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

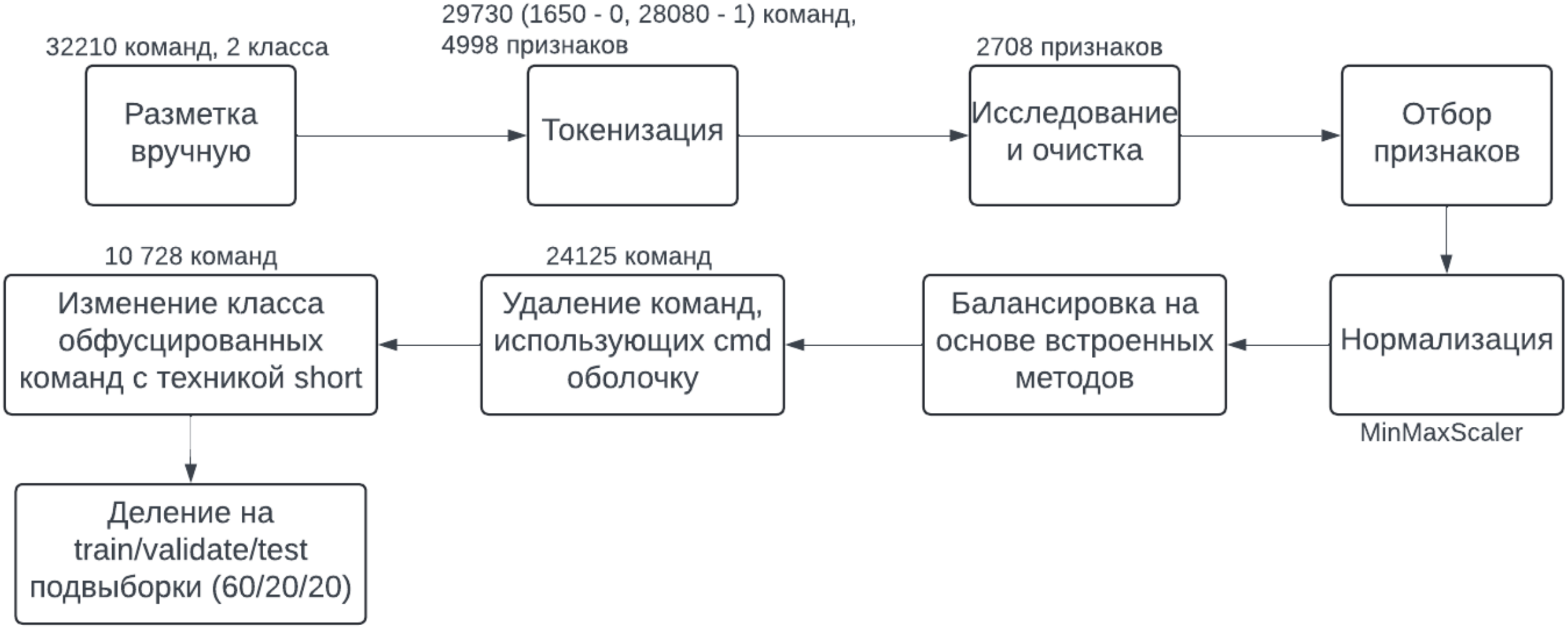
Название	Авторы	Набор данных	Алгоритм	Точность
PowerDP: De-Obfuscating and Profiling Malicious PowerShell Commands With Multi-Label Classifiers	Meng-Han Tsai, Chia-Ching Lin, Zheng-Gang He, Wei-Chieh Yang, Chin-Laung Lei	Собран вручную из данных песочницы и открытых источников	Random Forest, Deep Neural 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:** Jyputer Notebook
- **Библиотеки для машинного обучения**: scikit-learn, xgboost, catboost, lightgbm, pandas, numpy, matplotlib, seaborn

https://github.com/MaksimSkoblyuk/Obfuscation-Detecting

НАБОР ДАННЫХ



https://github.com/MaksimSkoblyuk/Obfuscation-Detecting

АЛГОРИТМЫ МАШИННОГО ОБУЧЕНИЯ

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

https://github.com/MaksimSkoblyuk/Obfuscation-Detecting

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

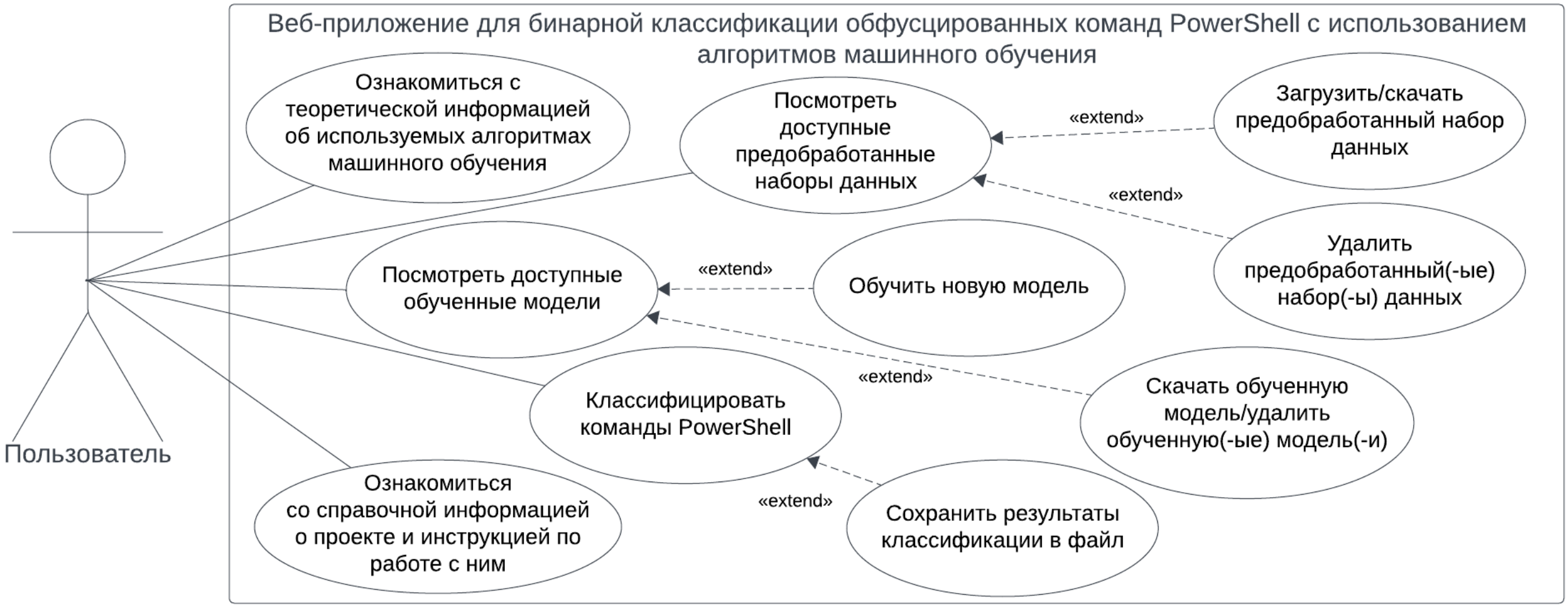
https://github.com/MaksimSkoblyuk/Obfuscation-Detecting

