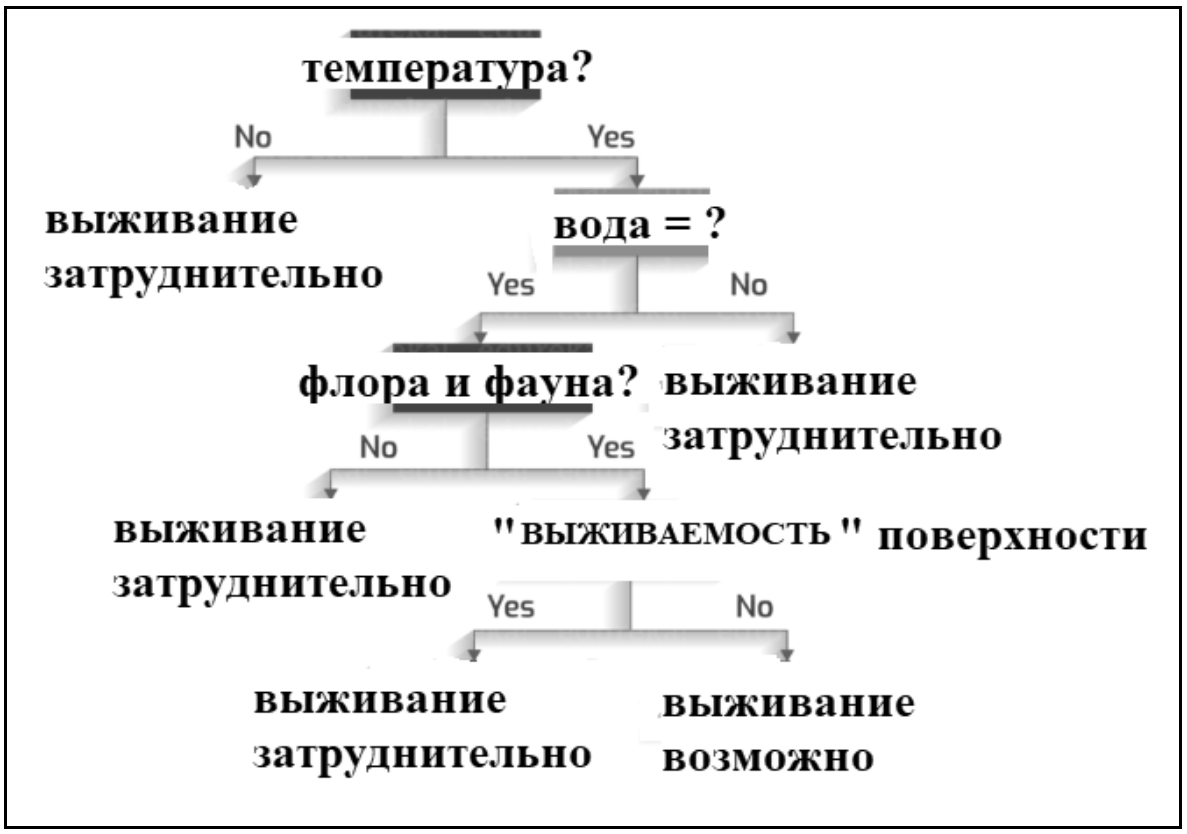

Рисунок 17 – Пример разбиения на узлы

1.3.4. Bagging

Суть бэггинга заключается в создании множества «копий» обучающих данных (каждая копия немного отличается от других) и последующем

применении слабого алгоритма к каждой копии с целью получить несколько слабых моделей, а затем объединить их. Широко используемым и эффективным алгоритмом машинного обучения, основанным на идее бэггинга, является случайный лес, который был использован в данной работе.

«Классический» алгоритм бэггинга работает следующим образом. Из имеющегося обучающего набора создается B случайных выборок S_b (для каждого $b = 1,\ldots,B$) и на основе каждой выборки S_b строится модель f_b дерева решений. Чтобы получить выборку S_b для некоторого b, производится выборка с заменой. То есть сначала создается пустая выборка, а затем из обучающего набора выбирается случайный образец, и его точная копия помещается в S_b, при этом сам образец остается в исходном обучающем наборе. Выбор данных продолжается, пока не выполнится условие равенства $|Sb| = N$.

В результате обучения получается B деревьев решений.

Случайный лес имеет только одно отличие от классического бэггинга. Он использует модифицированный алгоритм обучения дерева, который при каждом расщеплении в процессе обучения проверяет случайное подмножество признаков. Это делается с целью устранить корреляцию между деревьями: если один или несколько признаков имеют большую прогнозирующую способность, многие деревья будут выбирать их для расщепления данных. Это приведет к появлению в «лесу» большого числа коррелированных деревьев. Корреляция по признакам с большой прогнозирующей способностью препятствует повышению точности предсказания [10].

1.3.5. Boosting

Метод бустинга заключается в использовании исходных обучающих данных и итеративного создания нескольких моделей с применением слабого алгоритма.

Каждая новая модель отличается от предыдущих тем, что, конструируя ее, слабый алгоритм пытается «исправить» ошибки, допускаемые

предыдущими моделями. Окончательная ансамблевая модель представляет собой комбинацию этих многочисленных слабых моделей, построенных итеративно.

Пример работы бустинга показан на рисунке 18.

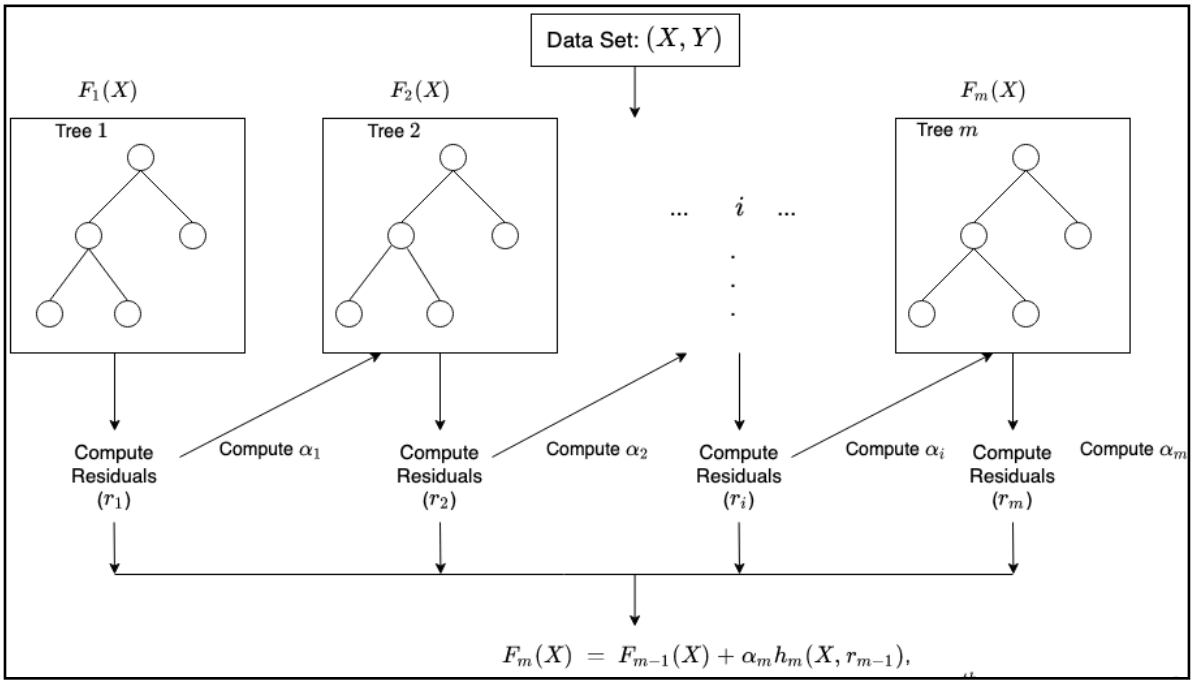

Рисунок 18 – Пример работы бустинга [11]

Процесс работы бустинга можно разбить на следующие шаги.

1. Задаются исходные данные и целевая переменная.

2. Создаются и обучаются множество слабых моделей. Примеры таких моделей могут включать деревья решений, линейные модели или нейронные сети. Каждая модель предсказывает целевую переменную на основе исходных данных.

3. Оцениваются ошибки предсказания каждой модели. Это может быть сделано с помощью различных метрик, таких как средняя квадратичная ошибка или кросс-энтропия.

4. Создается новая модель, где каждая следующая модель фокусируется на исправлении ошибок, сделанных предыдущими моделями. Для этого модель обучается на основе взвешенных данных, где веса присваиваются объектам, на которых предыдущие модели сделали больше ошибок.

5. Процесс создания моделей и их взвешивания повторяется множество раз, пока не будет достигнуто предоставленное количество моделей или не будет достигнута нужная точность [10].

1.3.6. Support Vector Machines

Метод опорных векторов (Support Vector Machines, SVM) является одним из классических алгоритмов машинного обучения, используемым для задач классификации и регрессии. Он основан на концепции разделения классов гиперплоскостью в пространстве признаков.

Принцип работы метода опорных векторов следующий.

Изначально, каждый объект обучающей выборки представляется в виде вектора признаков, где каждый признак описывает определенные характеристики объекта.

Находим гиперплоскость, которая лучше всего разделяет два класса объектов. Гиперплоскость – это (D-1)-мерное подпространство, в D-мерном пространстве признаков, где D – это количество признаков.

Целью SVM является нахождение оптимальной гиперплоскости, которая максимально разделяет классы, назначая объекты, которые находятся ближе к этой гиперплоскости, в качестве опорных векторов.

Опорные векторы — это объекты каждого класса, которые находятся на границе разделения и оказываются самыми близкими к оптимальной гиперплоскости.

SVM строит такую гиперплоскость, чтобы расстояние (зазор) между опорными векторами двух классов было максимальным. Это делается с помощью оптимизационной задачи, которая минимизирует функцию зазора и одновременно учитывает отступы (расстояния от объектов до гиперплоскости).

После того, как оптимальная гиперплоскость найдена, классификация новых объектов осуществляется путем определения, по какой стороне гиперплоскости они находятся.

Преимущества метода опорных векторов включают хорошую обобщающую способность, эффективность в пространствах большой размерности, способность обрабатывать нелинейные зависимости путем использования ядерных функций, а также устойчивость к выбросам в данных.

Однако, SVM также имеет некоторые ограничения, такие как чувствительность к выбору параметров и неэффективность при работе с большими наборами данных.

Тем не менее, метод опорных векторов остается одним из наиболее широко используемых и успешных алгоритмов машинного обучения для задач классификации и регрессии. Пример модели SVM показан на рисунке 19 [10].

