

Рабочий лист №1

Дата "1" февраля 2025 г.  
(заполняется оргкомитетом)

Шифр ЛИ-39  
(заполняется оргкомитетом)

Оценка работы

(таблица заполняется по итогам проверки работы членами жюри олимпиады)

| № задания | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 | Итого<br>(итоговый балл,<br>подпись<br>председателя<br>жюри) |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--|
| Балл      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 85   |
| № задания | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |  |
| Балл      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | ЛБ   |

Магистрия Ум

(название олимпиады, заполняется участником)

Прикладная информатика

(профиль олимпиады, заполняется участником)

Задание 3. Предположим, что сервис обработки телефонных звонков приводит буквенное произношение чисел в цифровую форму, например: "двадцать один" → "21". Кроме того, сервис обработки телефонных звонков «склеивает» рядом стоящие числа, т.е. убирает пробелы между ними, например: "слово 21 46 слово" → "слово 2146 слово".

За паттерн серии и номера паспорта возьмём десять идущих подряд цифр.

За паттерн номера СНИЛС возьмём одиннадцать идущих подряд цифр.

За паттерн номера банковской карты возьмём шестнадцать идущих подряд цифр.

За паттерн CVC/CVV-кода возьмём три идущих подряд цифры.

В данном задании ограничимся этими четырьмя паттернами.

Сдано листов: 2

Python-kod:

```
import re

def extract_passport(text):
    passport_pattern = r'\b\s\d{10}\s\b'
    return re.findall(passport_pattern, text)

def extract_snils(text):
    snils_passport_pattern = r'\b\s\d{11}\s\b'
    return re.findall(snils_pattern, text)

def extract_card_number(text):
    card_number_pattern = r'\b\s\d{16}\s\b'
    return re.findall(card_number_pattern, text)

def extract_cvc(text):
    cvc_pattern = r'\b\s\d{3}\s\b'
    return re.findall(cvc_pattern, text)

def extract_personal_data(text):
    personal_data = {
        'passport': extract_passport(text),
        'snils': extract_snils(text),
        'bank_card': extract_card_number(text),
        'cvc': extract_cvc(text)
    }
    return personal_data
```

Данный код позволит получить словарь с данными о человеке, если будут выполняться условия предобработки, о которых я упомянул. Хочу обратить внимание, что если в расшифровке данные, которые мы хотим извлечь, будут находиться в конце строки, то не выполнится условие о пробеле после числа. Либо мошенники должны продолжить разговор, либо жертва что-нибудь сказала, либо в обработчике ~~то~~ телефонных звонков можно добавить " \*конец разговора\* ".

Данный код не обработает, например, такой случай: 'номер 1234 серия 567899'. Можно отдельную обработку сделать для пар соседних чисел в расшифровке.

Дополнительный рабочий лист  
(без рабочего листа №1 недействителен)

Дата "1" февраля 2025 г.  
(заполняется участником)

Шифр ПИ-39  
(заполняется участником)

Задание 2.

1. Загрузить модель для классификации звонков:  
input classification
2. Загрузить расшифровку телефонного звонка:  
Select телер.звонки from ... as text
3. Предобработка расшифровки телефонного звонка:  
text, слова\_в\_числа # 'два' → '2'  
text, убрать\_пробелы\_между\_числами  
text, добавить\_факончание\_разговора # '...21' → '...21 \*Stop\*'  
text.toLowerCase()
4. Обработка расшифровки:  
for word in text:  
if word in dict\_keywords.keys:  
max\_momentaryness += dict\_keywords[word]
5. Принятие решения:  
if max\_momentaryness ≥ classification.max\_threshold:  
return большой max моментальности  
elif max\_momen. ≥ classification.mid\_threshold and  
max\_momen. < classification.max\_threshold:  
return средний max моментальности  
else:  
return низкий max моментальности

Я представил упрощённый алгоритм для выявления применения моментальности, посредством сравнения «баллов max моментальности» с заранее заданными порогами.

dict\_keywords является словарём, который в ключах хранит позитивные слова, а в значениях — балл max моментальности. Установлением таких связей займётся аналитик.

Например: 'предоставьте': 5. Я вижу  
всего два порога для упрощения, можно  
придумать и более изюминки методы.

### Задача 1. Проектирование архитектуры.

Я вижу следующие элементы архитектуры:

- База данных для хранения расшифровок телефонных разговоров. В ней будут храниться как обработанные расшифровки, так и расшифровки без обработки (конечно, в отдельных таблицах).
- База данных для хранения данных о классификации. В ней могут храниться те самые словари с «подозрительными» словами и баллами возможного мошенничества.
- Сервис для <sup>взаимодействий с внешними сервисами</sup> ~~принятия внешних данных~~. Например, этот сервис может принимать партию расшифровок телефонных разговоров из внешнего источника. Этот сервис отправит такие данные в БД для хранения расшифровок.
- Сервис обработки расшифровок телефонных звонков. Здесь происходит предобработка и обработка расшифровок. Взаимодействует с обеими БД.

Можно реализовать микросервисную архитектуру, где сервисы будут взаимодействовать по API.

Можно также представить и другие сервисы и БД, но для решения задачи этих достаточно. Например, сервис визуализации данных поможет командам принимать решения о противодействии мошенничеству.

В БД с расшифровками хранятся также и обработанные расшифровки с вероятностью мошенничества, которые сервис для взаимодействия с внешними сервисами может предоставлять.

БД <sup>с расшифр.</sup> ~~может быть~~ будет реляционной, а БД для классификации лучше сделать нереляционной: key-value, для сохранения памяти и скорости запросов.