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

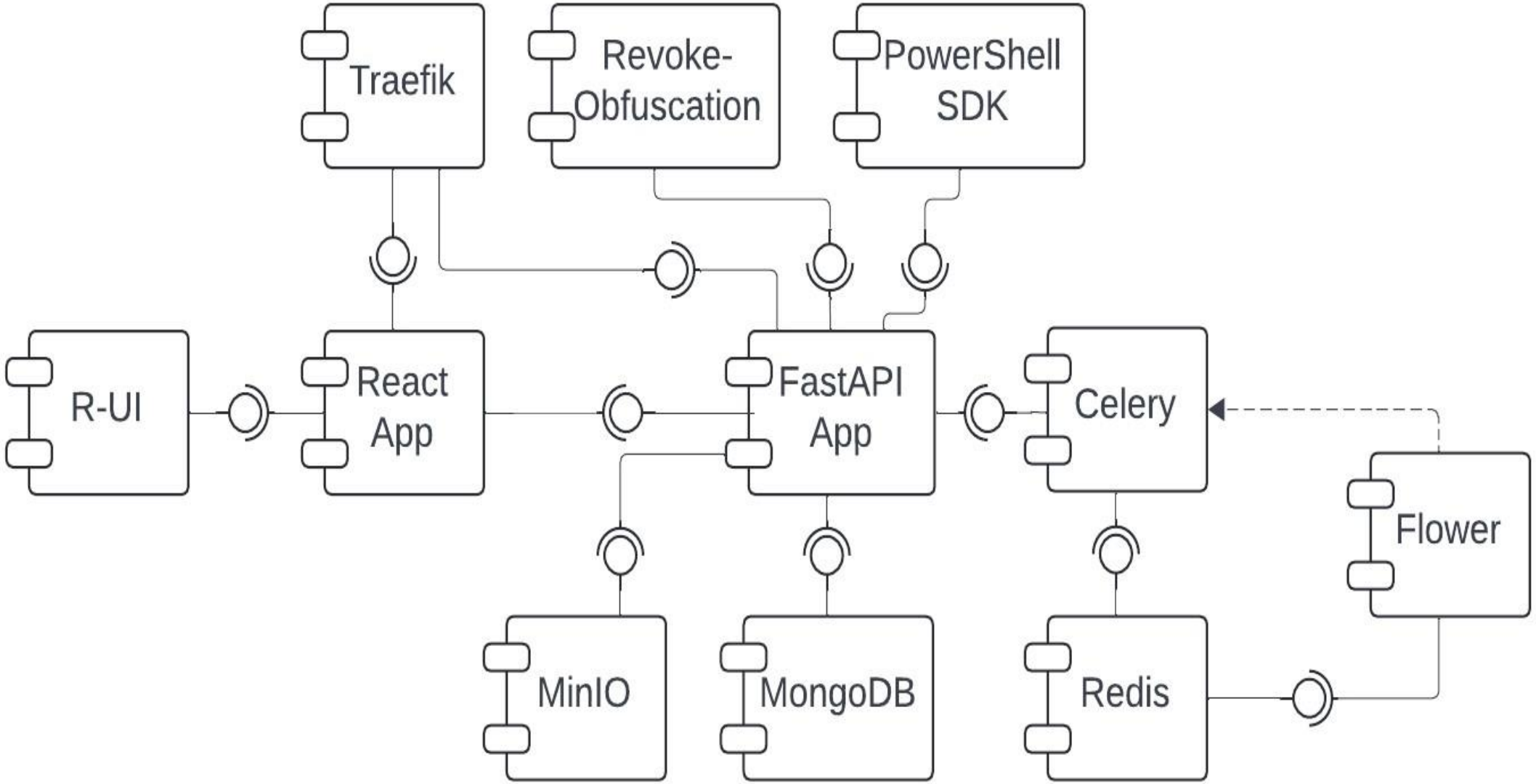
https://github.com/MaksimSkoblyuk/Obfuscation-Detecting

