

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

Разработка веб-приложения для бинарной классификации обфусцированных команд PowerShell с использованием алгоритмов машинного обучения

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

Автор:
студент группы КЭ-401
М.И. Скоблюк

Челябинск, 2024 г.

ОБФУСКАЦИЯ

Обфускация – это техника, которая заключается в изменении исходного кода программы, чтобы сделать его трудным для чтения и понимания

Техники обфускации:

1. `short` – использование сокращений
2. `variables` – манипуляции с переменными среды
3. `encoding` – изменение кодировки
4. `string` – манипуляции со строками
5. `symbol` – использование игнорируемых символов

АКТУАЛЬНОСТЬ

- Обфускация часто используется во вредоносном программном обеспечении (ПО)
- В 2016 году PowerShell стал открытым ПО
- С 2016 по 2017 год число вредоносного ПО, использующего PowerShell, выросло на 432%
- За 2020 год это число увеличилось почти в 12 раз по сравнению с 2019 годом
- В конце 2022 года PowerShell и обфусцированные файлы являлись двумя из трех самых популярных техник для кибератак
- В конце 2023 года известная группировка (Ex)Cobalt использовала обфускацию PowerShell в своей атаке
- Актуальные вредоносные файлы используют обфускацию, например, Gootkit

ЦЕЛЬ И ЗАДАЧИ ИССЛЕДОВАНИЯ

Цель: разработать веб-приложение для бинарной классификации обфусцированных команд PowerShell с использованием алгоритмов машинного обучения

Задачи:

1. Провести обзор научной литературы
2. Подготовить обучающий набор данных
3. Реализовать выбранные методы машинного обучения
4. Разработать веб-приложение для бинарной классификации обфусцированных команд PowerShell
5. Провести тестирование разработанного веб-приложения

АНАЛИЗ ЛИТЕРАТУРЫ

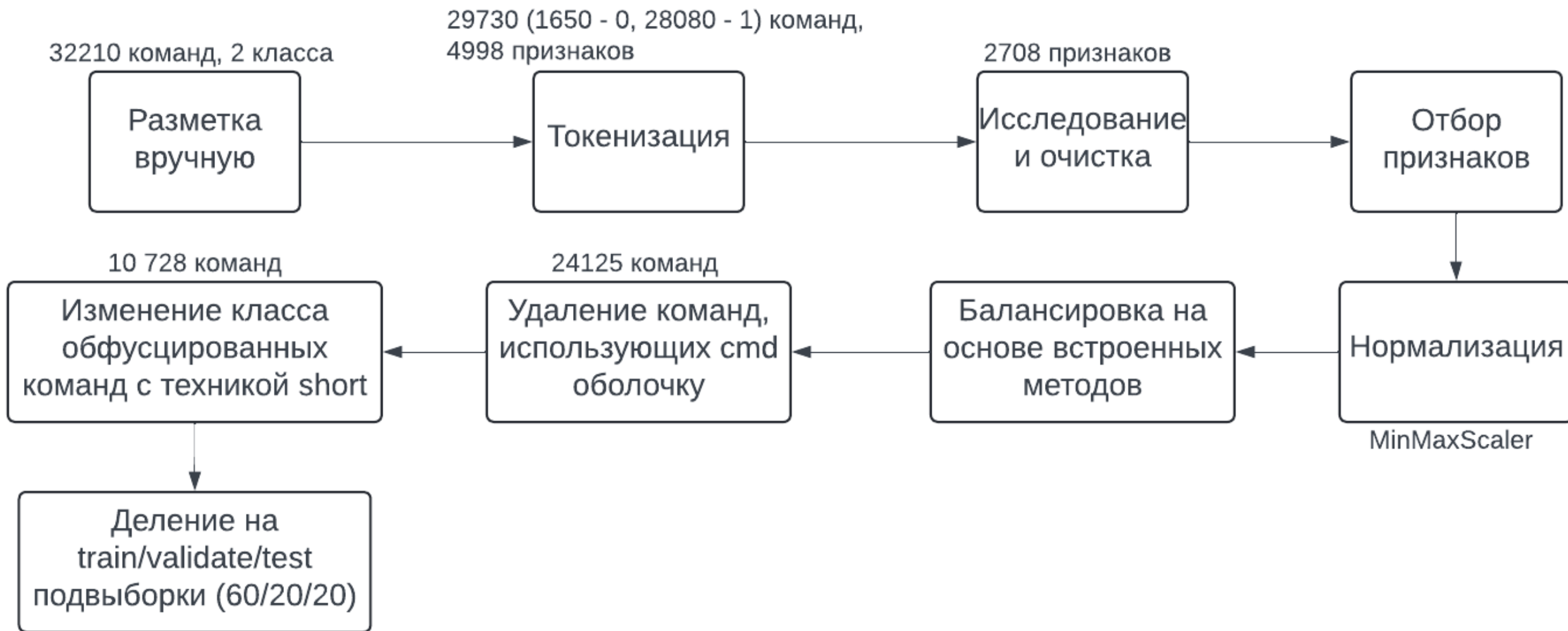
Название	Авторы	Набор данных	Алгоритм	Точность
PowerDP: De-Obfuscating and Profiling Malicious PowerShell Commands With Multi-Label Classifiers	Meng-Han Tsai, Chia-Ching Lin, Zheng-Gang He, Wei-Chieh Yang, Chin-Laung Lei	Собран вручную из данных песочницы и открытых источников	Random Forest, Deep Neural Network	97,56% 99,82%
Detecting Malicious PowerShell Commands using Deep Neural Networks	Danny Hendler, Shay Kels, Amir Rubin	База знаний компании Microsoft	N-gram, Convolutional Neural Network	99,49%
AST-Based Deep Learning for Detecting Malicious PowerShell	Gili Rusak, Abdullah Al-Dujaili, Una-May O'Reilly	Открытые наборы данных Palo Alto Networks	Random Forest	85%
Evaluations of AI-based malicious PowerShell detection with feature optimizations	Jihyeon Song, Jungtae Kim, Sunoh Choi, Jonghyun Kim, Ikkyun Kim	Собран вручную из открытых источников и данных компании ESTsecurity	Random Forest, Convolutional Neural Network	98,90% 98,50%