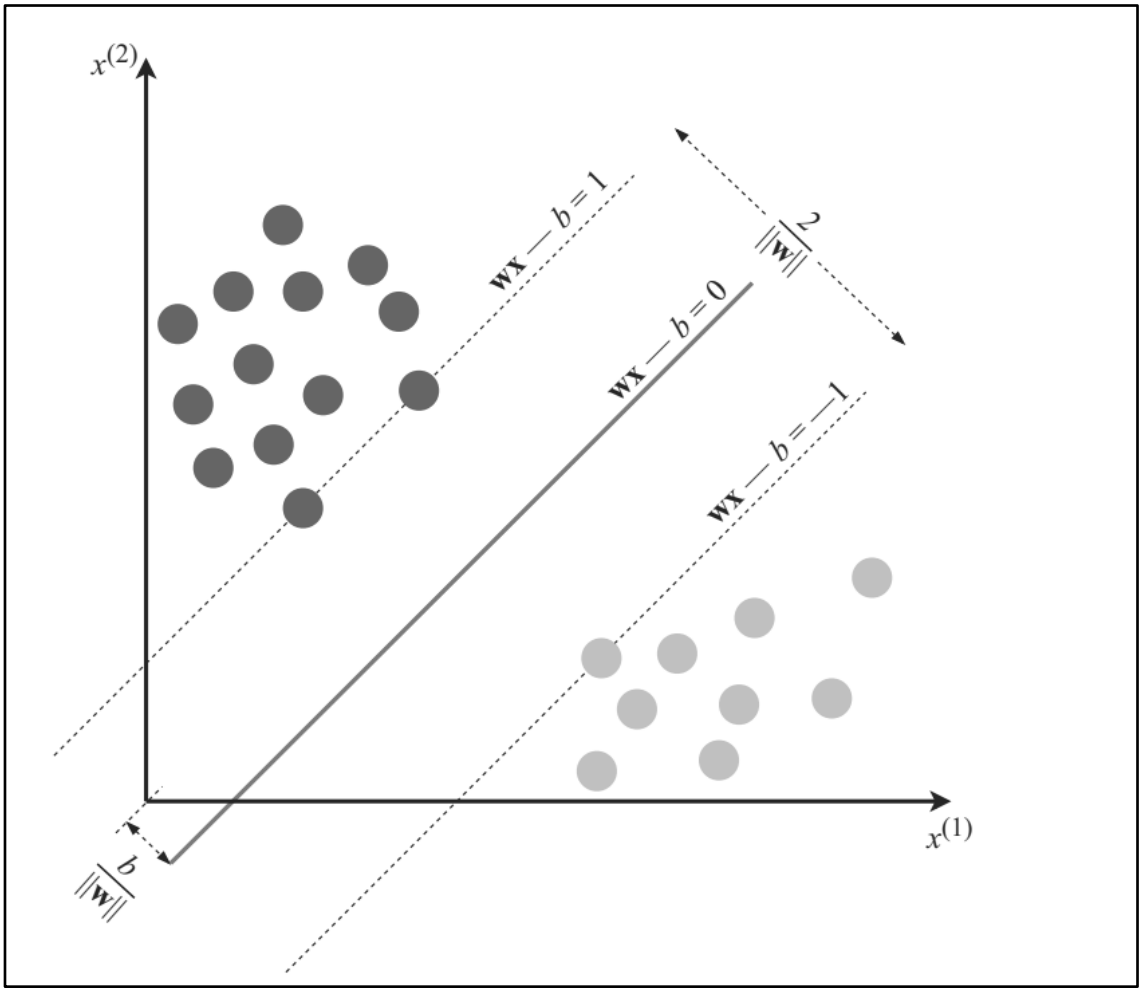

Рисунок 19 – Пример SVM модели [10]

1.3.7. K-Nearest Neighbours

K-Nearest Neighbors (K-NN) – это один из простых и широко используемых алгоритмов машинного обучения. Он относится к классу алгоритмов обучения с учителем, используемых для классификации и регрессии.

K-NN основывается на гипотезе компактности, согласно которой близким объектам соответствуют сходные значения целевой переменной. Алгоритм определяет классификацию нового примера, основываясь на k ближайших соседях, где k – это гиперпараметр, определяющий количество ближайших соседей, которые будут использоваться для классификации.

Работа алгоритма K-NN выглядит следующим образом:

1) выбирается значение k;

2) вычисляется расстояние (чаще всего используется евклидово расстояние) между новым примером и всеми обучающими объектами;

3) выбираются k объектов с наименьшим расстоянием до нового примера;

4) определяется метка класса для нового примера на основе меток классов выбранных ближайших соседей.

Например, для классификации с помощью голосования большинства, метка класса нового примера будет определяться классом, который чаще всего встречается среди k ближайших соседей.

Алгоритм K-NN не требует предварительного обучения, так как все вычисления проводятся на основе расстояний между объектами в обучающем наборе данных и новым примером. Также, K-NN является ленивым алгоритмом, так как не строит модель и не вычисляет параметров.

Однако, при использовании алгоритма K-NN необходимо учитывать, что выбор оптимального значения k, а также метрики расстояния между объектами, может оказывать существенное влияние на результаты классификации. Кроме того, K-NN применим только для числовых признаков и требует некоторой предварительной обработки данных для нормализации

значений и обработки категориальных признаков. Пример работы KNN модели показан на рисунке 20 [10].

Рисунок 20 – Работа KNN модели [12]

1.4. Набор данных

Набор данных «POWERSHELL.csv» содержит информацию о PowerShell командах. Данный набор был предоставлен компанией ООО «Р-Вижн». Этот набор данных был собран аналитиком по машинному обучению и безопасности. Предоставленный набор состоит их 1 743 записей, где каждая запись – это PowerShell команда. Пример команд приведен в таблице 5.

Предоставленный набор данных был разделен на обучающий и тестовый. В тестовый набор данных вошло 500 команд, а в обучающий – 1 243.

Из-за малого количества команд валидационный набор данных создать не получилось, так как, если взять команды из обучающей выборки, то есть вероятность, что модель обучится плохо и на тестовой выборке покажет плохие результаты, а если взять команды из тестовой выборки, то результаты тестирования модели могут быть неточными.

Таблица 5 – Примеры команд

Команда	Что делает?
Clear content	Команда «Clear content» в PowerShell используется для удаления содержимого файлов. Когда эту команду применяют к файлу, она приводит к удалению всех данных внутри этого файла, не затрагивая сам файл.
Clear History	Команда «Clear-History» в PowerShell используется для удаления истории командной строки, которая сохраняется в течение текущей сессии.
$computer = "<hostname>"	Команда «$computer = "<hostname>"» в PowerShell используется для создания переменной "$computer" и присваивания ей значения "<hostname>".
Get-LocalGroup | ft Name	Команда «Get-LocalGroup | ft Name» в PowerShell используется для получения списка локальных групп на компьютере и отображения только их имен в виде таблицы. «Get-LocalGroup» - это команда, которая извлекает информацию о локальных группах на компьютере. Локальные группы представляют собой группы пользователей и компьютеров, которые могут быть установлены и управляться на самом компьютере без использования домена. «| ft Name» – это команда «Format-Table», которая форматирует вывод команды «Get-LocalGroup» и отображает только столбец «Name» (имя группы) в виде таблицы.
$assembly = [Ref].Assembly.GetType(('{0}{1}i{2}' -f $a,$b,$u))	Команда «$assembly = [Ref].Assembly.GetType(('{0}{1}i{2}' -f $a,$b,$u))» в PowerShell создает переменную «$assembly» и присваивает ей тип сборки, определенной с помощью выражения '{0}{1}i{2}' -f $a,$b,$u.
$pass = ConvertTo-SecureString '<PASSWORD>' -AsPlainText -Force	Команда «$pass = ConvertTo-SecureString '<PASSWORD>' -AsPlainText -Force» конвертирует строку «PASSWORD» в защищенный формат данных. Используется для создания защищенных паролей.
netstat -nao	Команда «netstat -nao» используется для отображения списка всех подключений и портов. Параметр «-n» устанавливает отображение портов и адресов в числовом формате, «-a» показывает все подключения, а «-o» выводит PID процесса, который использует данное подключение.
Get-Process	Команда «Get-Process» получает процессы, выполняющиеся на локальном или удалённом компьютере. Она позволяет просматривать информацию о процессах, таких как их имена, идентификаторы, владелец, состояние и другие свойства.

2. ПРОЕКТИРОВАНИЕ

2.1. Модели методов машинного обучения

Модель машинного обучения – это метод создания алгоритма, опирающийся на набор данных о каком-либо явлении. За основу алгоритма берут данные об объекте: они поступают в модель, а выходят из нее уже в виде прогноза – готового решения.

Существует три вида машинного обучения, и у каждого свои параметры, по которым она обрабатывает информацию.

1. С учителем. В обучении с учителем набор данных организован как коллекция размеченных образцов $\{(x_i, y_i)\}_{i=1}^{N}$, где N – количество наблюдений, а x_i – вектор признаков. y_i – элемент конечного множества классов $\{1, 2, \dots, C\}$. Может быть представлена в виде вещественного числа, вектора, матрицы, дерево или граф. Эта метка показывает, к какой категории принадлежит образец. При использовании обучения с учителем в конечном результате должна получиться модель, которая получает на вход вектор признаков «x» и на выходе возвращает метку класса для этого набора признаков.

2. Без учителя. При использовании этого вида обучения набор данных не размечается заранее и является коллекцией образцов $\{x_i\}_{i=1}^{N}$, где x – вектор признаков. При обучении модели без учителя должна получиться модель, которая принимает вектор признаков на вход, а на выходе возвращает его преобразованную версию либо в другой вектор, либо в значение.

3. С подкреплением. это раздел машинного обучения, изучающий методы, с помощью которых компьютеры могут научиться действовать в среде для достижения определенной цели. Это достигается путем предоставления компьютеру положительного или отрицательного подкрепления за его действия.

2.2. Разметка набора данных

Для разметки команд была использована матрица MITRE ATT&CK в которой команды разделены по смыслу своего действия. Например, команда

«whoami» может быть использована для получения информации о пользователе, поэтому она считается вредоносной. В случае, если команда не была найдена в матрице – она считается невредоносной.