ДИАГРАММА ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ



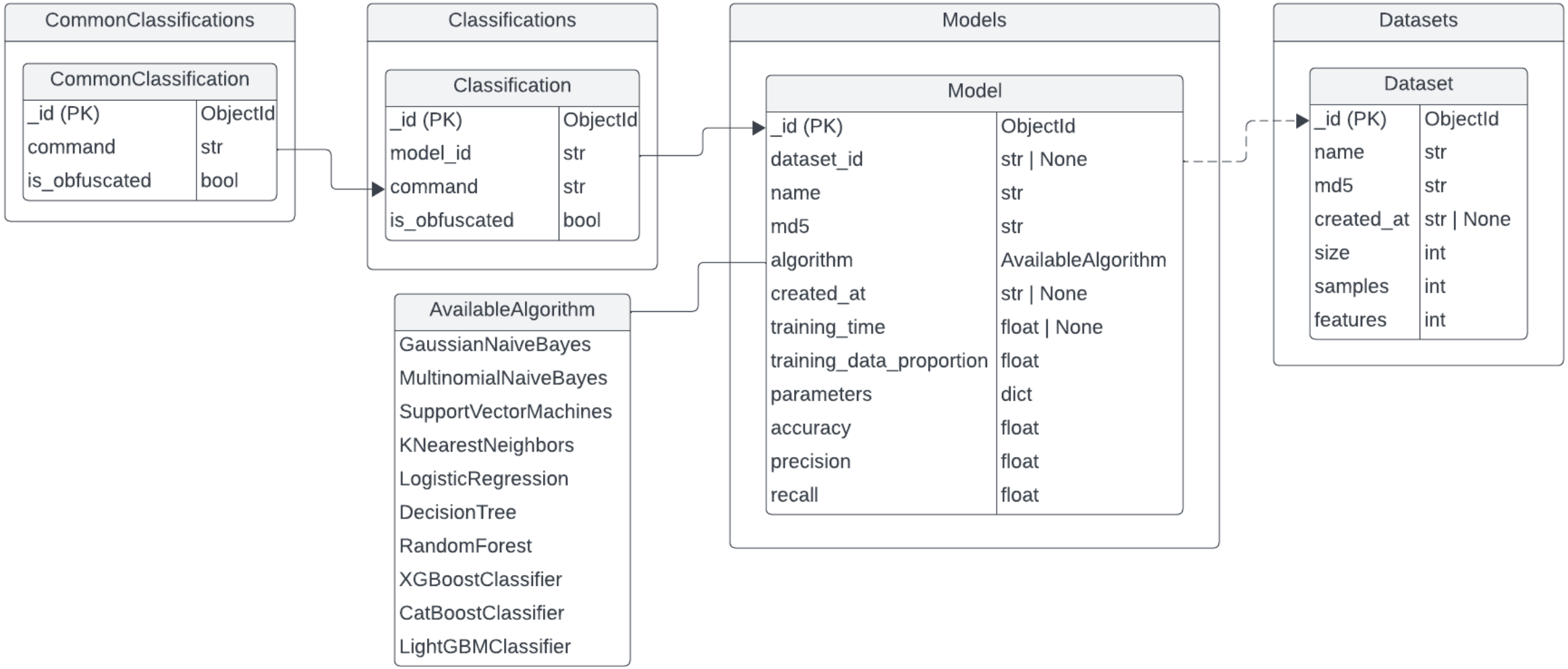
https://github.com/MaksimSkoblyuk/Obfuscation-Detecting

АРХИТЕКТУРА ВЕБ-ПРИЛОЖЕНИЯ



https://github.com/MaksimSkoblyuk/Obfuscation-Detecting

СХЕМА БАЗЫ ДАННЫХ



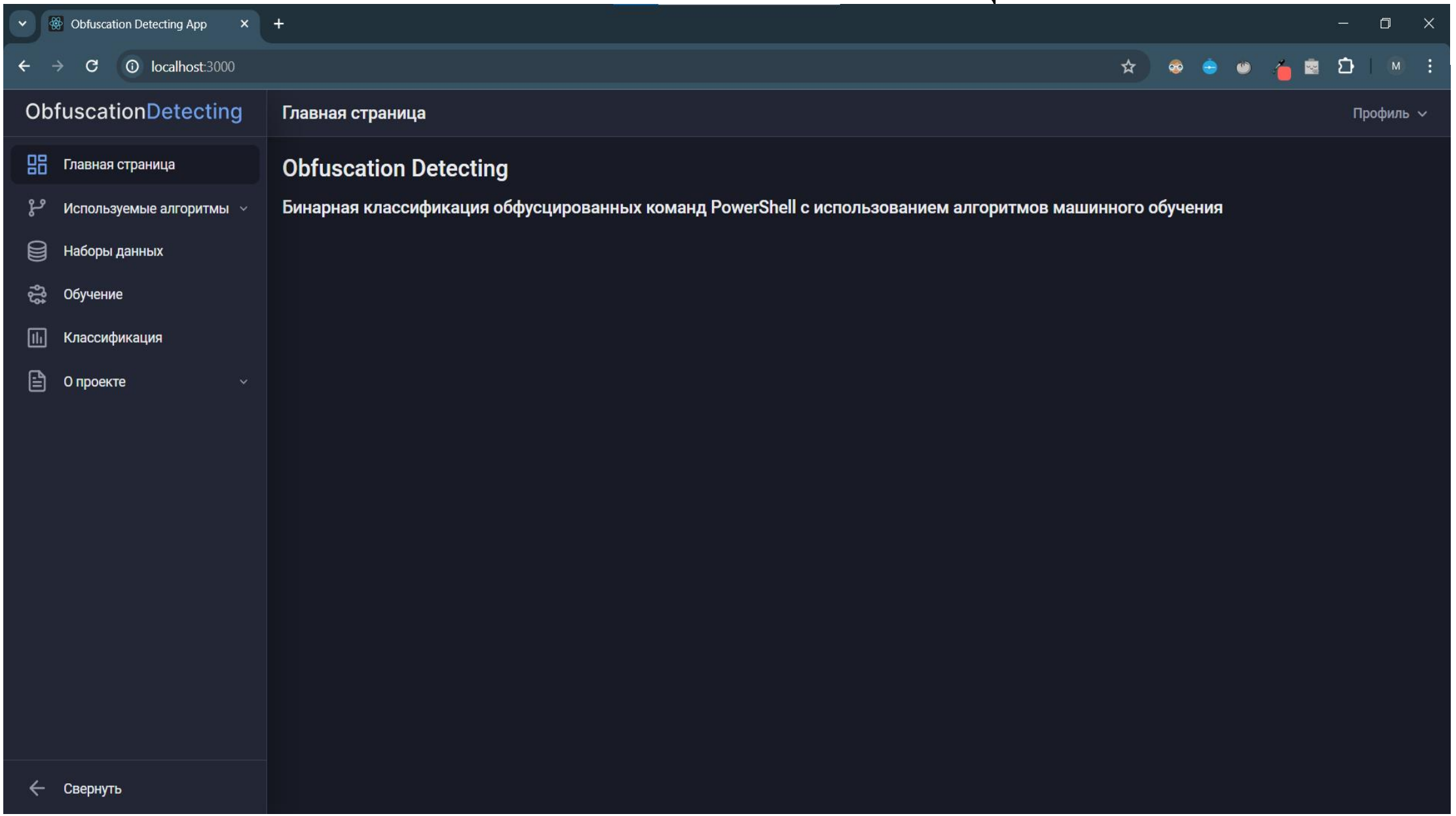
https://github.com/MaksimSkoblyuk/Obfuscation-Detecting