СРЕДСТВА РАЗРАБОТКИ



- **Языки программирования:** Python 3.10, TypeScript 4.9
- **Среда разработки исходного кода:** IDE PyCharm 2023.1.4
- **Среда разработки моделей машинного обучения:** Jupyter Notebook
- **Библиотеки для машинного обучения:** scikit-learn, xgboost, catboost, lightgbm, pandas, numpy, matplotlib, seaborn

НАБОР ДАННЫХ



АЛГОРИТМЫ МАШИННОГО ОБУЧЕНИЯ



1. Наивный Байес (Multinomial/Gaussian Naive Bayes)
2. Метод опорных векторов (Support Vector Machines)
3. Логистическая регрессия (Logistic Regression)
4. К-ближайших соседей (K-Nearest Neighbors)
5. Дерево решений (Decision Tree)
6. Случайный лес (Random Forest)
7. Бустинг (XGBoost, CatBoost, LightGBM)

ОБУЧЕНИЕ МОДЕЛЕЙ



Объем предобработанных данных: 10728 записей.

- Обучение: 60% (6 436 записи)
- Валидация: 20% (2 146 записи)
- Тестирование: 20% (2 146 записи)

Обучение производилось с помощью локальных возможностей:

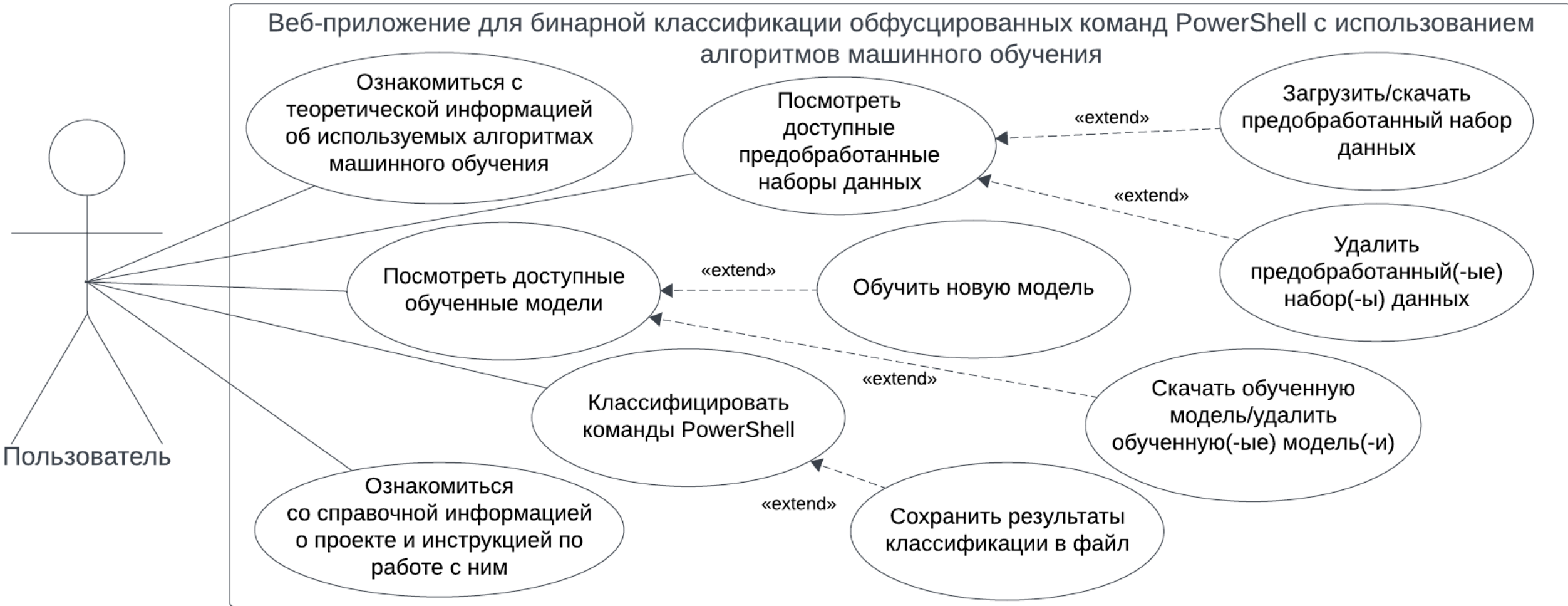
- GPU: NVIDIA GeForce GTX 1050 2GB
- CPU: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz

ТЕСТИРОВАНИЕ ОБУЧЕННЫХ МОДЕЛЕЙ



Алгоритм	Accuracy	Recall	Precision	F1 Score
Multinomial Naive Bayes	0,9082	0,8295	0,9845	0,9003
Gaussian Naive Bayes	0,9841	0,9888	0,9797	0,9842
Support Vector Machines	0,9846	0,9767	0,9924	0,9844
K-Nearest Neighbors	0,9622	0,9562	0,9682	0,9619
Logistic Regression	0,9743	0,9599	0,9885	0,9739
Decision Tree	0,9911	0,9888	0,9934	0,9906
Random Forest	0,9836	0,9822	0,9916	0,9864
XGBoost	0,9925	0,9925	0,9925	0,9925
CatBoost	0,9902	0,9888	0,9916	0,9901
LightGBM	0,9883	0,9869	0,9897	0,9883

ДИАГРАММА ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ



АРХИТЕКТУРА ВЕБ-ПРИЛОЖЕНИЯ

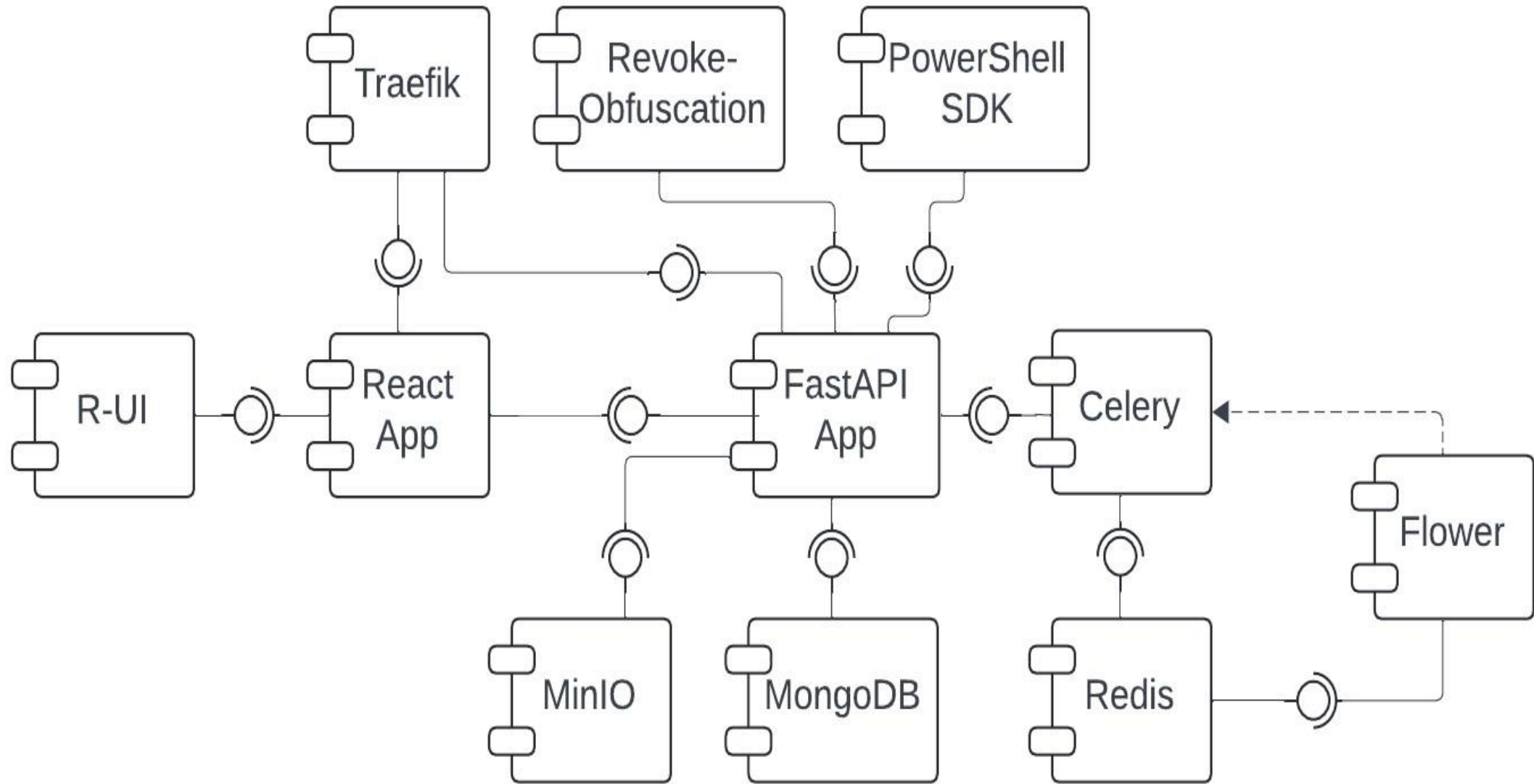
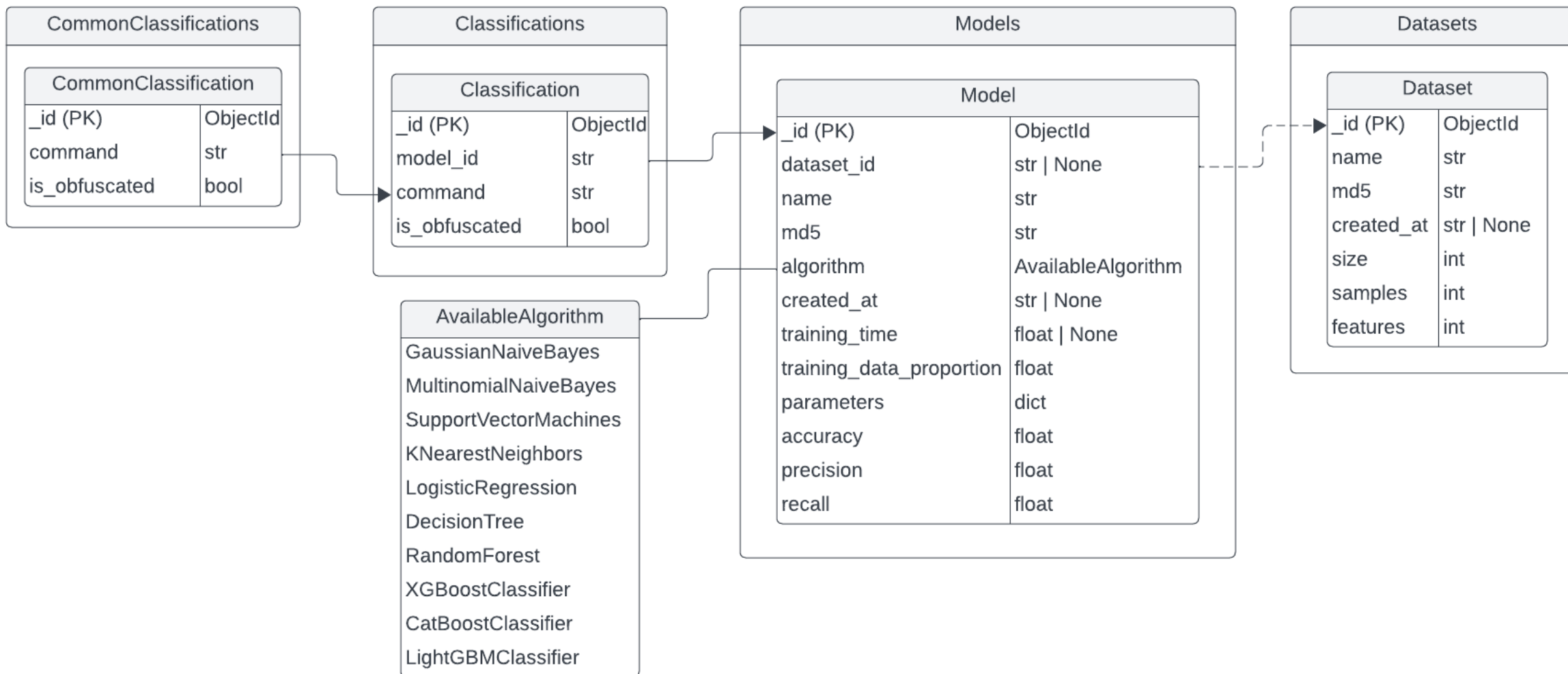