Разметка набора данных проводилась вручную. Все команды были разделены на два класса – вредоносные и безопасные (в столбце «malicious» 1 и 0 соответственно). Общее количество данных было равно 1 742 строк из которых 920 – вредоносные и 822 безопасные. Пример размеченного набора данных приведен на рисунке 21.

get-netadapater	0
get-process	0
get-service	0
install-module NTFSSecurity	0
invoke-command -ComputerName DC-Name -scriptblock {wbadmin start sy	0
set-executionpolicy unrestricted	0
systeminfo | more	0
Get-Help process	1
Get-Help Get-Item -Full	1
Get-Help Get-Item -Examples	1
Import-Module <modulepath>	1
Get-Command -Module <modulename>	1
powershell "IEX(New-Object Net.WebClient).downloadString('http://10.10.1	1
echo IEX(New-Object Net.WebClient).DownloadString('http://10.10.14.13:8	1
iex (iwr '10.10.14.9:8000/ipw.ps1')	1

Рисунок 21 – Пример разметки набора данных

2.3. Требования к приложению

2.3.1. Функциональные требования

Функциональные требования определяют, каким должно быть поведение продукта в тех или иных условиях. Они определяют, что разработчики должны создать, чтобы пользователи смогли выполнить свои задачи [13].

Разрабатываемое приложение должно удовлетворять следующим требованиям:

1) пользователь должен иметь возможность выбора между классификацией команд уже готовыми моделями машинного обучения и обучением собственной модели;

2) пользователь должен иметь возможность выбора файла в формате «.csv»;

3) пользователь должен иметь возможность выбрать обученную моделей для анализа команд;

4) пользователь должен иметь возможность настраивать гиперпараметры модели при обучении;

5) пользователь должен иметь возможность скачать обученную им модель;

6) приложение должно классифицировать входные данные на два класса;

7) приложение должно выводить информацию о результате классификации;

8) приложение должно иметь возможность сохранить результаты классификации в файл.

2.3.2. Нефункциональные требования

Нефункциональные требования – это требования к системе или продукту, которые не относятся к его функциональности, то есть не к тому, что система должна делать, а к тому, как это должно быть сделано, и каким должен быть результат [13].

Разрабатываемое приложение должно удовлетворять следующим нефункциональным требованиям:

1) приложение должно быть разработано на языке Python;

2) приложение должно быть реализовано в виде веб-приложения.

2.4. Диаграмма вариантов использования системы

На рисунке 22 представлена диаграмма вариантов использования.

Актер, взаимодействующий с системой – пользователь. Он имеет четыре варианта использования системы: обучить модель, классифицировать команды с помощью готовой модели, прочитать теорию, узнать про набор данных.

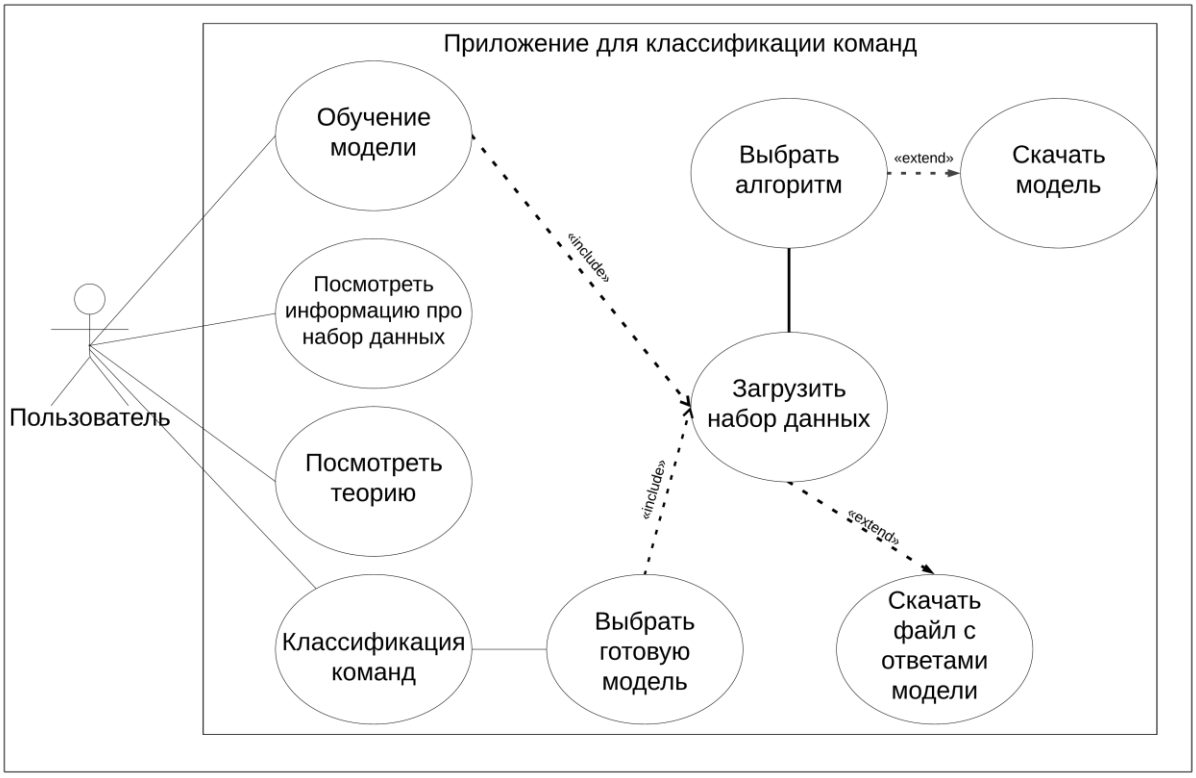

Рисунок 22 – Диаграмма вариантов использования

Находясь на главной станице, пользователь может перейти на страницу обучения моделей машинного обучения, посмотреть информацию про набор данных, изучить теорию про используемые алгоритмы машинного обучения и перейти на страницу для классификации команд. Если пользователь находиться на странице для классификации команд или обучения моделей, пользователю необходимо загрузить набор данных, после чего выбрать алгоритм для модели, если пользователь был на странице с обучением или дождаться результатов классификации, если он был на странице с классификацией.

2.5. Диаграмма компонентов системы

На рисунке 23 изображена диаграмма компонентов для приложения. Она показывает, какие компоненты используются в системе и как они взаимодействуют друг с другом.

Приложения состоит из следующих компонентов:

1) браузер пользователя – веб-браузер пользователя, через который он взаимодействует с системой;

2) приложение streamlit – приложение, написанное на python с использованием streamlit API;

3) приложение в контейнере Docker – запущенный контейнер docker, который содержит приложение внутри.

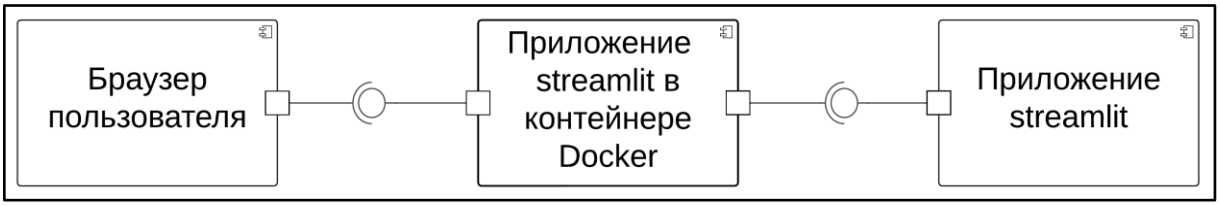

Рисунок 23 – Диаграмма компонентов

Приложение работает следующим образом.

1. Создается образ приложения с помощью docker.
2. Запускается контейнер с streamlit приложением.
3. Браузер пользователя используется для отображения интерфейса приложения.

3. РЕАЛИЗАЦИЯ

3.1. Программные средства разработки

Для разработки программы был выбран высокоуровневый язык программирования Python 3.10.6.

Разработка велась с использованием интерактивной вычислительной платформы с веб-интерфейсом Jupyter Notebook. Jupyter Notebook позволяет отдельные строки кода в отдельных ячейках, что упрощает работу в области машинного обучения [14].

Для реализации методов машинного обучения с учителем была использована библиотека scikit-learn 1.2.2. В данной библиотеке имеются готовые алгоритмы машинного обучения и вспомогательные инструменты для работы с данными и моделями машинного обучения [5].

Для работы с «.csv» файлами была использована библиотека pandas, которая позволяет считывать, сохранять, изменять файлы [6].

Streamlit – библиотека на языке Python, которая помогает разработчикам создавать интерактивные веб-приложения для визуализации данных и демонстрации моделей. Она имеет простой интерфейс, который позволяет быстро создать пользовательский интерфейс и визуализацию данных. Streamlit также легко работает с другими популярными библиотеками для анализа данных и машинного обучения, такими как pandas и Matplotlib [8].

Docker – это платформа для упаковки и запуска приложений в изолированных контейнерах.

Он упрощает разработку, развертывание и масштабирование приложений, обеспечивая единообразную среду выполнения.

Docker контейнеры являются портативными и безопасными, позволяя легко управлять приложениями в различных средах [9].

3.2. Реализация и обучение алгоритмов машинного обучения

3.2.1. Реализация модели Naïve Bayes

Модель была реализована с помощью класса GaussianNB [17] из библиотеки scikit-learn.

Входные параметры для инициализации модели используются следующие:

1) `priors` – предыдущие вероятности классов;

2) `var_smothing` – часть наибольшей дисперсии из всех характеристик, которая добавляется к дисперсиям для стабильности расчета.

Поскольку данная модель имеет всего два параметра было принято решение провести исследование с помощью GridSearchCV [19] только для векторайзера.

Параметры, описанные в документации [17], оставлены без изменений и принимают стандартные значения.

В листинге 1 представлен код реализации модели, обученной с помощью Naïve Bayes.

Листинг 1 – Код модели Naïve Bayes

```
vectorizer = CountVectorizer(analyzer='char', ngram_range=(4,
4)).fit(using_data['command_clear'])
df = vectorizer.transform(list(using_data['command_clear'])).toarray()
X_train, X_test, Y_train, Y_test = train_test_split(df,
using_data['malicious'], test_size=0.2)
clf = GaussianNB().fit(X_train, Y_train)
clf_pred = clf.predict(X_test)
confusion_matrix(Y_test, clf_pred)
filename = 'bayes_model.sav'
pickle.dump(clf, open(filename, 'wb'))
```

3.2.2. Реализация модели Decision Tree

Модель была реализована с помощью класса Decision Tree Classifier [18] из библиотеки scikit-learn. Входные параметры для инициализации модели используются следующие:

1) `criterion` – функция измерения качества разделения;

2) `splitter` – стратегия, используемая для выбора разделения в каждом узле;

3) `max_depth` – масимальная глубина дерева;

4) `min_samples_split` – минимальное количество выборок, необходимое для разделения внутреннего узла;

5) `min_samples_leaf` – минимальное количество выборок, необходимое для конечного узла;

6) `max_features` – ряд особенностей, которые нужно учитывать для каждого разбиения.

