

Программные средства сбора, консолидации и аналитики данных

Смоляков Руслан Игоревич

Тема: «Практическая работа 1. Начало работы с API»

Цель работы: получить углубленные навыки программного сбора и анализа данных из различных веб-источников с помощью API.

Вариант - 21

Ссылка на репозиторий:

https://github.com/Ruslanishka/soft-tools/tree/main/theme_1/practice_1

Задача 1: Поиск академических датасетов

Подготовка:

```
[10] # Установка необходимых библиотек
✓ !pip install -q kaggle requests pandas matplotlib seaborn PyGithub
```

Задание 1. Kaggle API: Поиск академических датасетов

```
[11] # Аутентификация Kaggle (в Colab)
✓ 24 сек.
from google.colab import files
import os

# Загрузка файла kaggle.json
uploaded = files.upload()

# Настройка токена
os.makedirs(os.path.expanduser("~/kaggle"), exist_ok=True)
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

Выбрать файлы | Файл не выбран | Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving kaggle.json to kaggle (1).json

Анализ и визуализация:

```
# Поиск датасетов, опубликованных университетами

from kaggle.api.kaggle_api_extended import KaggleApi
import pandas as pd

api = KaggleApi()
api.authenticate()

universities = ["Stanford University", "MIT", "Harvard University", "University of Oxford"]
all_datasets = []

for university in universities:
    search_results = api.dataset_list(search=university, file_type='csv')
    for dataset in search_results:
        ds_dict = dataset.to_dict()
        ds_dict['source_university'] = university # Добавляем информацию об университете
        all_datasets.append(ds_dict)

df_datasets = pd.DataFrame(all_datasets)