СХЕМА БАЗЫ ДАННЫХ



DOCKER



Сервис	Назначение	Образ
backend	Запуск FastAPI приложения	obfuscation-detecting-backend
frontend	Запуск React приложения	obfuscation-detecting-frontend
traefik	Выступает в роли прокси и служит для управления и балансировки трафиком	traefik:v2.10
minio	Запуск приложения MinIO для хранения файлов	minio/minio
create_buckets	Создание необходимых bucket-ов в MinIO	minio/mc
mongo	Запуск базы данных MongoDB для хранения документов	mongo:5.0
worker	Запуск обработчика задач	obfuscation-detecting-backend
flower	Запуск приложение для мониторинга выполнения задач	mher/flower
redis	Запуск базы данных Redis для хранения информации о задачах	redis:alpine

ГЛАВНАЯ СТРАНИЦА



The screenshot shows a web browser window with the following elements:

- Browser Tab:** Obfuscation Detecting App
- Address Bar:** localhost:3000
- Page Header:** ObfuscationDetecting (left), Главная страница (center), Профиль (right)
- Left Sidebar:**
 - Главная страница (selected)
 - Используемые алгоритмы
 - Наборы данных
 - Обучение
 - Классификация
 - О проекте
- Main Content Area:**
 - Obfuscation Detecting**
 - Бинарная классификация обфусцированных команд PowerShell с использованием алгоритмов машинного обучения
- Bottom Left:** Свернуть (collapse icon)

