ПРЕЗЕНТАЦИЯ ПРОДУКТА

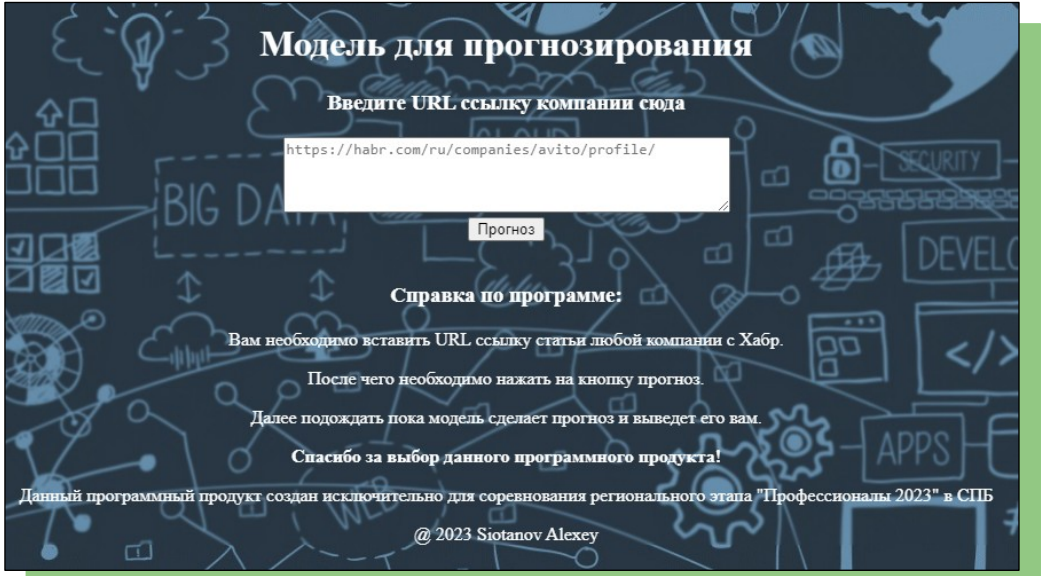
ГОТОВОЕ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ СОРЕВНОВАНИЙ

2023 ПРОФЕССИОНАЛЫ

ВСЕРОССИЙСКОЕ
ЧЕМПИОНАТНОЕ
ДВИЖЕНИЕ
ПО ПРОФЕССИОНАЛЬНОМУ
МАСТЕРСТВУ

Выполнил конкурсант: Сиотанов Алексей
Рабочее место №2

ПРИМЕР РАБОТОСПОСОБНОСТИ



Примечание: *Присутствует справка для пользователя*

МЕТОДЫ ДОСТИЖЕНИЯ: ПАРСЕР

Правильный **парсинг** является базовым и важным действием при сборе данных

```python
# Заготовка полного парсера с перечнем сайтов
url_list = ['https://habr.com/ru/companies/avito/profile/', 'https://habr.com/ru/companies/vtb/profile/',
        'https://habr.com/ru/companies/vk/profile/', 'https://habr.com/ru/companies/vk/profile/']
i = 0
while i != len(url_list):
    site_url = url_list[i]
    links = [site_url]

    constructor = {
        'Название': [],
        'Рейтинг': [],
        'Описание': [],
        'Сфера деятельности': [],
        'Дата публрикации': []
    }
    for link in links:
        response = requests.get(link)
        soup = Soup(response.content, 'html.parser')
        items = soup.find_all('div', class_='pull-down')
        for item in items:
            title = item.find('a', class_='tm-company-card__name').text.strip()
            reiting = item.find('span', class_='tm-votes-lever__score-counter').text.strip()
            info = item.find('span', class_='tm-company-profile__content').text.strip()
            sfera = item.find('div', class_='tm-company-profile__categories').text.strip()
            datepub = item.find('dd', class_='tm-description-list__body tm-description-list__body tm-description-list__body_
            constructor['Название'] += [title]
            constructor['Рейтинг'] += [reiting]
            constructor['Описание'] += [info]
            constructor['Сфера деятельности'] += [sfera]
            constructor['Дата публрикации'] += [datepub]
        pd.DataFrame(constructor).to_csv('./DataFrame2.csv', encoding="utf-8", index=False)
        i += 1
```


Примечание: *Код написан по стандарту PEP 8*

МЕТОДЫ ДОСТИЖЕНИЯ: ОБРАБОТКА ДАННЫХ

Токенизация, лемматизация, выделение значимых частей речи и удаление стоп-слов.

МЕТОДЫ ДОСТИЖЕНИЯ: ОБУЧЕНИЕ

Модель классификации текстов **fastText**

```
train, test = train_test_split(BaseDF, test_size = 0.2)
```

```
# Создадим текстовые файла для обучения модели с сферой и названием
with open('train.txt', 'w') as f:
    for each_text, each_label in zip(train['Название'], train['Сфера']):
        f.writelines(f'__label__{each_label} {each_text}\n')

with open('test.txt', 'w') as f:
    for each_text, each_label in zip(test['Название'], test['Сфера']):
        f.writelines(f'__label__{each_label} {each_text}\n')
```

```
# Первая модель без оптимизации гиперпараметров
model1 = fasttext.train_supervised('train.txt')
```


Примечание: *Выбор разделения данных на обучающую и тестирующую стал в соотношении 80:20 исходя из закона Парето*

МЕТОДЫ ДОСТИЖЕНИЯ: ОПТИМИЗАЦИЯ

Поиск наилучших гиперпараметров вручную может занять много времени

```
# Вторая модель с количеством эпох равной 25
model2 = fasttext.train_supervised('train.txt', epoch=25)
```

```
# Третья модель с количеством эпох равной 10 и скоростью обучния равной 1
model3 = fasttext.train_supervised('train.txt', epoch=10, lr=1.0)
```

```
# Четвёртая модель с аргументов wordNgrams = 2
# Используя биграммы слов, а не просто юниграммы
model4 = fasttext.train_supervised('train.txt', epoch=10, lr=1.0, wordNgrams =2)
```

```
# Позволяет оптимизировать гиперпараметры для получения наивысшего показателя
model5 = fasttext.train_supervised('train.txt', autotuneValidationFile='test.txt')
```

ПЕРСПЕКТИВЫ УЛУЧШЕНИЯ

1. Необходимо больше данных для улучшения качества обучения.

2. Произвести обучение другими различными способами, что обеспечит больший выбор качественной модели.

3. Произвести внедрение продукта другим способом, с использованием улучшенного интерфейса для пользователя.

Примечание: *Спасибо за внимание!*