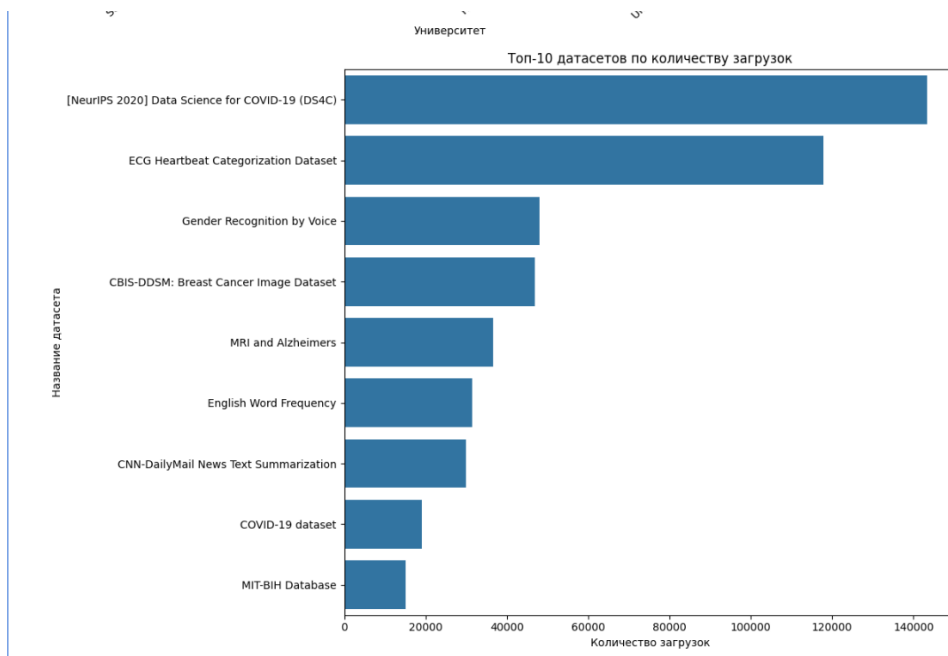
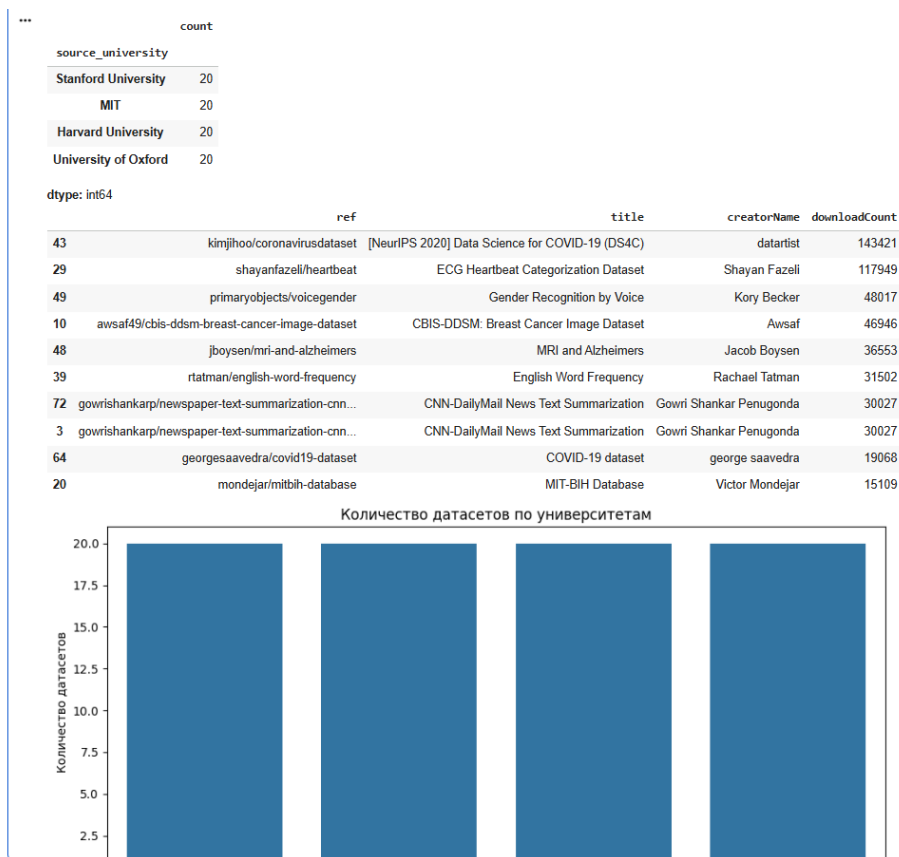
# Анализ и визуализация
# 1. Количество датасетов от каждого университета
university_counts = df_datasets['source_university'].value_counts()
display(university_counts)

# 2. Популярность датасетов (например, по количеству загрузок)
# Выберем топ-10 датасетов по количеству загрузок
top_downloads = df_datasets.sort_values(by='downloadCount', ascending=False).head(10)
display(top_downloads[['ref', 'title', 'creatorName', 'downloadCount']])

# 3. Визуализация количества датасетов по университетам
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.barplot(x=university_counts.index, y=university_counts.values)
plt.title('Количество датасетов по университетам')
plt.xlabel('Университет')
plt.ylabel('Количество датасетов')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

# 4. Визуализация топ-10 датасетов по загрузкам
plt.figure(figsize=(12, 8))
sns.barplot(x='downloadCount', y='title', data=top_downloads, orient='h')
plt.title('Топ-10 датасетов по количеству загрузок')
plt.xlabel('Количество загрузок')
plt.ylabel('Название датасета')
plt.tight_layout()
plt.show()
```



Эти выводы могут быть полезны для понимания активности различных университетов на Kaggle и выявления наиболее востребованных академических датасетов.

Задача 2: Анализ миграции проектов: найти проекты, которые недавно перешли с Python 2 на Python 3 (анализ коммитов).

Подготовка:

```
3]
# Аутентификация в GitHub API

import requests
from datetime import datetime, timedelta

# ВАЖНО: Замените 'ВАШ_TOKEN' на свой персональный токен GitHub
ACCESS_TOKEN = "ghp_M1VnAfYjKBOxu3A2p3Q0peuA7JcVlh156mzd"
GITHUB_API_URL = "https://api.github.com"

headers = {
    "Authorization": f"token {ACCESS_TOKEN}",
    "Accept": "application/vnd.github.v3+json"
}
```

Получение данных:

```
# Функция для получения коммитов проекта
def get_project_commits(repo_name):
    commits_url = f"{GITHUB_API_URL}/repos/{repo_name}/commits"
    all_commits = []
    page = 1
    while True:
        response = requests.get(commits_url, headers=headers, params={"page": page, "per_page": 100})
        if response.status_code != 200:
            print(f"Error fetching commits for {repo_name}: {response.status_code}")
            break
        commits = response.json()
        if not commits:
            break
        all_commits.extend(commits)
        page += 1
    return all_commits

# Список проектов для анализа (замените на свои проекты)
# Например: ["tensorflow/tensorflow", "django/django"]
projects_to_analyze = ["psf/requests", "pallets/flask", "plotly/plotly.py"] # Замените на список интересующих вас проектов

project_commits = {}
for project in projects_to_analyze:
    print(f"Fetching commits for {project}...")
    project_commits[project] = get_project_commits(project)

print("Finished fetching commits.")

Fetching commits for psf/requests...
Fetching commits for pallets/flask...
Fetching commits for plotly/plotly.py...
Finished fetching commits.
```

```
# Преобразование данных о коммитах в DataFrame
import pandas as pd

all_commits_list = []
for repo, commits in project_commits.items():
    for commit in commits:
        all_commits_list.append({
            'repo': repo,
            'sha': commit['sha'],
            'message': commit['commit']['message'],
            'date': commit['commit']['committer']['date']
        })

commits_df = pd.DataFrame(all_commits_list)

# Преобразование строки с датой в формат datetime
commits_df['date'] = pd.to_datetime(commits_df['date'])

# Анализ сообщений коммитов на наличие ключевых слов, связанных с миграцией
migration_keywords = ["python3", "python 3", "py3", "migrate to python 3", "python 2 to 3"]
commits_df['is_migration_commit'] = commits_df['message'].str.contains(''.join(migration_keywords), case=False, na=False)

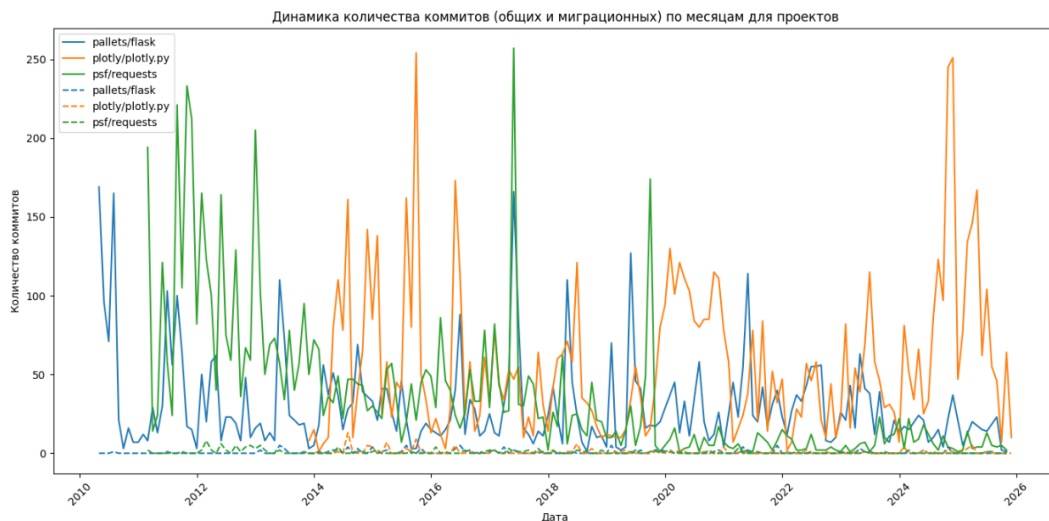
display(commits_df.head())
```

Анализ:

```
1
2
3 # Анализ динамики коммитов по проектам
4
5 # Группировка по проекту и дате (по месяцам) для подсчета коммитов
6 monthly_commits = commits_df.groupby(['repo', pd.Grouper(key='date', freq='ME')]).size().reset_index(name='commit_count')
7
8 # Группировка по проекту и дате (по месяцам) для подсчета коммитов, связанных с миграцией
9 monthly_migration_commits = commits_df[commits_df['is_migration_commit']].groupby(['repo', pd.Grouper(key='date', freq='ME')]).size().reset_index(name='migration_commit_count')
10
11 # Объединение данных о всех коммитах и коммитах миграции
12 monthly_analysis = monthly_commits.merge(monthly_migration_commits, on=['repo', 'date'], how='left').fillna(0)
13
14 display(monthly_analysis.head())
```

Визуализация:

```
1
2
3 # Визуализация динамики коммитов и коммитов миграции
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6
7 plt.figure(figsize=(14, 7))
8 sns.lineplot(data=monthly_analysis, x='date', y='commit_count', hue='repo')
9 sns.lineplot(data=monthly_analysis, x='date', y='migration_commit_count', hue='repo', linestyle='--')
10
11 plt.title('Динамика количества коммитов (общих и миграционных) по месяцам для проектов')
12 plt.xlabel('Дата')
13 plt.ylabel('Количество коммитов')
14 plt.xticks(rotation=45)
15 plt.legend()
16 plt.tight_layout()
17 plt.show()
```



Вывод: на основе этого анализа и визуализации можно попытаться выявить изменения в активности коммитов, которые могут быть связаны с миграцией на Python 3. Резкое увеличение или изменение характера коммитов в определенный период может указывать на активную работу над миграцией.

Задача 3: Поиск вакансий с релокацией: найти вакансии в сфере IT в Москве, которые предлагают помощь с переездом.

К API hh не надо подключаться, поэтому сразу получаем данные

```
import time
from collections import Counter
import requests # Импортируем requests, если он еще не был импортирован

HH_API_URL = "https://api.hh.ru/vacancies"
all_vacancies = []
pages_to_load = 5 # 100 вакансий (по 20 на странице)

params = {
    'text': 'IT', # Ищем вакансии в сфере IT
    'area': 1, # Москва
    'relocation': 'true', # Ищем вакансии с помощью переезда
    'per_page': 20
}

for page in range(pages_to_load):
    params['page'] = page
    response = requests.get(HH_API_URL, params=params)
    if response.status_code == 200:
        vacancies_data = response.json()
        all_vacancies.extend(vacancies_data['items'])
        print(f"Загружена страница {page + 1}/{pages_to_load}")
    else:
        print(f"Ошибка при загрузке страницы {page + 1}: {response.status_code}")
        break
    time.sleep(0.2) # Соблюдаем вежливость к API

print(f"\nВсего найдено вакансий: {len(all_vacancies)}")
```

*** Загружена страница 1/5
Загружена страница 2/5
Загружена страница 3/5
Загружена страница 4/5
Загружена страница 5/5

Всего найдено вакансий: 100

Далее проанализировал какие ключевые навыки и какой опыт требуется

```
all_skills = []
all_experience = []

for vacancy in all_vacancies:
    # Получаем детальную информацию о вакансии, т.к. ключевые навыки могут быть не в кратком описании
    vacancy_url = vacancy['url']
    vacancy_details_response = requests.get(vacancy_url)
    if vacancy_details_response.status_code == 200:
        vacancy_details = vacancy_details_response.json()
        if 'key_skills' in vacancy_details:
            for skill in vacancy_details['key_skills']:
                all_skills.append(skill['name'].lower()) # Приводим к нижнему регистру для стандартизации