СТРАНИЦА ИСПОЛЬЗУЕМЫЕ АЛГОРИТМЫ



Obfuscation Detecting App x +

localhost:3000/algorithms/mnb

ObfuscationDetecting

Используемые алгоритмы

Профиль

- Главная страница
- Используемые алгоритмы
 - Multinomial Naive Bayes**
 - Gaussian Naive Bayes
 - Support Vector Machines
 - K-Nearest Neighbors
 - Logistic Regression
 - Decision Tree
 - Random Forest
 - XGBoost
 - CatBoost
 - LightGBM
- Наборы данных
- Обучение
- Свернуть

Multinomial Naive Bayes

Метод наивного Байеса – это статический алгоритм машинного обучения, основанный на теореме Байеса и используемый в основном для классификации текстовых данных. Алгоритм предполагает, что все признаки независимы друг от друга, что делает его «наивным».

Пусть имеется набор данных, содержащий M признаков, объекты которого могут принадлежать N различным классам. Существует конечное множество классов $\{c_1, \dots, c_N\}$. Необходимо определить такой класс c_k , чтобы вероятность принадлежности объекта $X = (x^{(1)}, \dots, x^{(M)})$ данному классу была максимальна: $c_k = \operatorname{argmax}_{c_i} [P(c_i | X)]$.

Вероятность принадлежности объекта X классу c_k вычисляется по формуле Байеса: $P(c_k | X) = P(c_k) * P(X | c_k) / P(X)$, где $P(X | c_k)$ – совокупная вероятность всех признаков для класса c_k .

Так как признаки независимы, то вероятности $P(X)$ и $P(X | c_k)$ вычисляются по следующим формулам: $P(X) = \prod_{i=1}^M P(x^{(i)})$ и $P(X | c_k) = \prod_{i=1}^M P(x^{(i)} | c_k)$.

В случае непрерывных случайных величин предполагается использовать вместо вероятности $P(X | c_k)$ плотность распределения $f(X | c_k)$.

Данный метод является достаточно быстрым и, несмотря на строгость независимости признаков, эффективным при анализе текста.

Multinomial Naive Bayes – это алгоритм машинного обучения, основанный на методе наивного Байеса. Применяется в случае полиномиального распределения признаков в используемых данных.

Алгоритм содержит важный гиперпараметр `smoothing`, который незаменим в случаях, когда значение какого-либо признака объекта равно 0. Полученное нулевое значение способствует нулевой вероятности признака для определенного класса, которая при перемножении аннулирует совокупную вероятность объекта для данного класса. Гиперпараметр «сглаживает» значение вероятности, не допуская аннулирования совокупной вероятности путем добавления указанного значения к значениям всех признаков всех объектов и пересчета вероятностей, учитывая обновленные значения.

СТРАНИЦА НАБОРЫ ДАННЫХ



Obfuscation Detecting App

localhost:3000/datasets

ОбfuscationDetecting

Наборы данных

Профиль

+ Добавить

Удалить

<input type="checkbox"/>	Имя файла	Размер файла, байты	Количество объектов	Количество признаков	
<input type="checkbox"/>	source.csv	170594821	10728	2708	↓
<input type="checkbox"/>	5000_samples.csv	77195602	5000	2708	↓

Свернуть

1 из 1 20

Всего: 2

СТРАНИЦА ОБУЧЕНИЕ



Obfuscation Detecting App

localhost:3000/learning

Обучение

Профиль

+ Добавить Удалить

<input type="checkbox"/>	Имя файла	Используемый алг	Набор данных	Объем обучающей	Время обучения, с	Accuracy	Precision	Recall
<input type="checkbox"/>	SupportVectorMachines_model.pkl	SupportVectorMachines	source.csv	0.6		0.984624	0.992456	0.976718
<input type="checkbox"/>	MultinomialNB_model.pkl	MultinomialNaiveBayes	source.csv	0.6		0.908204	0.98455	0.829589
<input type="checkbox"/>	KNearestNeighbors_model.pkl	KNearestNeighbors	source.csv	0.6		0.962256	0.968285	0.956205
<input type="checkbox"/>	RandomForestClassifier_model.pkl	RandomForest	source.csv	0.6		0.983686	0.991636	0.982299
<input type="checkbox"/>	XGBoostClassifier_model.pkl	XGBoostClassifier	source.csv	0.6		0.992545	0.992579	0.992545
<input type="checkbox"/>	LogisticRegression_model.pkl	LogisticRegression	source.csv	0.6		0.974375	0.988519	0.959956
<input type="checkbox"/>	CatBoostClassifier_model.pkl	CatBoostClassifier	source.csv	0.6		0.990214	0.991601	0.988815
<input type="checkbox"/>	DecisionTreeClassifier_model.pkl	DecisionTree	source.csv	0.6		0.991145	0.993492	0.988819
<input type="checkbox"/>	GaussianNB_model.pkl	GaussianNaiveBayes	source.csv	0.6		0.984156	0.979776	0.988824
<input type="checkbox"/>	test_gnb.pkl	GaussianNaiveBayes	5000_samples.csv	0.8	0.443308	0.986	0.987755	0.98374

