УДК 004.989

ПЕРСОНАЛИЗИРОВАННЫЙ АНАЛИЗ ПИЩЕВЫХ ПРОДУКТОВ С ИСПОЛЬЗОВАНИЕМ PYTHON

Канавальчик А.Д., студент гр.253502

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Жвакина А.В. – канд. техн. наук, доцент

Аннотация. В данной научной работе представлен персонализированный подход к анализу пищевых продуктов с использованием языка Python. Демонстрируется процесс загрузки и фильтрации данных с открытого API Open Food Facts, визуализация продуктов по уровню сахара, соли и калорийности, а также создание простого пользовательского интерфейса. Актуальность обусловлена потребностями групп с хроническими заболеваниями, такими как диабет и гипертония.

Ключевые слова. Персонализированный анализ, API, фильтрация, визуализация данных, Python.

Введение.

Здоровое питание – ключевой фактор в профилактике и контроле многих хронических заболеваний. В последние годы наблюдается рост интереса к персонализированным методам анализа и подбора продуктов, особенно среди людей с особенностями здоровья. К примеру, людям с сахарным диабетом необходимо тщательно следить за содержанием сахара в пище, а гипертоникам – ограничивать потребление соли. Тем не менее, современные магазины и производители предлагают огромное количество продуктов, и выбор подходящих из них без поддержки цифровых инструментов может быть затруднительным.

С применением языков программирования и аналитических библиотек открываются новые возможности для создания персонализированных систем поддержки принятия решений. Язык Python благодаря своей гибкости и богатой экосистеме инструментов особенно подходит для подобных задач. С его помощью можно автоматизировать сбор данных, произвести фильтрацию, построить визуальные отчёты и предоставить пользователю удобный интерфейс для работы с результатами анализа.

Данная работа посвящена разработке такого инструмента: описываются этапы построения системы персонализированного анализа продуктов питания, от получения информации до визуализации и взаимодействия с пользователем. Основой для анализа служат данные, полученные с открытого API Open Food Facts [1].

Основная часть.

Для реализации персонализированного анализа был разработан программный инструмент на языке Python. Он включает в себя автоматический сбор данных о продуктах питания с использованием Open Food Facts API, фильтрацию с учётом состояния здоровья пользователя и визуализацию результатов через графический интерфейс.

1 Сбор и подготовка данных

Первым шагом стало получение актуальной информации о продуктах питания из открытого API – Open Food Facts [1]. Этот сервис предоставляет свободный доступ к данным о составе, пищевой ценности и энергетических характеристиках продуктов по всему миру.

Перед проведением анализа данные предварительно очищаются от строк с пропущенными значениями, что гарантирует корректность последующих вычислений. В зависимости от выбранного пользователем состояния здоровья (например, диабет или гипертония), автоматически определяется ключевая метрика, наиболее значимая в конкретном контексте – будь то содержание сахара, соли или общая калорийность. На основе этой метрики для каждого продукта вычисляется так называемый "индекс здоровья", отражающий степень его соответствия заданным условиям: чем ниже значение индекса, тем более предпочтительным считается продукт. После ранжирования всех позиций по этому индексу система выделяет пять наиболее полезных и пять наименее подходящих продуктов, позволяя пользователю сориентироваться в выборе с учётом индивидуальных потребностей [2].

Программа обращается к API и постранично загружает данные (рисунок 1).

```python
# Шаг 1: Получение данных с нескольких страниц API Open Food Facts
def get_food_data(page_size=100, num_pages=3):
    all_food_data = []
    for page in range(1, num_pages + 1):
        url = f'https://world.openfoodfacts.org/api/v2/search?fields=code,product_name,nutriments,image_url&page_size={page_size}&page={page}'
        response = requests.get(url)
        data = response.json()

        if not data['products']:
            break  # Если на странице нет продуктов, выходим из цикла

        products = data['products']

        for product in products:
            # Пропускаем продукт, если у него нет имени (пустая строка)
            if not product.get('product_name'):
                continue

            nutriments = product.get('nutriments', {})
            all_food_data.append({
                'product_name': product.get('product_name', 'Unknown'),
                'image_url': product.get('image_url', ''),
                'proteins': nutriments.get('proteins_100g', None),
                'fats': nutriments.get('fat_100g', None),
                'sugars': nutriments.get('sugars_100g', None),
                'carbohydrates': nutriments.get('carbohydrates_100g', None),
                'energy': nutriments.get('energy_100g', None),
                'sodium': nutriments.get('salt_100g', None)
            })

    df = pd.DataFrame(all_food_data)
    return df

# Получаем данные (например, 5 страниц по 100 продуктов)
food_df = get_food_data(page_size=100, num_pages=5)
print(food_df);
```