Для подбора значений было проведено исследование модели с помощью GridSearchCV [19] из библиотеки scikit-learn. Результат исследования представлен на рисунке 24.

```
CountVectorizer()
Best parameter {'classifier__criterion': 'gini', 'classifier__max_depth': 20, 'classifier__max_features': 'sqrt', 'c
lassifier__min_samples_leaf': 1, 'classifier__min_samples_split': 2, 'classifier__splitter': 'best', 'vectorizer__an
alyzer': 'char_wb', 'vectorizer__ngram_range': (2, 2)} (CV score=0.933)

TfidfVectorizer()
Best parameter {'classifier__criterion': 'gini', 'classifier__max_depth': 15, 'classifier__max_features': 'sqrt', 'c
lassifier__min_samples_leaf': 1, 'classifier__min_samples_split': 3, 'classifier__splitter': 'best', 'vectorizer__an
alyzer': 'char_wb', 'vectorizer__ngram_range': (2, 2)} (CV score=0.935)
```

Рисунок 24 – Поиск значений параметров для модели Decision Tree

В результате исследования были выбраны следующие значения параметров:

1) `criterion` – «gini»;

2) `splitter` – «best»;

3) `max_depth` – «15»;

4) `min_samples_split` – «3»;

5) `min_samples_leaf` – «1»;

6) `max_features` – «sqrt».

Оставшиеся параметры, описанные в документации [18], оставлены без изменений и принимают стандартные значения.

В листинге 2 представлен код реализации модели, обученной с помощью алгоритма Decision Tree Classifier.

Листинг 2 – Модель Decision Tree Classifier

```
vectorizer = TfidfVectorizer(analyzer='char_wb', ngram_range=(2,
2)).fit(using_data['command_clear'])
df = vectorizer.transform(list(using_data['command_clear'])).toarray()
X_train, X_test, Y_train, Y_test = train_test_split(df,
using_data['malicious'], test_size=0.2)
clf = DecisionTreeClassifier(criterion='gini', splitter='best',
max_depth=15, min_samples_leaf=1, min_samples_split=3, max_features='sqrt',
class_weight='balanced').fit(X_train, Y_train)
clf_pred = clf.predict(X_test)
confusion_matrix(Y_test, clf_pred)
filename = 'des_tree_model.sav'
pickle.dump(clf, open(filename, 'wb'))
```

3.2.3. Реализация модели Bagging

Модель была реализована с помощью класса Random Forest Classifier [20] из библиотеки scikit-learn. Входные параметры для инициализации модели используются следующие:

1) `n_estimators` – количество деревьев в лесу;

2) `bootstrap` – используются ли данные предыдущих деревьев для построения следующих;

3) `max_depth` – максимальная глубина дерева;

4) `max_features` – ряд особенностей, которые нужно учитывать для каждого разбиения;

5) `min_samples_leaf` – минимальное количество выборок, необходимое для конечного узла;

6) `min_samples_split` – минимальное количество выборок, необходимое для разделения внутреннего узла.

Для подбора значений было проведено исследование модели с помощью GridSearchCV [19] из библиотеки scikit-learn. Результат исследования представлен на рисунке 25.

```
CountVectorizer()
Best parameter {'classifier__bootstrap': False, 'classifier__max_depth': 30, 'classifier__max_features': 'sqrt', 'cl
assifier__min_samples_leaf': 1, 'classifier__min_samples_split': 2, 'classifier__n_estimators': 200, 'vectorizer__an
alyzer': 'char', 'vectorizer__ngram_range': (2, 2)} (CV score=0.971)
```

Рисунок 25 – Поиск значений параметров для модели Bagging

В результате исследования были выбраны следующие значения параметров:

1) `n_estimators` – «200»;
2) `bootstrap` – «False»;
3) `max_depth` – «30»;
4) `max_features` – «sqrt»;
5) `min_samples_leaf` – «1»;
6) `min_samples_split` – «2».

Оставшиеся параметры, описанные в документации [20], оставлены без изменений и принимают стандартные значения.

В листинге 3 представлен код реализации модели, обученной с помощью алгоритма Random Forest Classifier.

Листинг 3 – Модель Random Forest Classifier

```
vectorizer = CountVectorizer(analyzer='char', ngram_range=(2, 2)).fit(us-
ing_data['command_clear'])
df = vectorizer.transform(list(using_data['command_clear'])).toarray()
X_train, X_test, Y_train, Y_test = train_test_split(df,
                                                    using_data['mali-
cious'],
                                                    test_size=0.2)
clf = RandomForestClassifier(n_estimators=200, max_depth=30,
                             min_samples_leaf=1, min_samples_split=2,
                             max_features='sqrt', boot-
strap=False).fit(X_train, Y_train)
clf_pred = clf.predict(X_test)
confusion_matrix(Y_test, clf_pred)
filename = 'rand_for_model.sav'
pickle.dump(clf, open(filename, 'wb'))
```

3.2.4. Реализация модели Logistic Regression

Модель была реализована с помощью класса Logistic Regression [21] из библиотеки scikit-learn. Входные параметры для инициализации модели используются следующие:

1) `C` – обратная сила регуляризации;
2) `class_weight` – корректирует вес класса модели для учета дисбаланса классов в обучающем наборе данных;
3) `fit_intercept` – указывает, следует ли добавить константу (также известную как смещение) в функцию принятия решения;

4) `intercept_scaling` – к экземпляру класса добавляется «синтетический» признак с постоянным значением, равным intercept_scaling;

5) `penalty` – указывает норму штрафа модели;

6) `solver` – алгоритм, используемый для оптимизации.

Для подбора значений было проведено исследование модели с помощью GridSearchCV [19] из библиотеки scikit-learn. Результат исследования представлен на рисунке 26.

```
CountVectorizer()
Best parameter {'classifier__C': 100.0, 'classifier__class_weight': 'balanced', 'classifier__fit_intercept': True, '
classifier__intercept_scaling': 0.01, 'classifier__penalty': 'l2', 'classifier__solver': 'lbfgs', 'scaler__analyzer
': 'word', 'scaler__ngram_range': (1, 1)} (CV score=0.970)
```

Рисунок 26 – Поиск значений параметров для модели Logistic Regression

В результате исследования были выбраны следующие значения параметров:

1) `C` – «100»;

2) `class_weight` – «balanced»;

3) `fit_intercept` – «True»;

4) `intercept_scaling` – «0.01»;

5) `penalty` – «l2»;

6) `solver` – «lbfgs».

Оставшиеся параметры, описанные в документации [21], оставлены без изменений и принимают стандартные значения.

В листинге 4 представлен код реализации модели, обученной с помощью алгоритма Logistic Regression.

Листинг 4 – Модель Logistic Regression

```
vectorizer = CountVectorizer(analyzer='word', ngram_range=(1, 1)).fit(us-
ing_data['command_clear'])
df = vectorizer.transform(list(using_data['command_clear'])).toarray()
X_train, X_test, Y_train, Y_test = train_test_split(df, using_data['mali-
cious'], test_size=0.2)
clf = LogisticRegression(max_iter=100000, C=100, class_weight='balanced',
fit_intercept=True, intercept_scaling=0.01, penalty='l2', solver='lbfgs',
).fit(X_train, Y_train)
clf_pred = clf.predict(X_test)
```

```
confusion_matrix(Y_test, clf_pred)
filename = 'log_res_model.sav'
pickle.dump(clf, open(filename, 'wb'))
```

3.2.5. Реализация модели Boosting

Модель была реализована с помощью класса XGBClassifier [24] из библиотеки xgboost. Входные параметры для инициализации модели используются следующие:

1) `colsample_bytree` – представляет собой отношение подвыборок столбцов при построении каждого дерева;

2) `gamma` – минимальное уменьшение потерь, необходимое для создания дополнительного разбиения на конечном узле дерева;

3) `learning_rate` – уменьшение размера шага, используемое при обновлении, предотвращает переобучение;

4) `max_depth` – максимальная глубина дерева;

5) `min_child_weight` – минимальная сумма веса экземпляра (гессиан), необходимая для узла;

6) `n_estimators` – количество базовых деревьев решений, которые будут использоваться в ансамбле градиентного бустинга;

7) `subsample` – соотношение подвыборок обучающих экземпляров.

Для подбора значений было проведено исследование модели с помощью GridSearchCV [19] из библиотеки scikit-learn. Результат исследования представлен на рисунке 27.

```
CountVectorizer()
Best parameter {'classifier__colsample_bytree': 0.8, 'classifier__gamma': 0.75, 'classifier__learning_rate': 0.2, 'c
lassifier__max_depth': 7, 'classifier__min_child_weight': 2, 'classifier__n_estimators': 200, 'classifier__subsample
': 1.0, 'scaler__analyzer': 'char', 'scaler__ngram_range': (2, 2)} (CV score=0.964)

TfidfVectorizer()
Best parameter {'classifier__colsample_bytree': 0.8, 'classifier__gamma': 0.75, 'classifier__learning_rate': 0.1, 'c
lassifier__max_depth': 7, 'classifier__min_child_weight': 2, 'classifier__n_estimators': 200, 'classifier__subsample
': 0.65, 'scaler__analyzer': 'char_wb', 'scaler__ngram_range': (2, 2)} (CV score=0.959)
```

Рисунок 27 – Поиск значений параметров для модели Boosting

В результате исследования были выбраны следующие значения параметров:

1) `colsample_bytree` – «0,8»;

2) `gamma` – «0,75»;

3) `learning_rate` – «0,2»;

4) `max_depth` – «7»;

5) `min_child_weight` – «2»;

6) `n_estimators` – «200»;

7) `subsample` – «1,0».

Оставшиеся параметры, описанные в документации [24], оставлены без изменений и принимают стандартные значения.

В листинге 5 представлен код реализации модели, обученной с помощью алгоритма XGBClassifier.

Листинг 5 – Модель XGBClassifier

```
vectorizer = CountVectorizer(analyzer='word',  ngram_range=(1,1)).fit(us-
ing_data['command_clear'])
X_train = vectorizer.transform(using_data['command_clear']).toarray()
Y_train = list(using_data['malicious'])
X_test = vectorizer.transform(data_test['command_clear']).toarray()
clf = xgb.XGBClassifier(colsample_bytree=0.8, gamma=0.75, learn-
ing_rate=0.2, max_depth=7, min_child_weight=2,
n_estimators=200, subsample=1).fit(X_train, Y_train)
```

3.2.6. Реализация модели Support Vector Machines

Модель была реализована с помощью класса SVC [25] из библиотеки scikit-learn. Входные параметры для инициализации модели используются следующие:

1) `gamma` – коэффициент ядра для «rbf», «poly» и «sigmoid»;

2) `kernel` – определяет тип ядра, который будет использоваться в алгоритме;

3) `C` – сила регуляризации обратно пропорциональна C.