        # Извлекаем опыт работы
        if 'experience' in vacancy_details and vacancy_details['experience'] is not None:
            all_experience.append(vacancy_details['experience']['name'])

    time.sleep(0.2)

# Считаем частоту встречаемости каждого навыка
skill_counts = Counter(all_skills)

# Выводим топ-10 самых популярных навыков
top_10_skills = skill_counts.most_common(10)
print("Топ-10 ключевых навыков для работы с переездом:")
for skill, count in top_10_skills:
    print(f"- {skill}: {count} раз")

# Считаем частоту встречаемости каждого уровня опыта
experience_counts = Counter(all_experience)

# Выводим частоту встречаемости каждого уровня опыта
print("\nТребуемый опыт работы:")
for experience, count in experience_counts.most_common():
    print(f"- {experience}: {count} раз")
```

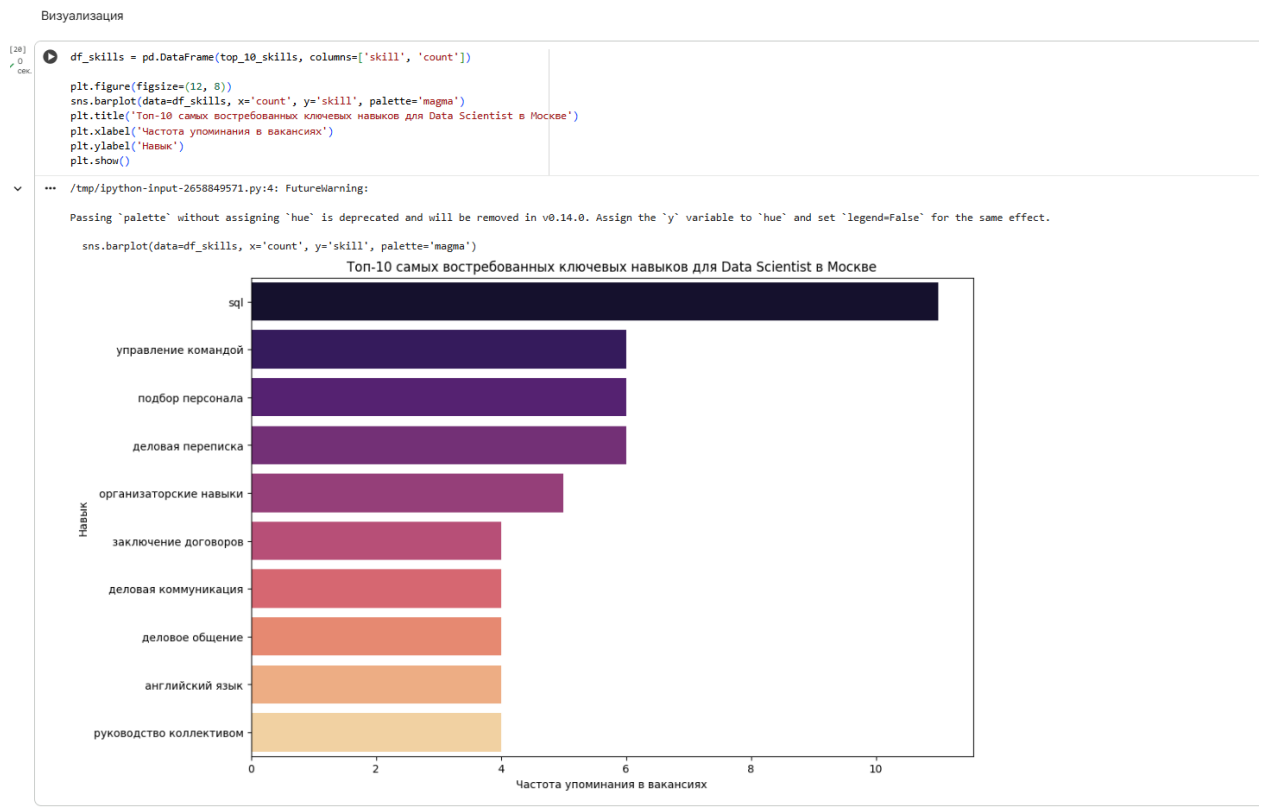
Топ-10 ключевых навыков для работы с переездом:

- sql: 11 раз
- управление командой: 6 раз
- подбор персонала: 6 раз
- деловая переписка: 6 раз
- организаторские навыки: 5 раз
- заключение договоров: 4 раз
- деловая коммуникация: 4 раз
- деловое общение: 4 раз
- английский язык: 4 раз
- руководство коллективом: 4 раз

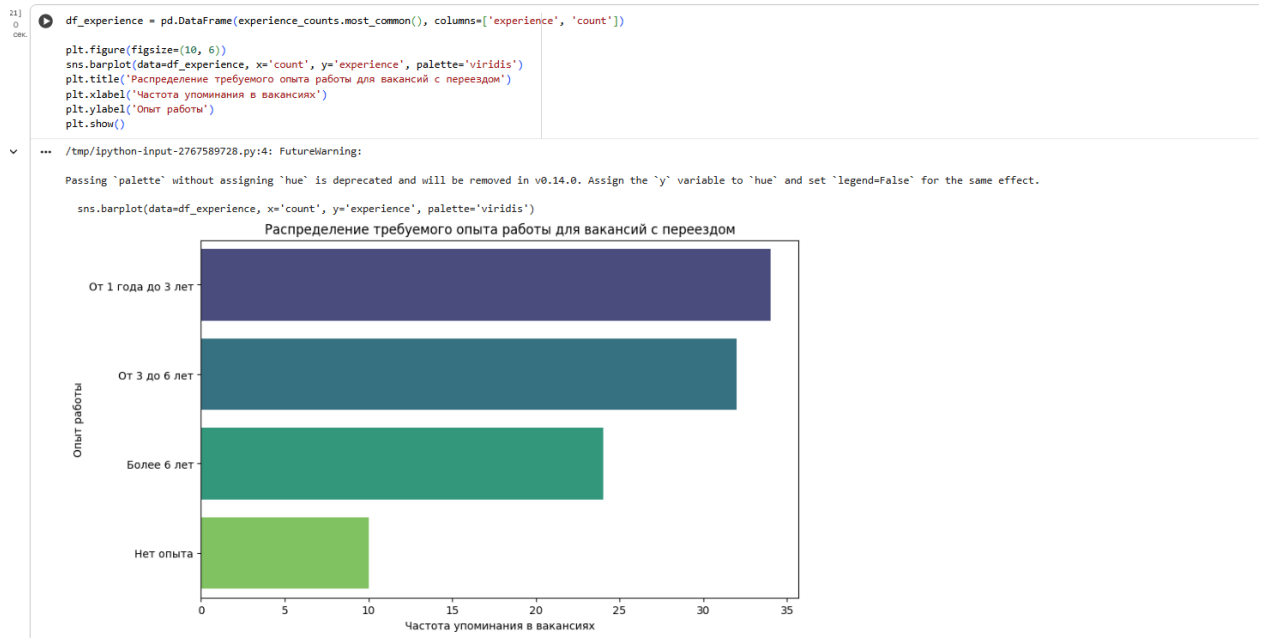
Требуемый опыт работы:

- От 1 года до 3 лет: 34 раз
- От 3 до 6 лет: 32 раз
- Более 6 лет: 24 раз
- Нет опыта: 10 раз

Визуализировал навыки:



И опыт:



Вывод: благодаря этому анализу можно легко оценить насколько конкретный человек подходит под основные требования к конкретной работе по заданным параметрам.