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

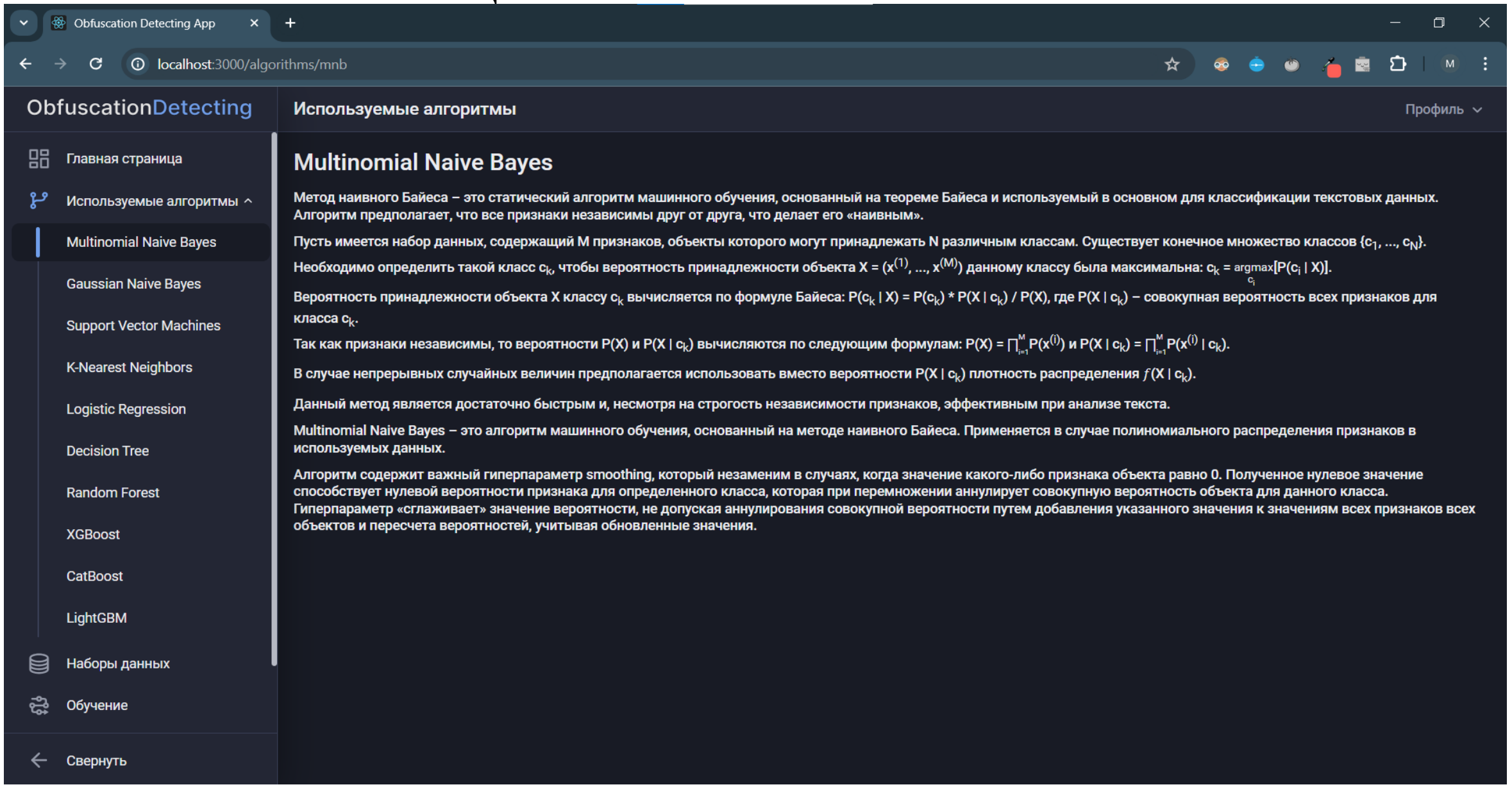
https://github.com/MaksimSkoblyuk/Obfuscation-Detecting

ГЛАВНАЯ СТРАНИЦА



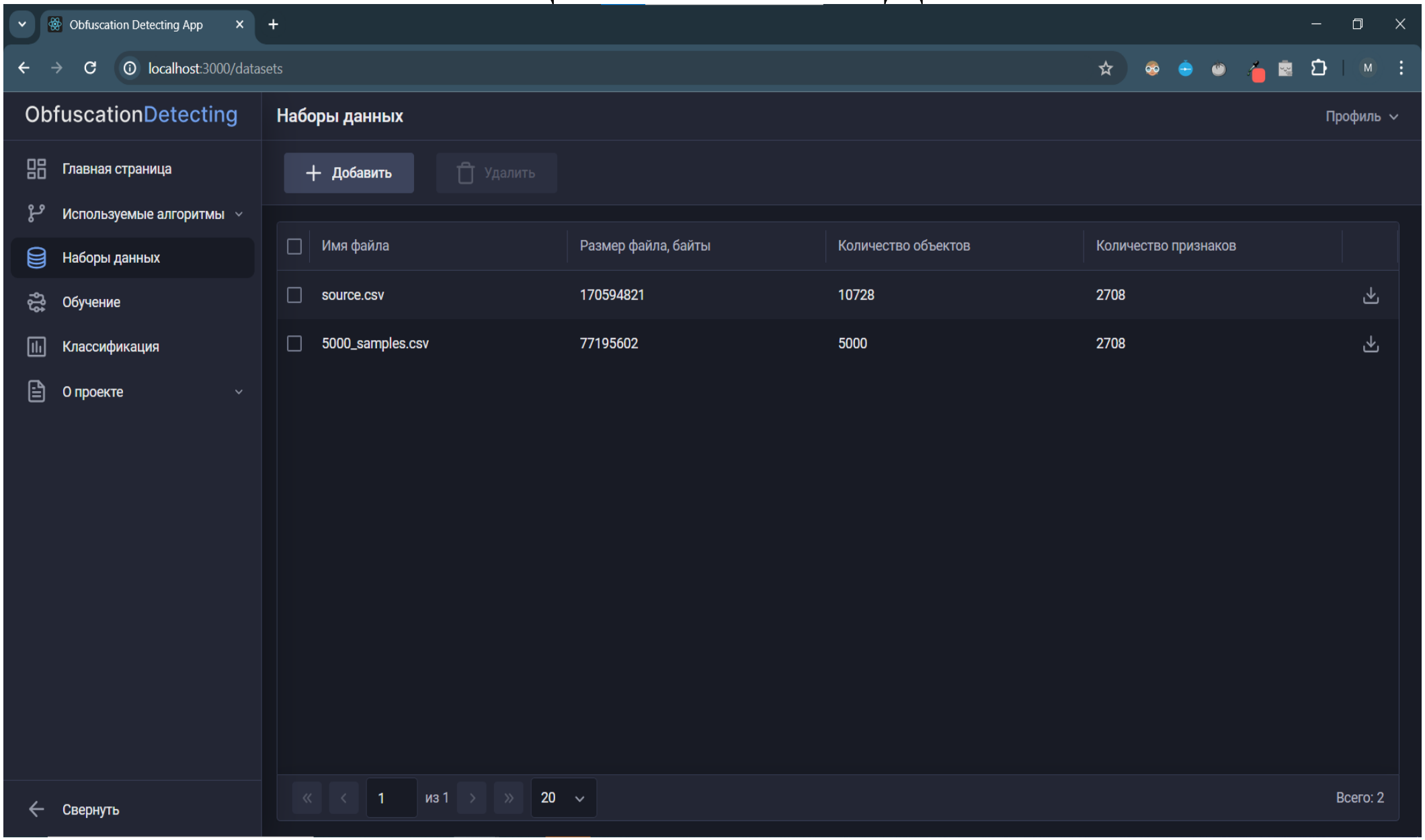
https://github.com/MaksimSkoblyuk/Obfuscation-Detecting