Для подбора значений было проведено исследование модели с помощью GridSearchCV [19] из библиотеки scikit-learn. Результат исследования представлен на рисунке 28.

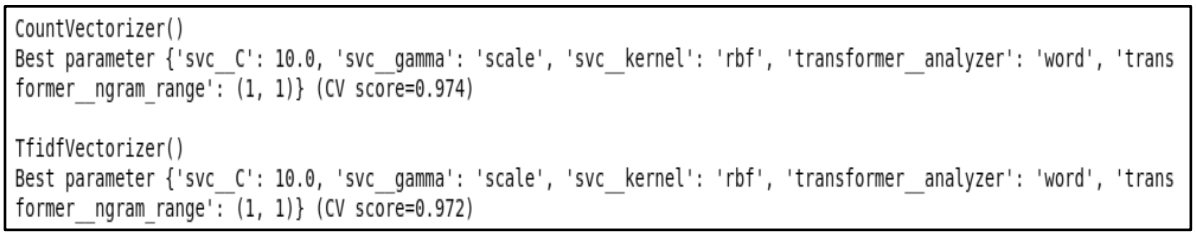

Рисунок 28 – Поиск значений параметров для модели Support Vector Machines

В результате исследования были выбраны следующие значения параметров:

1) `gamma` – «scale»;
2) `kernel` – «rbf»;
3) `C` –10,0.

Оставшиеся параметры, описанные в документации [25], оставлены без изменений и принимают стандартные значения.

В листинге 6 представлен код реализации модели, обученной с помощью алгоритма SVC.

Листинг 6 – Модель SVC

```
vectorizer = CountVectorizer(analyzer='word', ngram_range=(1, 1)).fit(us-
ing_data['command_clear'])
X_train = vectorizer.transform(using_data['command_clear']).toarray()
Y_train = list(using_data['malicious'])
X_test = vectorizer.transform(data_test['command_clear']).toarray()
clf = SVC(C=10, gamma='scale', kernel='rbf').fit(X_train, Y_train)
```

3.2.7. Реализация модели K-Nearest Neighbours

Модель была реализована с помощью класса KNeighborsClassifier [26] из библиотеки scikit-learn. Входные параметры для инициализации модели используются следующие:

1) `algorithm` – алгоритм, используемый при нахождении ближайших соседей;

2) `leaf_size` – размер листьев дерева, которые могут повлиять на скорость построения, а также на объем памяти, необходимый для хранения дерева (для некоторых алгоритмов);

3) `n_neighbors` – количество соседей, используемых по умолчанию;

4) `p` – параметр мощность для метрики Минковского;

5) `weights` – весовая функция, используемая для предсказания.

Для подбора значений было проведено исследование модели с помощью GridSearchCV [19] из библиотеки scikit-learn. Результат исследования представлен на рисунке 29.

```
CountVectorizer()
0.96125 {'classifier__algorithm': 'auto', 'classifier__leaf_size': 30, 'classifier__n_neighbors': 3, 'classifier__p'
: 3, 'classifier__weights': 'distance', 'pca__svd_solver': 'auto', 'transformer__analyzer': 'char_wb', 'transformer_
_ngram_range': (2, 2)}
TfidfVectorizer()
0.9681249999999999 {'classifier__algorithm': 'auto', 'classifier__leaf_size': 30, 'classifier__n_neighbors': 3, 'cla
ssifier__p': 3, 'classifier__weights': 'distance', 'pca__svd_solver': 'auto', 'transformer__analyzer': 'char_wb', 't
ransformer__ngram_range': (2, 2)}
```

Рисунок 29 – Поиск значений параметров для модели K-Nearest Neighbours

В результате исследования были выбраны следующие значения параметров:

1) `algorithm` – «auto»;

2) `leaf_size` – «30»;

3) `n_neighbors` – «3»;

4) `p` – «3»;

5) `weights` – «distance».

Оставшиеся параметры, описанные в документации [26], оставлены без изменений и принимают стандартные значения.

В листинге 7 представлен код реализации модели, обученной с помощью алгоритма SVC.

Листинг 7 – Модель KNN

```
vectorizer = CountVectorizer(analyzer='char_wb', ngram_range=(2,
2)).fit(using_data['command_clear'])
X_train = vectorizer.transform(using_data['command_clear']).toarray()
Y_train = list(using_data['malicious'])
X_test = vectorizer.transform(data_test['command_clear']).toarray()
clf = KNeighborsClassifier(algorithm='auto', leaf_size=30, n_neighbors=3,
p=3, weights='distance').fit(X_train, Y_train)
```

3.3. Реализация веб-клиента

Для реализации пользовательского интерфейса было принято решение написать веб-клиент. Веб-клиент должен предоставлять пользователю возможность взаимодействия с моделями машинного обучения такие как: выбор алгоритма модели машинного обучения, настройка гиперпараметров выбранной модели машинного обучения, переключение между возможностью обучить новую модель и классифицировать команды с помощью уже готовых.

Выбор был сделан в пользу фреймворка «streamlit» потому что он позволяет без знаний front-end разработки создавать веб-приложения для машинного обучения и Data Science. Также, streamlit совместим с такими библиотеками как «NumPy», «matplotlib», «scikit-learn», «pandas», которые также используются в работе. Главная страница веб-клиента показана на рисунке 30.

Рисунок 30 – Главная страница веб-клиента

Для начала использования фреймворка его необходимо установить. Для этого на главной странице ресурса [8] нажимаем «Try Streamlit now» и находим инструкцию по установке.

После установки фреймворка создаем главную страницу приложения, код которой показан в листинге 8.

Листинг 8 – Главная страница приложения

```
import streamlit as st
st.write("# Приложения для классификации команд")
st.markdown("""
Это приложение для обучения моделей машинного обучения для
классификации команд
### Как использовать приложение:
- Чтобы обучить модель - перейдите в раздел [Обуче-
ние](http://localhost:8501/Обучение)
- Чтобы использовать обученные модели - перейдите в раздел [Классифика-
ция](http://localhost:8501/Классификация)
- Чтобы узнать про используемый для обучения набор данных - перейдите в
раздел [Набор данных](http://localhost:8501/Датасет)
- Чтобы узнать про используемые алгоритмы - перейдите в раздел [Тео-
рия](http://localhost:8501/Теория)
""")
```

Далее создаем страницу для обучения новых моделей и прописываем логику работы приложения.

Сначала, пользователь должен загрузить набор данных, на которых будет обучать модель. После загрузки система предложит выбрать алгоритм, на котором будет обучаться модель, и его гиперпараметры. По окончании обучения пользователю будет предложено скачать получившуюся модель.

Внешний вид страницы показан на рисунке 31.

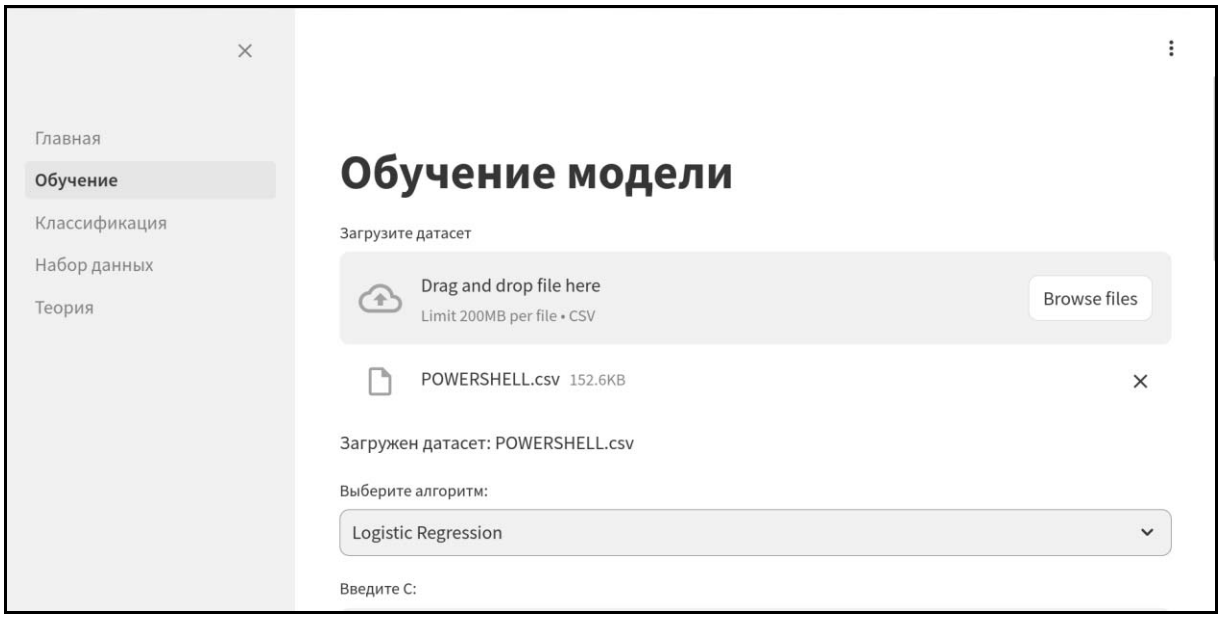

Рисунок 31 – Страница «Обучение модели»

Чтобы использовать обученные модели, необходимо создать страницу «Классификация». На этой странице готовые модели машинного обучения должны классифицировать команды и выводить результат классификации.

При открытии данный страницы система предложит загрузить готовую модель машинного обучения. Далее, пользователю будет предложено загрузить набор данных после загрузки которого автоматически начнется классификация команд. По окончании классификации система выведет ответ в виде таблицы, где правильно классифицированные команды будет выделены зеленым цветом, а неправильно – красным.

Внешний вид страницы представлен на рисунке 32.