Рисунок 1 – Загрузка данных с API

На выходе формируется таблица, где каждая строка – это отдельный продукт, а столбцы содержат ключевые питательные характеристики.

2 Персонализированная фильтрация по состоянию здоровья

После сбора данных система адаптирует рекомендации под состояние здоровья конкретного пользователя. Реализовано три режима:

– diabetic – приоритет отдается продуктам с низким содержанием сахара;

– hypertension – исключаются продукты с высоким содержанием соли;

– general – отбираются продукты с низкой энергетической ценностью.

Алгоритм персонализированной фильтрации начинается с предварительной очистки данных – удаляются все записи с отсутствующими значениями ключевых показателей, чтобы обеспечить надёжность анализа. Далее система автоматически выбирает наиболее релевантную метрику в зависимости от состояния здоровья пользователя: для людей с диабетом приоритет отдается уровню сахара, для гипертоников – содержанию соли, а для остальных пользователей – общей энергетической ценности продуктов (рисунок 2).

```python
# Шаг 2: Фильтрация данных по состоянию здоровья
def filter_by_condition(df, condition):
    # Убираем продукты с отсутствующими важными значениями
    df = df.dropna(subset=['sugars', 'energy', 'sodium'])

    # Если критерием является диабет, проверяем, чтобы сахар не был нулевым
    if condition == 'diabetic':
        df['health_score'] = df['sugars']  # Чем выше сахар, тем хуже
        criteria = 'Сахар'
        df = df[df['health_score'] != 0]  # Исключаем продукты с нулевым сахаром
        # Сортировка по сахару, минимальное количество сахара - полезные продукты
        df = df.sort_values(by='health_score', ascending=True)
    elif condition == 'hypertension':
        df['health_score'] = df['sodium']  # Чем выше натрий (соль), тем хуже
        criteria = 'Соль'
        df = df[df['health_score'] != 0]  # Исключаем продукты с нулевым натрием
        # Сортировка по соли, минимальное количество соли - полезные продукты
        df = df.sort_values(by='health_score', ascending=True)
    else:
        df['health_score'] = df['energy']  # Для других категорий используем энергию
        criteria = 'Энергия'
        df = df[df['health_score'] != 0]  # Исключаем продукты с нулевой энергией
        # Сортировка по энергии, минимальное количество энергии - полезные продукты
        df = df.sort_values(by='health_score', ascending=True)

    # Получаем топ-5 лучших и худших продуктов
    top_good = df.head(5)
    top_bad = df.tail(5)

    # Гарантируем минимум 5 продуктов для отображения
    top_good = ensure_min_products(top_good)
    top_bad = ensure_min_products(top_bad)

    return top_good, top_bad, criteria
```

Рисунок 2 – Фильтрация данных по состоянию здоровья

На основе выбранной метрики каждому продукту присваивается индивидуальная "оценка здоровья", которая отражает степень его соответствия заданным критериям: чем ниже это значение, тем продукт считается более полезным. Затем происходит ранжирование – продукты упорядочиваются от лучших к худшим в соответствии с оценкой. В результате формируются две группы: пять наиболее подходящих и пять наименее рекомендуемых продуктов. Такой подход обеспечивает высокую точность и значимость рекомендаций, адаптированных под конкретные потребности пользователя.