СТРАНИЦА ИСПОЛЬЗУМЫЕ АЛГОРИТМЫ



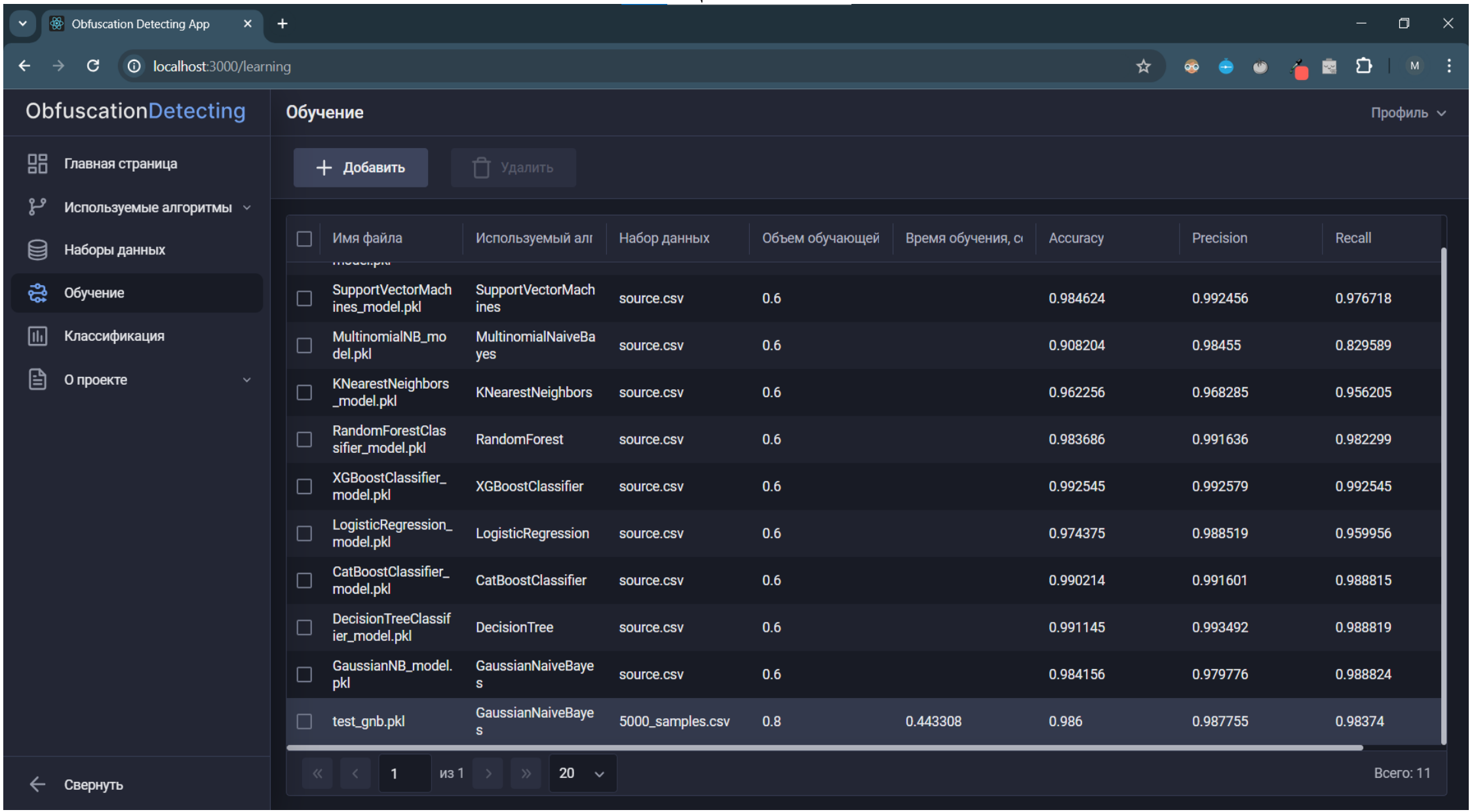
https://github.com/MaksimSkoblyuk/Obfuscation-Detecting