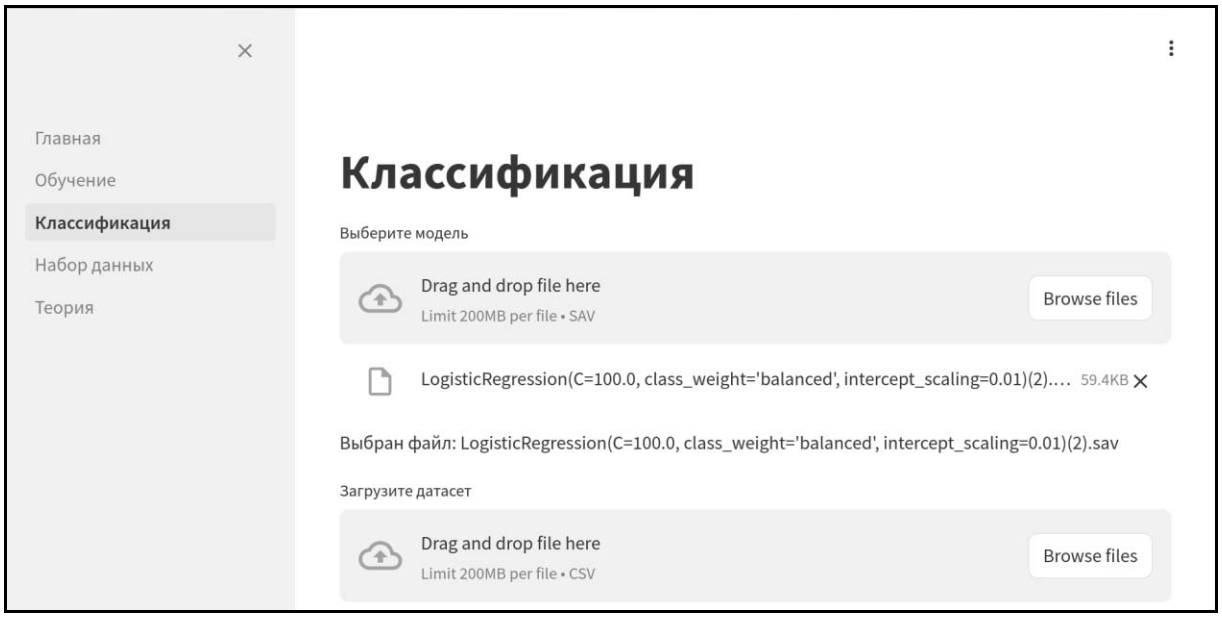

Рисунок 32 – Страница «Классификация»

Также были созданы две страницы, которые должны упростить взаимодействие пользователя с системой. Первая из них – «Набор данных». На данной странице описывается набор данных, который был использован для обучения готовых моделей, а также как он должен быть подготовлен для обучения и использования готовых моделей.

Последняя страница – «Теория». На ней можно прочитать теорию про алгоритмы машинного обучения, которые были использованы или могут

быть использованы при обучении моделей. Внешний вид показан на рисунке 33.

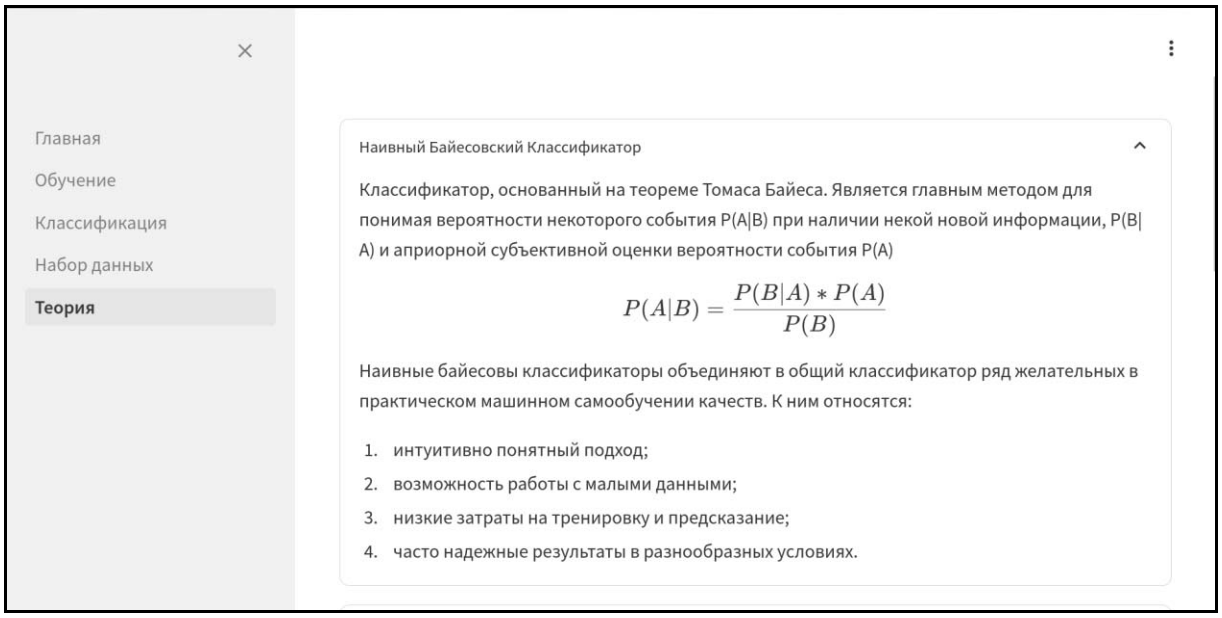

Рисунок 33 – Страница «Теория»

3.4. Docker контейнер

Docker контейнер – это стандартизированная единица программного обеспечения, которая включает в себя приложение и все необходимые для его работы зависимости (библиотеки, файлы конфигурации, исполняемые файлы). Контейнер обеспечивает изоляцию приложения от окружающей среды хоста, что гарантирует его независимость и стабильную работу в любой системе.

Docker контейнеры используют механизм виртуализации на уровне операционной системы, позволяющий запускать несколько изолированных контейнеров на одном хосте. Контейнеры создаются из образов, которые содержат все необходимые компоненты приложения и его конфигурации. Docker обладает простым интерфейсом командной строки для управления контейнерами: создание, запуск, остановка и удаление.

Использование Docker упрощает разработку, тестирование и развертывание приложений, так как обеспечивает единообразную среду выполнения на различных системах. Контейнеры позволяют избежать конфликтов

между версиями зависимостей и упрощают процесс масштабирования приложений. Docker также повышает безопасность, так как контейнеры изолированы друг от друга и от хостовой системы [9].

Чтобы использовать Docker его нужно установить. Для этого открываем официальную документацию [9] и следуем ей.

Чтобы убедится, что Docker установлен и работает в консоль вводим команду «sudo docker run hello-world», если в консоли появилось то, что показано на рисунке 34, значит Docker установлен правильно.

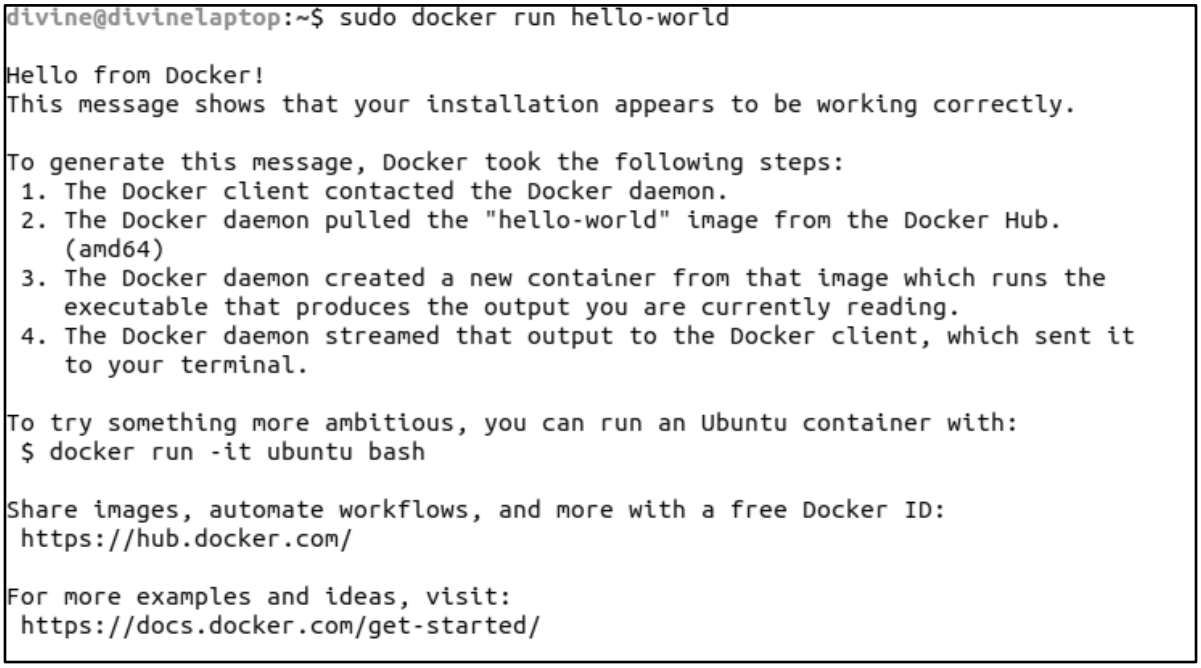

Рисунок 34 – Проверка установки Docker контейнера

Далее, нужно создать Dockerfile. Dockerfile – это текстовый документ, содержащий все команды, которые пользователь может вызвать в командной строке для сборки образа. Созданный Dockerfile показан на рисунке 35.

Первая команда – `FROM python:3.10` – определяет базовый образ, на основе которого будет создан новый образ Docker. В данном случае используется образ python версии 3.10.

`COPY requirements.txt` – копирует файл «requirements.txt» из текущего каталога сборки в рабочий каталог контейнера (корень образа). Этот файл содержит список зависимостей python, необходимые для установки.

`RUN pip install –no-cache-dir -r requirements.txt` – запускает команду «pip install», чтобы установить зависимости python, перечисленные в файле «requirements.txt». Флаг «--no-cache-dir» используется для установки пакетов без кэша, что помогает уменьшить размер образа Docker.