Свернуть

1 из 1 20

Всего: 11

СТРАНИЦА КЛАССИФИКАЦИЯ



Obfuscation Detecting App x +

localhost:3000/classifications

Обфускация

Профиль

Главная страница

Используемые алгоритмы

Наборы данных

Обучение

Классификация

О проекте

Свернуть

+ Добавить

Сохранить результаты

Команда PowerShell	Результат классификации
Backup-Gpo -All -Path E:GPObackup	Не обфусцирована
Start-Service	Не обфусцирована
Clear content	Не обфусцирована
Search-ADAccount -LockedOut	Не обфусцирована
Measure-Object	Не обфусцирована
Get-ADGroupMember -identity ГҮВҮВҮьHR FullГҮВҮВк	Не обфусцирована
For-Each Object	Не обфусцирована
Enter-PSSession -ComputerName	Не обфусцирована
Get-Service	Не обфусцирована
Invoke-Expression	Не обфусцирована
Stop-Service	Не обфусцирована

Всего: 98

СТРАНИЦА О ПРОЕКТЕ



The screenshot shows a web browser window with the following elements:

- Browser Tab:** Obfuscation Detecting App
- Address Bar:** localhost:3001/about/contacts
- Page Header:** ObfuscationDetecting (left) and Профиль (right)
- Left Sidebar (Navigation Menu):**
 - Главная страница
 - Используемые алгоритмы
 - Наборы данных
 - Обучение
 - Классификация
 - О проекте (highlighted)
 - Справочная информация
 - Документация
 - Контакты
- Main Content Area:**
 - О проекте**
 - Контакты**
 - Telegram: профиль в Telegram
 - GitHub: аккаунт в GitHub
 - VK: страница в VK
- Bottom Left:** Свернуть

MINIO



MinIO Console

localhost:9001/browser

Object Browser

Filter Buckets

Name	Objects	Size	Access
datasets	1	380.2 KiB	R/W
models	10	146.2 MiB	R/W

FLOWER



The screenshot shows the Flower web interface in a browser window. The address bar displays 'localhost:5555'. The interface has a dark green header with navigation links: 'Flower', 'Workers', 'Tasks', 'Broker', and 'Documentation'. Below the header, there is a control for showing 15 workers and a search input field. A table displays the status of a single worker, 'celery@9fb76c782c04', which is 'Online'. The table also shows summary statistics for the worker, including 0 active tasks, 4 processed tasks, 0 failed tasks, 5 succeeded tasks, and 0 retried tasks. The load average is listed as 0.12, 0.73, 0.86. At the bottom of the table, it indicates 'Showing 1 to 1 of 1 workers' and includes 'Previous' and 'Next' navigation buttons.

Worker	Status	Active	Processed	Failed	Succeeded	Retried	Load Average
celery@9fb76c782c04	Online	0	4	0	5	0	0.12, 0.73, 0.86
Total		0	4	0	5	0	

Showing 1 to 1 of 1 workers

Previous 1 Next

API СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ (1)

Метод	Путь	Параметры	Принцип работы	Результат
GET	/algorithms/{algorithm_name}/schemas	Path: algorithm_name (название алгоритма)	Для выбранного алгоритма формируется JSON-схема параметров обучения, которая будет использоваться в генераторе форм	JSON-схема параметров обучения выбранного алгоритма
POST	/datasets/	Body: datasets (список файлов)	Добавление в очередь задачи загрузки файлов в базу данных и объектное хранилище	Ответ в формате JSON, возвращающий 201 статус (операция запущена)
GET	/datasets/	Query: page (сдвиг), size (лимит)	Загрузка документов из базы данных, содержащих информацию о наборах данных	Список документов и мета-информация о пагинации
DELETE	/datasets/	Query: ids (список id наборов данных в базе данных)	Добавление в очередь задач удаления наборов данных с заданными id из базы данных и объектного хранилища, а также обновление всех связанных моделей (удаление информации о наборе данных)	Ответ в формате JSON, возвращающий 201 статус (операция запущена)
GET	/datasets/{id}/download	Path: id (id набора данных в базе данных)	Поиск набора данных с заданным id в базе данных и скачивание его файла из объектного хранилища	CSV-файл

API СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ (2)

Метод	Путь	Параметры	Принцип работы	Результат
POST	/models/	Body: filename, dataset_id, training_params, algorithm_name, training_data_proportion	Добавление в очередь задач задачи обучения выбранного алгоритма на заданных параметрах и наборе данных, тестирования модели на оставшихся данных и сохранения результатов в базу данных и объектное хранилище	Ответ в формате JSON, возвращающий 201 статус (операция запущена)
GET	/models/	Query: page (сдвиг), size (лимит)	Загрузка документов из базы данных, содержащих информацию об обученных моделях, поиск связанных наборов данных и возвращение DTO объектов	Список DTO объектов и мета-информация о пагинации
DELETE	/models/	Query: ids (список id моделей в базе данных)	Добавление в очередь задач задачи удаления обученных моделей с заданными id из базы данных и объектного хранилища, а также всех классификаций (включая пересчет совокупной классификации команд)	Ответ в формате JSON, возвращающий 201 статус (операция запущена)
GET	/models/{id}/download	Path: id (id модели в базе данных)	Поиск обученной модели с заданным id в базе данных и скачивание её файла из объектного хранилища	Pickle-файл

API СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ (3)

Метод	Путь	Параметры	Принцип работы	Результат
POST	/classifications/	Body: models_ids, commands	Добавление в очередь задач задачи классификации входных данных на заданных обученных моделях, включая подсчет совокупного результата обфусцированности команд на основе результатов всех моделей	Ответ в формате JSON, возвращающий 201 статус (операция запущена)
GET	/classifications/	Query: limit (максимальное число объектов)	Загрузка заданного числа документов, содержащих информацию о классификациях команд, из базы данных	Список документов, содержащих информацию о классификациях команд, размер которого не превышает limit и общее число доступных классификаций
GET	/classifications/commands	Query: command (команда)	Загрузка всех документов из базы данных, содержащих информацию о классификации заданной команды каждой выбранной отдельно взятой обученной моделью	Список документов с информацией о классификациях заданной команды на отдельно взятых моделях
GET	/classifications/download	Query: filename (имя файла)	Формирование результатов классификации всех команд в совокупности и на отдельно взятых выбранных моделях в виде файла	CSV-файл с заданным именем

ФУНКЦИОНАЛЬНОЕ ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ



10 тест-кейсов:

- Загрузка/скачивание/удаление предобработанных данных
- Обучение модели
- Генерация формы с параметрами обучения
- Скачивание обученной модели
- Удаление обученных моделей
- Классификация команд
- Подробные результаты классификации команды
- Сохранение результатов классификации

Все тесты были пройдены успешно

ОСНОВНЫЕ РЕЗУЛЬТАТЫ

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

ОБФУСКАЦИЯ

Техника обфускации	Обфусцированная команда	Необфусцированная команда
short – использование сокращений	<pre>powershell -NoExi "\$os = Get-WmiObject win32_operatingsystem \$uptime = (Get-Date) - \$os.ConvertToDateTime(\$os.LastBootUpTime) Write-Output ("Last boot: \" + \$os.ConvertToDateTime(\$os.LastBootUpTime))"</pre>	<pre>\$os = Get-WmiObject win32_operatingsystem \$uptime = (Get- Date) - \$os.ConvertToDateTime(\$os.LastBootUpTi me) Write-Output ("Last boot: " + \$os.ConvertToDateTime(\$os.LastBootUpTi me))</pre>
variables – манипуляции с переменными среды	<pre>C:\WINDOWS\system32\cmd.exe/c "set fnqj=\$os = Get-WmiObject win32_operatingsystem \$uptime = (Get-Date) - \$os.ConvertToDateTime(\$os.LastBootUpTime) Write-Output ("Last boot: " + \$os.ConvertToDateTime(\$os.LastBootUpTime)) &&set bhpo=echo \${ExecutionContext}.InvokeCommand.InvokeScript((Item env:fnqj).Value) ^ powershell \${Input}^^^ Invoke- Expression&&C:\WINDOWS\system32\cmd.exe/c%bhpo%"</pre>	<pre>\$os = Get-WmiObject win32_operatingsystem \$uptime = (Get- Date) - \$os.ConvertToDateTime(\$os.LastBootUpTi me) Write-Output ("Last boot: " + \$os.ConvertToDateTime(\$os.LastBootUpTi me))</pre>
encoding – изменение кодировки	<pre>Invoke-Expression (('Add-ADG'+r'+oupMembe'+r+' -Ide'+ntit'+y group'+-name -Me'+mbers+' S'+se'+r1, us'+e'+r2'))</pre>	<pre>Add-ADGroupMember -Identity group- name -Members Sser1, user2</pre>
string – манипуляции со строками	<pre>\$members = Import-CSV ("{0}{5}{4}{1}{2}{3}" -f'c:itad','o- gro','up','.csv','t','d-') Select-Object -ExpandProperty ("{0}{1}{2}{3}" - f'samacc','ount','n','ame') ("{0}{3}{2}{1}" -f'A','r','oupMembe','dd-ADGr') -Identity ("{2}{1}{0}" -f 'e-rw','v','hr-n-dri') -Members \$members</pre>	<pre>\$members = Import-CSV c:itadd-to- group.csv Select-Object -ExpandProperty samaccountname Add-ADGroupMember - Identity hr-n-drive-rw -Members \$members</pre>
symbol – использование игнорируемых символов	<pre>Add-AD`Grou`pMe`mber -Identity group-name -Members Sser1, user2</pre>	<pre>Add-ADGroupMember -Identity group- name -Members Sser1, user2</pre>