СТРАНИЦА НАБОРЫ ДАННЫХ



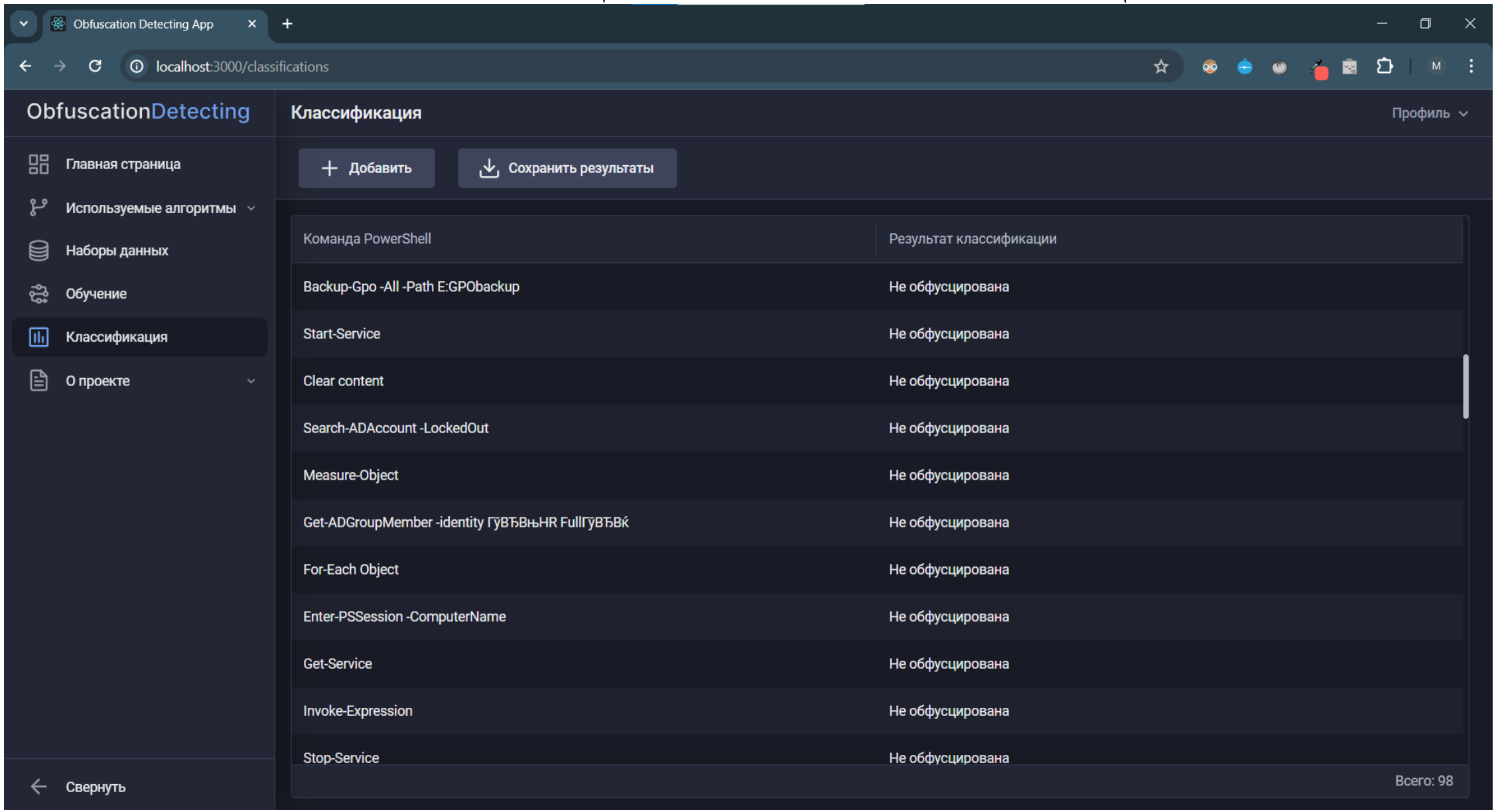
https://github.com/MaksimSkoblyuk/Obfuscation-Detecting

СТРАНИЦА ОБУЧЕНИЕ



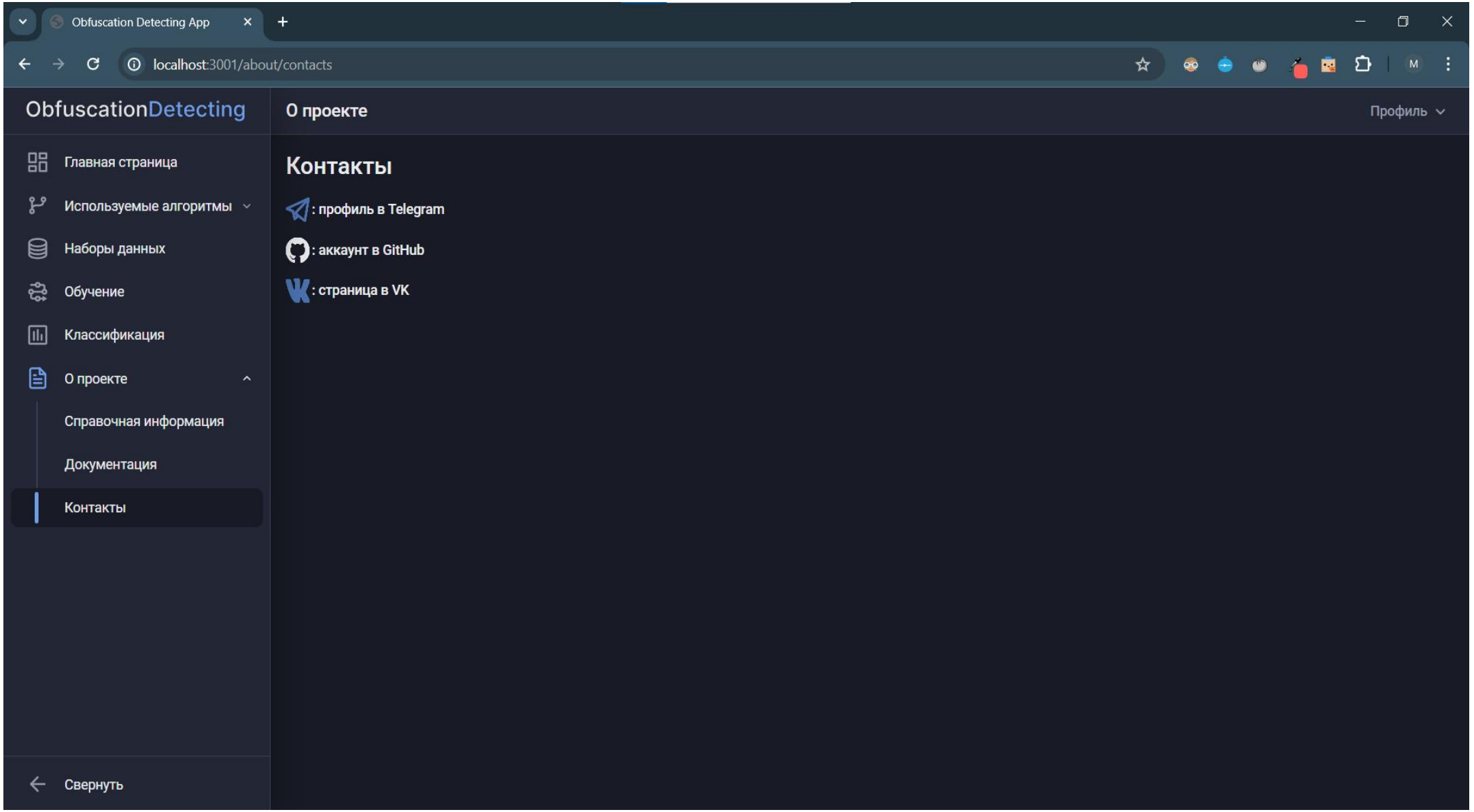
https://github.com/MaksimSkoblyuk/Obfuscation-Detecting