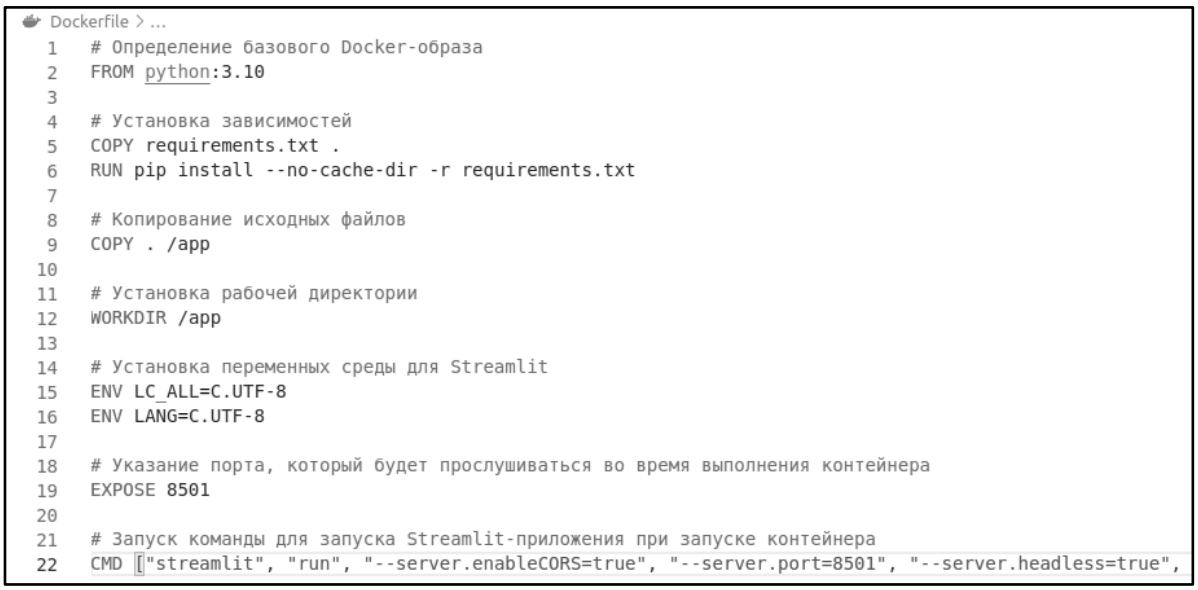

Рисунок 35 – Созданный Dockerfile

`COPY . /app` – устанавливает директорию «/app» в качестве текущей рабочей директории для последующих инструкций Docker. Все последующие команды будут выполнятся относительно этой директории.

`ENV LC_ALL=C.UTF-8` и `ENV LANG=C.UTF-8` – устанавливают переменные окружения в контейнере для установки локали UTF-8, что помогает корректно отображать различные языковые символы.

`EXPOSE 8501` – экспонирует порт 8501 из контейнера на хостовую машину, что позволяет обращаться к приложению, использующему этот порт извне.

`CMD […]` – задает команду, которая будет выполнена при запуске контейнера. Здесь выполняется запуск приложения Streamlit с указанными параметрами. Опции, такие как «--server.enableCORS=true».

После настройки файла Dockerfile нужно собрать образ проекта. Это делается с помощью ввода в консоль команды «docker build» после чего

Docker начинает процесс сборки образа контейнера. Шаги сборки образа описаны ниже.

1. Чтение Dockerfile. Docker считывает файл Dockerfile в текущем каталоге, который содержит инструкции для построения контейнера.

2. Создание временного образа. Docker создает временный контейнер для обработки команд, указанных в Dockerfile. Этот шаг включает выполнение инструкций, таких как «RUN», «COPY», и других, перечисленных в Dockerfile.

3. Кэширование результатов. Docker использует кэш результатов предыдущих команд в Dockerfile для ускорения сборки. Если какая-то команда не изменилась с предыдущего запуска сборки, Docker использует сохраненный результат из кэша.

4. Окончательная сборка образа. После выполнения всех инструкций Dockerfile и создания временного контейнера согласно указаниям, Docker создает окончательный образ контейнера.

5. Завершение сборки. После завершения сборки образа получается готовый образ Docker, с которым можно работать: запускать контейнеры, распространять его и т.д.

Пример выполнения команды «docker build» показан на рисунке 36.

```
divine@divinelaptop:~/PycharmProjects/stream$ sudo docker buildx build -t my_app /home/divine/PycharmProjects/stream/
[+] Building 101.7s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 946B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/python:3.10
 => [1/5] FROM docker.io/library/python:3.10@sha256:b54e76c629a98430ac9c92e4f6bddeb672396a895b44a85022d12ee2f7239144
 => [internal] load build context
 => => transferring context: 1.62MB
 => CACHED [2/5] COPY requirements.txt .
 => [3/5] RUN pip install --no-cache-dir -r requirements.txt
 => [4/5] COPY . /app
 => [5/5] WORKDIR /app
 => exporting to image
 => => exporting layers
 => => writing image sha256:3460d50fd54e936978805b0a8d82d5a2955192ea62d93fbf4a8f31373a5699c3
 => => naming to docker.io/library/my_app
```

Рисунок 36 – Выполнение команды «docker build»

4. ТЕСТИРОВАНИЕ

4.1. Тестирование алгоритмов машинного обучения

Для тестирования алгоритмов машинного обучения проводилось A/B тестирование.

В машинном обучении A/B тестирование может использоваться для определения, какой алгоритм или модель работает наилучшим образом для определенной задачи. В этом случае могут быть созданы различные модели машинного обучения и протестированы на пользовательских данных. После сравнения результатов можно определить, какая модель лучше работает [21].

В ходе A/B тестирования были проведены несколько экспериментов по обучению моделей на разном количестве данных из исходного набора. Эксперименты проводились с целью выявить зависимость количества данных на точность предсказания модели. Результаты экспериментов представлены в таблице 6.

Таблица 6 – Результаты A/B тестирования

Количество данных	**Алгоритм**	**Score**
300	GaussianNB	0,98
	LogisticRegression	0,97
	DecisionTreeClassifier	0,98
	RandomForestClassifier	0,96
	XGBClassifier	0,97
	SVC	0,98
	KNN	0,97
600	GaussianNB	0,96
	LogisticRegression	0,98
	DecisionTreeClassifier	0,98
	RandomForestClassifier	0,99
	XGBClassifier	0,97
	SVC	0,99
	KNN	0,98
900	GaussianNB	0,98
	LogisticRegression	0,99
	DecisionTreeClassifier	0,98
	RandomForestClassifier	0,99
	XGBClassifier	0,98
	SVC	0,96
	KNN	0,97
1200	GaussianNB	0,99

Окончание таблицы 6

Количество данных	Алгоритм	Score
	LogisticRegression	0,99
	DecisionTreeClassifier	0,98
	RandomForestClassifier	0,99
	XGBClassifier	0,98
	SVC	0,96
	KNN	0,99
1500	GaussianNB	0,98
	LogisticRegression	0,96
	DecisionTreeClassifier	0,97
	RandomForestClassifier	0,99
	XGBClassifier	0,99
	SVC	0,97
	KNN	0,97

В ходе экспериментов было установлено, что количество данных не влияют на результат предсказания моделей.

4.2. Тестирование веб-приложения

Функциональное тестирование – это процесс тестирования программного обеспечения, который направлен на проверку соответствия системы заданным функциональным требованиям. Функциональное тестирование позволяет определить, соответствует ли система требованиям заказчика, а также выявить ошибки и недоработки, которые могут привести к некорректной работе системы [20].

4.2.1. Тестирование страницы «обучение»

Таблица 7 – Тестирование страницы «обучение»

Название теста	Шаги	Ожидаемый результат	Тест пройден?
Загрузка набора данных	1. Нажать на кнопку «загрузить файл». 2. Выбрать файл в формате «.csv».	Программа должна вывести название файла.	Да
Выбор алгоритма и гиперпараметров модели	1. Выбрать алгоритм из списка. 2. Настроить значения гиперпараметров.	Значения гиперпараметров должны примениться к модели.	Да

Окончание таблицы 7

Название теста	Шаги	Ожидаемый результат	Тест пройден?
Начать обучение модели	Нажать на кнопку «Начать обучение».	Программа должна вывести на экран слово «Готово!» и точность получившейся модели.	Да
Скачать получившеюся модели	Нажать на кнопку «Скачать модель».	Файл скаченной модели должен появиться в загрузках браузера.	Да

В результате функционального тестирования все полученные результаты соответствуют ожидаемым. Все тесты пройдены успешно.

4.2.2. Тестирование страницы «классификация»

Таблица 8 – Тестирование страницы «классификация»

Название теста	Шаги	Ожидаемый результат	Тест пройден?
Загрузка готовой модели машинного обучения.	1. Нажать на кнопку «Загрузить файл». 2. Выбрать файл.	Программа должна вывести на экран название файла.	Да
Загрузка набора данных	1. Нажать на кнопку «загрузить файл». 2. Выбрать файл в формате «.csv».	Программа должна вывести название файла.	Да
Вывод результата классификации модели.	После загрузки набора данных автоматически запуститься классификация команд.	Результат классификации будет выведен в виде таблицы, где правильные ответы выделены зеленым а неправильные – красным.	Да
Скачать результат классификации.	Нажать на кнопку «Скачать файл».	Файл с результатами классификации в формате «.csv» должен появиться в загрузках браузера.	Да

В результате функционального тестирования все полученные результаты соответствуют ожидаемым. Все тесты пройдены успешно.

4.3. Тестирование Docker контейнера

При тестировании docker контейнера проверим, правильно ли собрался, запускаются ли нужные компоненты внутри этого образа.

При выполнении команды «docker run» в консоли должно появиться сообщение от библиотеки streamlit о том, что приложение запущено (рисунок 37).

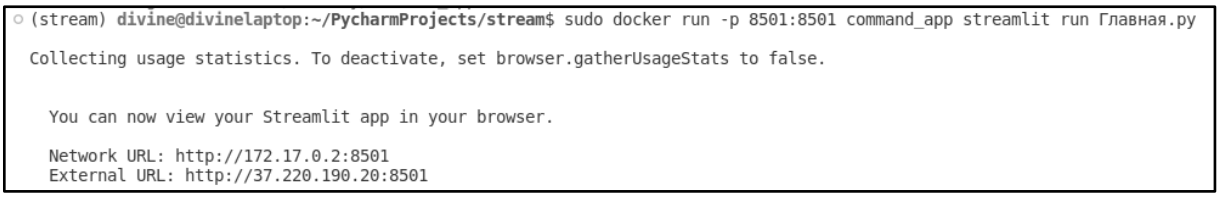

Рисунок 37 – Проверка запуска streamlit внутри docker контейнера

Далее проверим, запустился ли сам контейнер с помощью команды «docker ps». Команда «docker ps» выводит список активных контейнеров в системе. Она позволяет получить информацию о работающих контейнерах, включая их ID, имена, статус, порты и использование ресурсов. Если контейнер работает, то в консоли будет выведена информация о контейнере (рисунок 38):

Рисунок 38 – Проверка запуска docker контейнера

После этого проверим, с правильными ли параметрами запустился docker контейнер. Делается это с помощью команды «docker inspect». После выполнения команды в консоль будут выведены все параметры docker контейнера (рисунок 39).

Рисунок 39 – Проверка параметров docker контейнера

Команда «docker inspect» используется для получения подробной информации о выбранном контейнере. Она отображает различные параметры и настройки контейнера, такие как идентификаторы, размеры, сетевые подключения и другие сведения.

По умолчанию команда выводит данные в формате JSON. Однако можно использовать опцию «--format» для изменения формата вывода информации.

В результатах отображения команды виден «id» образа, его версия, дата сборки этого образа.