3 Визуализация результатов

Для наглядности и простоты восприятия данные визуализируются в виде горизонтальных столбчатых диаграмм. Одна диаграмма показывает продукты с низким содержанием "опасного" компонента, а вторая – с высоким (рисунок 3).

```python
# Шаг 3: Визуализация данных
def visualize_top_products(top_5_good, top_5_bad, criteria):
    fig, ax = plt.subplots(1, 2, figsize=(12, 6))

    # Проверка на наличие данных для полезных продуктов
    if not top_5_good.empty:
        ax[0].barh(top_5_good['product_name'], top_5_good['health_score'], color='green')
        ax[0].set_title(f"Топ 5 полезных продуктов (низкий {criteria})")
        ax[0].set_xlabel(f"Содержание {criteria} (г/100г)")
    else:
        ax[0].axis('off')  # Если нет данных, не показываем график

    # Проверка на наличие данных для вредных продуктов
    if not top_5_bad.empty:
        ax[1].barh(top_5_bad['product_name'], top_5_bad['health_score'], color='red')
        ax[1].set_title(f"Топ 5 вредных продуктов (высокий {criteria})")
        ax[1].set_xlabel(f"Содержание {criteria} (г/100г)")
    else:
        ax[1].axis('off')  # Если нет данных, не показываем график

    plt.tight_layout()
    plt.show()
```

Рисунок 3 – Визуализация данных

Модуль визуализации преобразует числовые данные в наглядную форму, делая результат анализа легко воспринимаемым даже без технических знаний. Интерфейс отображает два отдельных графика: один демонстрирует топ-5 наиболее полезных продуктов, другой – те, которых желательно избегать [3]. На горизонтальной оси каждого графика представлены значения ключевого параметра – сахара, соли или энергии – в зависимости от выбранного состояния здоровья пользователя. Вертикальная ось содержит наименования продуктов, отобранных в ходе анализа.

Для усиления восприятия используются цветовые акценты: зелёный цвет подчёркивает благоприятные продукты, в то время как красный сигнализирует о потенциально вредных. Такой подход позволяет быстро и интуитивно сравнить альтернативы, делая выбор более осознанным. На рисунках 4-6 представлены примеры графиков, полученных в результате работы программы.