СТРАНИЦА КЛАССИФИКАЦИЯ

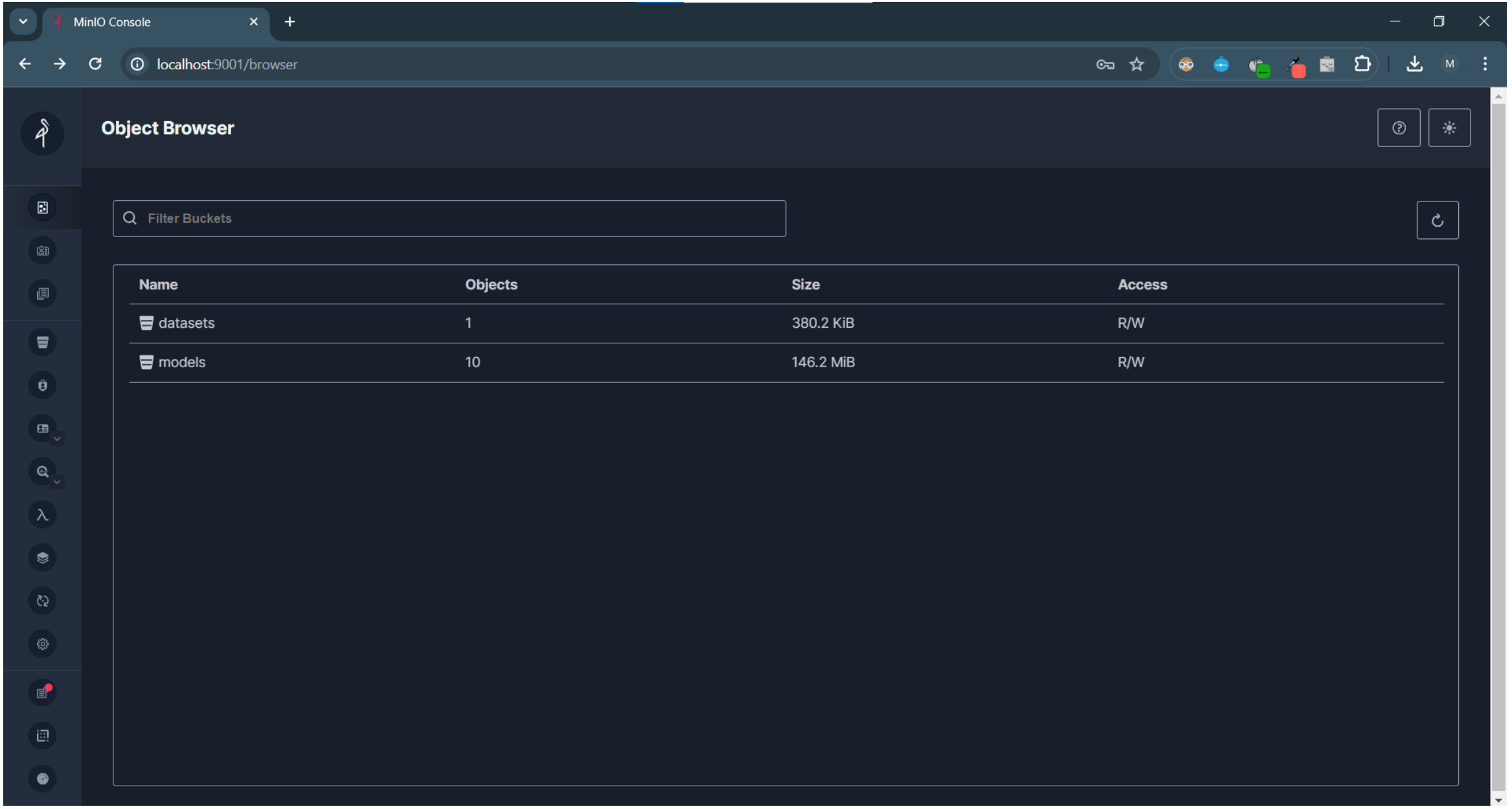


https://github.com/MaksimSkoblyuk/Obfuscation-Detecting

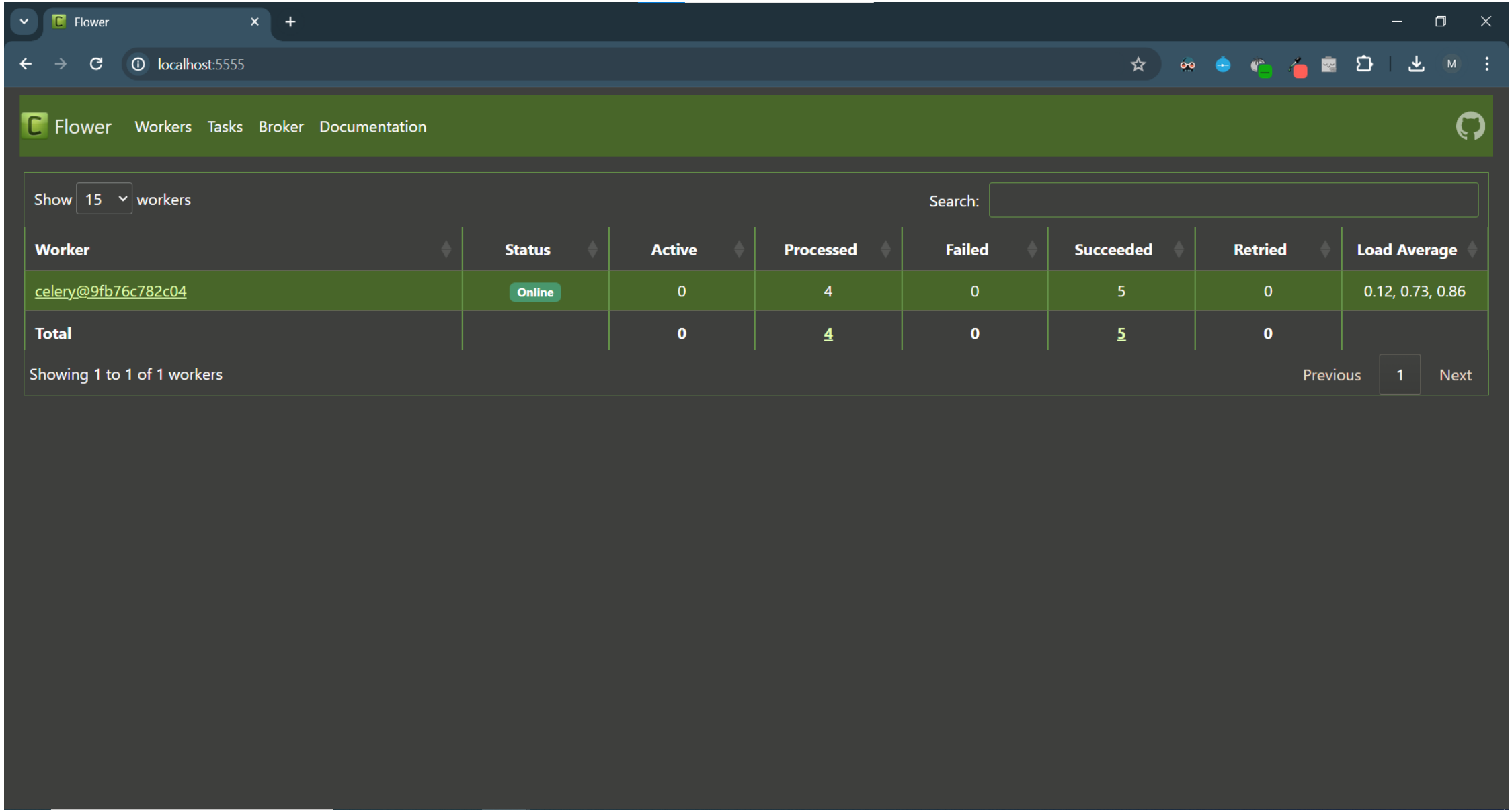
СТРАНИЦА О ПРОЕКТЕ



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


Flower
localhost:5555
Flower
Workers
Tasks
Broker
Documentation
Show 15 workers
Search:
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

https://github.com/MaksimSkoblyuk/Obfuscation-Detecting

АКТ О ВНЕДРЕНИИ

АКТ
о внедрении научно-технической продукции

Данный акт удостоверяет, что в ООО «Р-Вижн» внедрено в опытную эксплуатацию приложение для бинарной классификации обфусцированных команд PowerShell с использованием алгоритмов машинного обучения, разработанный студентом группы КЭ-301 Скоблюком Максимом Игоревичем, научный руководитель – старший преподаватель кафедры системного программирования ФГАОУ ВО «ЮУрГУ (НИУ)» Никольская Ксения Юрьевна.

Приложение для бинарной классификации обфусцированных команд PowerShell с использованием алгоритмов машинного обучения используется в коммерческих целях.

Акт подписал
Генеральный директор
_____ Бондаренко А.В.
31.05.2023 г.
МП

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	powershell -NoExi "$os = Get-WmiObject win32_operatingsystem $uptime = (Get-Date) - $os.ConvertToDateTime($os.LastBootUpTime) Write-Output (\"Last boot: \" + $os.ConvertToDateTime($os.LastBootUpTime))"	$os = Get-WmiObject win32_operatingsystem $uptime = (Get-Date) - $os.ConvertToDateTime($os.LastBootUpTime) Write-Output ("Last boot: " + $os.ConvertToDateTime($os.LastBootUpTime))
variables – манипуляции с переменными среды	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%"	$os = Get-WmiObject win32_operatingsystem $uptime = (Get-Date) - $os.ConvertToDateTime($os.LastBootUpTime) Write-Output ("Last boot: " + $os.ConvertToDateTime($os.LastBootUpTime))
encoding – изменение кодировки	Invoke-Expression (('Add-ADG'+'r'+'oupMembe'+'r'+' -Ide'+'ntit'+'y group'+'-name -Me'+'mbers'+' S'+'se'+'r1, us'+'e'+'r2'))	Add-ADGroupMember -Identity group-name -Members Sser1, user2
string – манипуляции со строками	$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	$members = Import-CSV c:itadd-to-group.csv | Select-Object -ExpandProperty samaccountname Add-ADGroupMember -Identity hr-n-drive-rw -Members $members
symbol – использование игнорируемых символов	Add-AD`Grou`pMe`mber -Identity group-name -Members Sser1, user2	Add-ADGroupMember -Identity group-name -Members Sser1, user2

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	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	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	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	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	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	1. Выбрать в боковой панели вкладку «Обучение». 2. Нажать на кнопку скачивания в строке таблицы у обученной модели.	Скачан файл в формате pkl с сериализованной обученной моделью.	Да

ФУНКЦИОНАЛЬНОЕ ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ (3)

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
7	Удаление обученных моделей	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	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	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	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

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

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

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

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