ПАРАМЕТРЫ ОБУЧЕННЫХ МОДЕЛЕЙ

Алгоритм	Набор параметров
Multinomial Naive Bayes	alpha=1e-05
Gaussian Naive Bayes	var_smoothing=1e-06
Support Vector Machines	kernel=«linear», C=0.1, degree=3, gamma=«scale»
K-Nearest Neighbors	metric=«minkowski», n_neighbors=5, weights=«uniform»
Logistic Regression	solver=«liblinear», penalty=«l1», C=0.1, multi_class=«ovr», max_iter=1000
Decision Tree	criterion=«entropy», max_depth=20, min_samples_leaf=4, min_samples_split=2
Random Forest	n_estimators=100, max_depth=80, min_samples_leaf=2, min_samples_split=10
XGBoost	n_estimators=500, max_depth=9, learning_rate=0.01
CatBoost	iterations=100, learning_rate=0.05, depth=6
LightGBM	num_leaves=31, learning_rate=0.01, max_depth=5, n_estimators=50

ФУНКЦИОНАЛЬНОЕ ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ (1)

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
1	Загрузка предобработанных наборов данных	<ol style="list-style-type: none">1. Выбрать в боковой панели вкладку «Наборы данных».2. Нажать на кнопку «Добавить».3. В появившемся окне нажать на кнопку «Выбрать файл».4. Выбрать один или несколько предобработанных наборов данных в формате CSV.5. Нажать на кнопку «Подтвердить».	Наборы данных загружены в базу данных и объектное хранилище. Информация о загруженных файлах спустя некоторое время появится в таблице.	Да
2	Скачивание предобработанного набора данных	<ol style="list-style-type: none">1. Выбрать в боковой панели вкладку «Наборы данных».2. Нажать на кнопку скачивания в строке таблицы у набора данных.	Скачан файл в формате CSV, содержащий набор данных.	Да
3	Удаление предобработанных наборов данных	<ol style="list-style-type: none">1. Выбрать в боковой панели вкладку «Наборы данных».2. Заполнить чекбоксы в строках таблицы у наборов данных.3. Нажать на кнопку «Удалить».4. В появившемся окне нажать на кнопку «Удалить».	Наборы данных удалены из базы данных, объектного хранилища и таблицы. Во всех документах связанных с набором данных обученных моделях очищено поле, содержащее id набора данных.	Да

ФУНКЦИОНАЛЬНОЕ ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ (2)

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
4	Обучение моделей	<ol style="list-style-type: none">1. Выбрать в боковой панели вкладку «Обучение».2. Нажать на кнопку «Добавить».3. Выбрать алгоритм для обучения.4. Заполнить имя сериализованного файла.5. Выбрать набор данных для обучения.6. Заполнить объем обучающей выборки.7. Заполнить параметры обучения.8. Нажать на кнопку «Подтвердить».	Модель выбранного алгоритма обучилась на заданных параметрах и объеме обучающей выборке выбранного набора данных, а затем была сохранена в базу данных и объектное хранилище.	Да
5	Настройки обучения	<ol style="list-style-type: none">1. Выбрать в боковой панели вкладку «Обучение».2. Нажать на кнопку «Добавить».3. Выбрать алгоритм для обучения.4. Выбрать другой алгоритм для обучения.	Пересобрана форма параметров обучения.	Да
6	Скачивание обученной модели	<ol style="list-style-type: none">1. Выбрать в боковой панели вкладку «Обучение».2. Нажать на кнопку скачивания в строке таблицы у обученной модели.	Скачан файл в формате rkl с сериализованной обученной моделью.	Да

ФУНКЦИОНАЛЬНОЕ ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ (3)

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
7	Удаление обученных моделей	<ol style="list-style-type: none">1. Выбрать в боковой панели вкладку «Обучение».2. Заполнить чекбоксы в строках таблицы у обученных моделей.3. Нажать на кнопку «Удалить».4. В появившемся окне нажать на кнопку «Удалить».	Обученные модели удалены из таблицы, базы данных и объектного хранилища. Все связанные с удаленными моделями классификации удалены из базы данных, а для команд, присутствующих в удаленных классификациях пересчитаны результаты совокупной классификации, которые затем обновятся в таблице.	Да
8	Классификация команд	<ol style="list-style-type: none">1. Выбрать в боковой панели вкладку «Классификация».2. Нажать на кнопку «Добавить».3. В появившемся окне нажать на кнопку «Выбрать файл».4. Выбрать файла в формате CSV, содержащий предобработанные команды.5. Выбрать обученные модели для классификации.6. Нажать на кнопку «Подтвердить».	Предыдущие классификации, если они есть, удаляются из таблицы. Появляются новые результаты в количестве, указанном внизу таблицы.	Да

ФУНКЦИОНАЛЬНОЕ ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ (4)

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
9	Подробные результаты классификации команды	<ol style="list-style-type: none">1. Выбрать в боковой панели вкладку «Классификация».2. Нажать на строку таблицы с результатами классификации команды.	Для выбранной команды должна появиться карточка с подробными результатами классификации каждой выбранной обученной модели.	Да
10	Сохранение результатов классификации	<ol style="list-style-type: none">1. Выбрать в боковой панели вкладку «Классификация».2. Нажать на кнопку «Сохранить результаты».	Результаты подробной и общей классификации каждой команды сформированы в файл в формате CSV и скачаны.	Да

МЕТРИКИ

$$\textit{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\textit{Precision} = \frac{TP}{TP + FP}$$

$$\textit{Recall} = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 * \textit{Precision} * \textit{Recall}}{\textit{Precision} + \textit{Recall}}$$