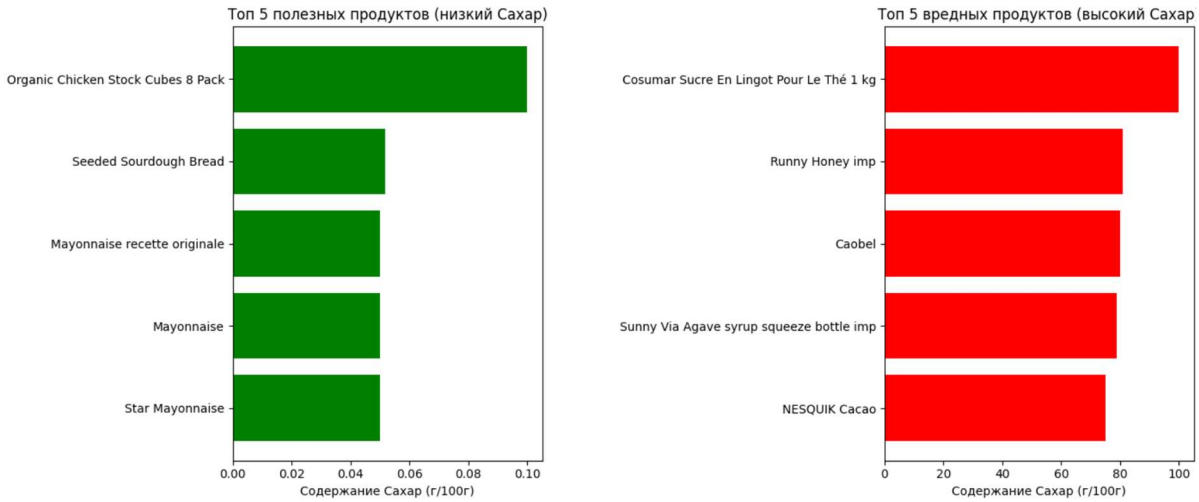

Рисунок 4 – Топ вредных и полезных продуктов для категории людей с диабетом

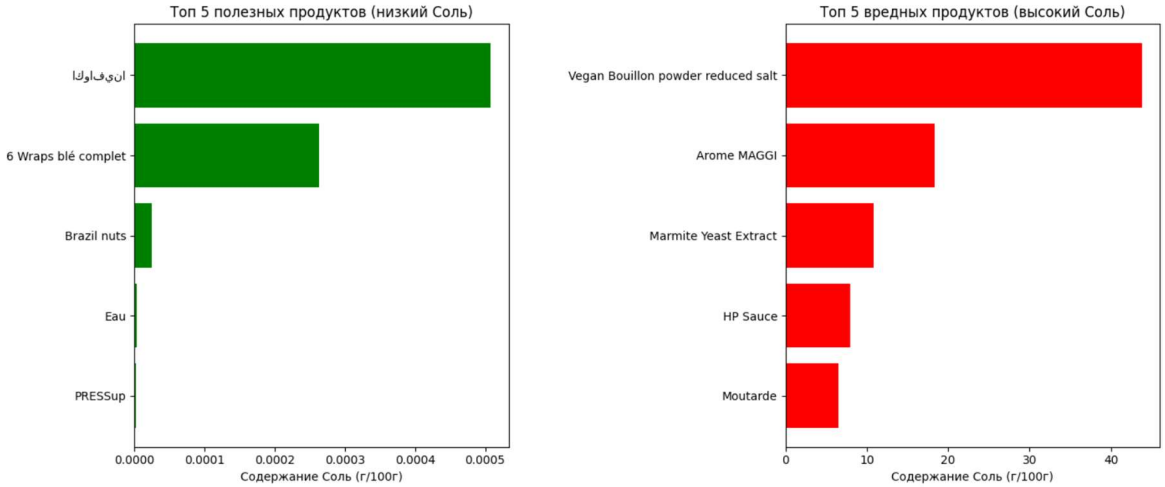

Рисунок 5 – Топ вредных и полезных продуктов для категории людей с гипертонией

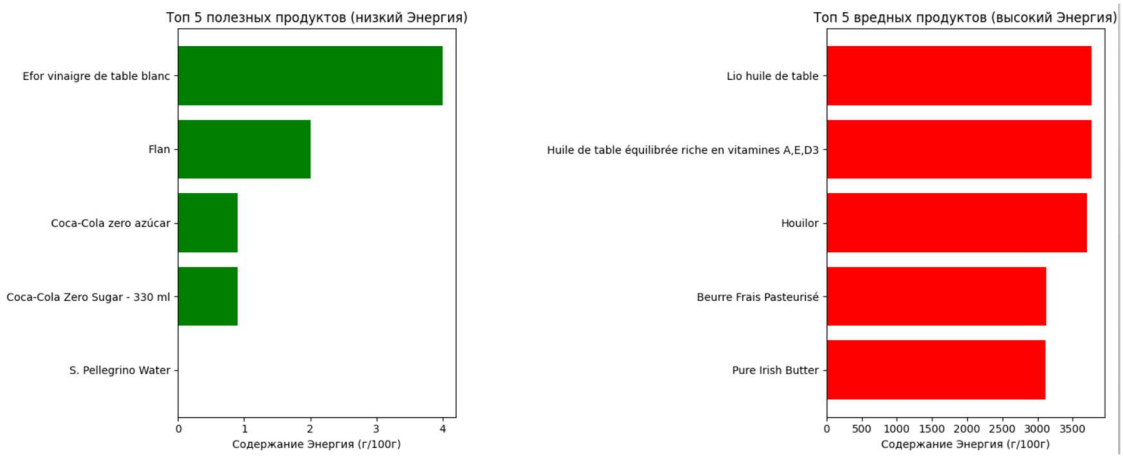

Рисунок 6 – Топ вредных и полезных продуктов относительно энергетической ценности