Также здесь отображаются параметры переменных, которые были прописаны в «Dokerfile». Например, порт «8501».

Некоторые из переменных остались пустыми, такие как «Author», «DockerVersion» и другие, поскольку их описания в «Dokcerfile» не было.

Так как все три результаты выполнения этих трех команды верны, следовательно, образ работает верно.

ЗАКЛЮЧЕНИЕ

В рамках данной работы было разработано веб-приложение, которое позволяет обучать модели машинного обучения и использовать их для бинарной классификации команд по метрике MITRE. При этом были решены следующие задачи.

1. Проведен обзор научной литературы.
2. Подготовлен обучающий набор данных.
3. Реализованы выбранные методы машинного обучения.
4. Разработано веб-приложение для бинарной классификации вредоносных команд по метрике MITRE.
5. Проведено тестирование разработанного веб-приложения.

Для улучшения работы приложения и повышения качества моделей машинного обучения можно собрать более большой набор данных, чтобы иметь возможность разделить его на тестовую, обучающую и валидационную выборки.

Также можно добавить возможность использования и обучения нейронных сетей.

Также возможно добавить личные кабинеты для пользователей, где будут храниться скаченные, созданные модели машинного обучения, пользовательский набор данных, параметры и результаты последних используемых моделей.

ЛИТЕРАТУРА

1. Bloedorn E.E., Talbot L.M., DeBarr D.D. Data Mining Applied to Intrusion Detection: MITRE Experiences. Machine Learning and Data Mining for Computer Security // Springer, 2020. – 24 с.

2. Xiong W., Legrand E., Åberg O. Cyber security threat modeling based on the MITRE Enterprise ATT&CK Matrix. // Software and Systems Modeling, 2021. – 21 с.

3. Dziech A., Mees W., Czyżewski A. Multimedia Communications, Services and Security. Communications in Computer and Information Science. // MCSS, 2020. – 364 с.

4. Sentuna A., Alsadoon A., Prasad P., Saadeh M., Alsadoon O. A Novel Enhanced Naïve Bayes Posterior Probability (ENBPP) Using Machine Learning: Cyber Threat Analysis. Neural Processing Letters. // Springer Science, 2020. – 33 с. DOI:10.1007/s11063-020-10381-x.

5. Scikit-learn. Machine learning in Python. [Электронный ресурс] URL: https://scikit-learn.org/stable/ (дата обращения: 11.02.2024 г.).

6. Pandas. [Электронный ресурс] URL: https://pandas.pydata.org/ (дата обращения: 11.02.2024 г.).

7. NumPy. [Электронный ресурс] URL: https://numpy.org/ (дата обращения: 11.02.2024 г.).

8. Streamlit. [Электронный ресурс] URL: https://streamlit.io/ (дата обращения: 11.02.2024 г.).

9. Docker. [Электронный ресурс] URL: https://docker.com/ (дата обращения: 11.02.2024 г.).

10. Бурков А. Машинное обучение без лишних слов. // СПб: Питер, 2020. – 192 с.

11. Пример работы бустинга [Электронный ресурс] URL: https://aws.amazon.com/ru/what-is/boosting/ (дата обращения: 11.02.2024 г.).

12. Пример работы KNN. [Электронный ресурс] URL: https://intuitivetutorial.com/2023/04/07/k-nearest-neighbors-algorithm/ (дата обращения: 11.02.2024 г.).

13. Элбон К. Машинное обучение с использованием Python. Сборник рецептов // Пер. с англ. – СПб.: БХВ-Петербург, 2019. – 384 с.

14. Basu A, Basu S. A User's Guide to Business Analytics // Chapman & Hall, 2016. – 400 с.

15. Вигерс К., Битти Д. Разработка требований к программному обеспечению. 3-е изд., дополненное // Пер. с англ. – М.: Издательство «Русская редакция»; СПб.: БХВ-Петербург, 2014. – 736 с.

16. Jupyter Notebook. [Электронный ресурс] URL: https://jupyter.org/ (дата обращения: 11.02.2024 г.).

17. GaussianNB. [Электронный ресурс] URL: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html (дата обращения: 11.02.2024 г.).

18. DecisionTreeClassifier. [Электронный ресурс] URL: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html (дата обращения: 11.02.2024 г.).

19. GridSearchCV. [Электронный ресурс] URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html (дата обращения: 11.02.2024 г.).

20. RandomForestClassifier. [Электронный ресурс] URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (дата обращения: 11.02.2024 г.).

21. LogisticRegression. [Электронный ресурс] URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (дата обращения: 11.02.2024 г.).

22. Brian O. Python Testing with pytest: Simple, Rapid, Effective, and Scalable. // Pragmatic Bookshelf, 2022. – 248 с.

23. Хултен Дж. Разработка интеллектуальных систем. // пер. с анг. В. С. Яценкова. – М.: ДМК Пресс, 2019. – 284 с.

24. XGBoost – XGBoost Documentation [Электронный ресурс] URL:https://xgboost.readthedocs.io/en/stable/parameter.html (дата обращения: 11.02.2024 г.).

25. SVC. [Электронный ресурс] URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html (дата обращения: 11.02.2024 г.).

26. KNN. [Электронный ресурс] URL: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html (дата обращения: 11.02.2024 г.).

ПРИЛОЖЕНИЕ. Спецификации вариантов использования

Спецификации вариантов использования (ВИ) системы приведена в таблицах 1–9.

Таблица 1 – Спецификация варианта использования «Обучение модели»

Прецедент: Обучение модели
ID: 1
Аннотация: Обучить новую модель для классификации команд.
Главные актеры: Пользователь.
Второстепенные актеры: Нет.
Предусловия: Нет.
Основной поток: 1. Поток инициируется, когда пользователь выбирает в боковом меню пункт «обучение». 2. Система открывает страницу для обучения моделей.
Постусловия: Была открыта страница для обучения моделей.
Альтернативные потоки: Нет.

Таблица 2 – Спецификация варианта использования «Загрузить набор данных»

Прецедент: Загрузить набор данных
ID: 2
Аннотация: Загрузка набора данных.
Главные актеры: Пользователь.
Второстепенные актеры: Нет.
Предусловия: Была открыта страница «обучение модели» или «классификация».
Основной поток: 1. Поток инициируется, когда пользователь нажимает на кнопку «просмотр файлов». 2. Система открывает окно с выбором файлов. 3. Пользователь выбирает нужный файл.
Постусловия: Был загружен набор данных.
Альтернативные потоки: Нет.

Таблица 3 – Спецификация варианта использования «Выбрать алгоритм»

Прецедент: Выбрать алгоритм.
ID: 3
Аннотация: Выбор алгоритма для модели.
Главные актеры: Пользователь.
Второстепенные актеры: Нет.
Предусловия: Был загружен набор данных.
Основной поток: 1. Поток инициируется после того, как пользователь загрузил файл с набором данных. 2. Система открывает выпадающее меню с возможностью выбора алгоритма. 3. Пользователь выбирает нужный алгоритм.
Постусловия: Был выбран алгоритм для обучения модели.
Альтернативные потоки: Нет.

Таблица 4 – Спецификация варианта использования «Скачать модель»

Прецедент: Скачать модель.
ID: 4
Аннотация: Скачивание обученной модели.
Главные актеры: Пользователь.
Второстепенные актеры: Нет.
Предусловия: На выбранном ранее алгоритме была обучена модель.
Основной поток: 1. Поток инициируется, когда система закончила обучение модели. 2. Появляется кнопка «скачать модель», с помощью которой пользователь может скачать получившуюся модель машинного обучения. 3. Файл сохраняется на устройстве.
Постусловия: Была скачана модель машинного обучения.
Альтернативные потоки: Нет.

Таблица 5 – Спецификация варианта использования «Посмотреть информацию про набор данных»

Прецедент: Посмотреть информацию про набор данных.
ID: 5
Аннотация: Просмотр информации о наборе данных, используемого для обучения и классификации.
Главные актеры: Пользователь.
Второстепенные актеры: Нет.
Предусловия: Нет.
Основной поток: 1. Поток инициируется, когда пользователь выбирает в боковом меню пункт «набор данных». 2. Система открывает страницу с информацией о наборе данных.
Постусловия: Была открыта страница с информацией о наборе данных.
Альтернативные потоки: Нет.

Таблица 6 – Спецификация варианта использования «Посмотреть теорию»

Прецедент: Посмотреть теорию.
ID: 6
Аннотация: Просмотр теории про используемые алгоритмы машинного обучения.
Главные актеры: Пользователь.
Второстепенные актеры: Нет.
Предусловия: Нет.
Основной поток: 1. Поток инициируется, когда пользователь нажимает на кнопку «теория». 2. Система открывает страницу с теорией про используемые алгоритмы.
Постусловия: Была открыта страница с теорией про используемые алгоритмы.
Альтернативные потоки: Нет.

Таблица 7 – Спецификация варианта использования «Классификация команд»

Прецедент: Классификация команд.
ID: 7
Аннотация: Классификация команд с помощью готовых моделей машинного обучения.
Главные актеры: Пользователь.
Второстепенные актеры: Нет.
Предусловия: Нет.
Основной поток: 1. Поток инициируется, когда пользователь выбирает в боковом меню пункт «классификация». 2. Система открывает страницу для классификации команд.
Постусловия: Была открыта страница для классификации команд.
Альтернативные потоки: Нет.

Таблица 8 – Спецификация варианта использования «Выбрать готовую модель»

Прецедент: Выбрать готовую модель.
ID: 8
Аннотация: Выбор модели машинного обучения для классификации команд.
Главные актеры: Пользователь.
Второстепенные актеры: Нет.
Предусловия: Открыта страница для классификации команд.
Основной поток: 1. Поток инициируется, когда пользователь нажимает на кнопку «просмотр файлов». 2. Система открывает окно с выбором файлов. 3. Пользователь выбирает нужный файл с заранее обученной моделью машинного обучения.
Постусловия: Выбрана модель машинного обучения.
Альтернативные потоки: Нет.

Таблица 9 – Спецификация варианта использования «Скачать файл с ответами модели»

Прецедент: Скачать файл с ответами модели.
ID: 9
Аннотация: Скачать .csv файл с ответами модели машинного обучения.
Главные актеры: Пользователь.
Второстепенные актеры: Нет.
Предусловия: Модель машинного обучения классифицировала команды.
Основной поток: 1. Поток инициируется, когда система закончила классификацию команд. 2. Пользователь нажимает на кнопку «скачать ответы». 3. «.csv» файл с ответами скачивается на устройство.
Постусловия: Были скачены ответы модели машинного обучения.
Альтернативные потоки: Нет.