4 Интерактивный пользовательский интерфейс

Чтобы сделать приложение удобным для широкой аудитории, добавлен простой графический интерфейс на базе Tkinter [4]. Он позволяет выбрать интересующую категорию состояния здоровья из выпадающего списка и сразу получить визуальные рекомендации (рисунок.7).

```python
# Шаг 4: Создание интерфейса с Tkinter
def on_select_condition(event):
    condition = condition_combobox.get()
    top_5_good, top_5_bad, criteria = filter_by_condition(food_df, condition)
    visualize_top_products(top_5_good, top_5_bad, criteria)

# Создаем главное окно
root = tk.Tk()
root.title("Food Health Checker")

# Заголовок
label = tk.Label(root, text="Выберите категорию людей:")
label.pack(pady=10)

# Выпадающий список для выбора категории
condition_combobox = ttk.Combobox(root, values=["diabetic", "hypertension", "general"])
condition_combobox.bind("<<ComboboxSelected>>", on_select_condition)
condition_combobox.pack(pady=10)

# Запуск интерфейса
root.mainloop()
```

Рисунок 7 – Использование библиотеки Tkinter для построения пользовательского интерфейса

Графический интерфейс приложения построен таким образом, чтобы взаимодействие с системой было максимально простым и интуитивным. При запуске пользователь видит главное окно с лаконичным оформлением и выпадающим списком, в котором предлагается выбрать одну из категорий – диабет, гипертония или общее снижение калорийности.

Как только пользователь делает выбор, система автоматически запускает процесс анализа: происходит фильтрация данных в соответствии с выбранным состоянием здоровья, а затем отображаются соответствующие визуализации. Всё это происходит в реальном времени – без необходимости перезапуска приложения или дополнительных действий со стороны пользователя.

Интерфейс компактный, лёгкий и не требует специальных навыков, что делает его удобным как для личного, так и для образовательного или медицинского использования.

На рисунке 8 представлен внешний вид интерфейса.

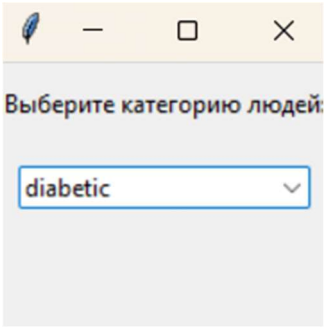

Рисунок 8 – Внешний вид интерфейса

Заключение.

Разработанная система персонализированного анализа пищевых продуктов на Python демонстрирует, как современные технологии могут помочь людям в принятии осознанных решений о питании. Система автоматически загружает данные, учитывает состояние здоровья пользователя и визуализирует полученные результаты, облегчая восприятие информации.

Преимущества разработанного решения:

– использование открытых данных без лицензий;
– высокая скорость обработки и фильтрации;
– наглядная визуализация и простой интерфейс;
– возможность расширения и интеграции с другими системами (например, фитнес-приложениями).

Перспективы развития проекта включают:

– добавление анализа микронутриентов (витамины, минералы);
– прогнозирование рисков с использованием ИИ;
– создание мобильного приложения с использованием той же логики;
– поддержка пользовательских профилей с историей предпочтений.

Таким образом, система закладывает основу для создания интеллектуальных помощников в области питания и здоровья, ориентированных на конкретные потребности человека.

Список использованных источников:

1. Open Food Facts API [Electronic resource]. – Mode of access: https://world.openfoodfacts.org/data. – Date of access: 10.04.2025.

2. Pandas: библиотека для анализа данных [Электронный ресурс]. – Режим доступа: https://pandas.pydata.org. – Дата доступа: 10.04.2025.

3. Matplotlib: визуализация данных [Электронный ресурс]. – Режим доступа: https://matplotlib.org. – Дата доступа: 10.04.2025.

4. Tkinter – встроенная библиотека Python для GUI [Электронный ресурс]. – Режим доступа: https://docs.python.org/3/library/tkinter.html. – Дата доступа: 10.04.2025.

UDC 004.989

PERSONALIZED ANALYSIS OF FOOD PRODUCTS USING PYTHON

Kanavalchik A.D.

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Zhvakina A.V. – Candidate of Technical Sciences, Associate Professor

Annotation. This scientific work presents a personalized approach to the analysis of food products using the Python programming language. It demonstrates the process of data loading and filtering from the open Open Food Facts API, visualization of products based on sugar, salt, and calorie levels, as well as the creation of a simple user interface. The relevance of this study is driven by the needs of groups with chronic conditions such as diabetes and hypertension.

Keywords. Personalized analysis, API, filtering, data visualization, Python.