

Создание веб-приложения на Flask для прогнозирования цен с использованием моделей машинного обучения

Малик Александр Игоревич

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной статье описывается процесс создания веб-приложения на Flask, которое позволяет пользователям делать прогнозы цены на основе обученных моделей машинного обучения. Приложение предоставляет интерфейс, где пользователи могут вводить финансовые данные и выбирать модель для прогнозирования. Используемые модели, такие как Ridge, XGBoost и LightGBM, были предварительно обучены на данных о цене биткойна. Статья включает в себя пошаговое руководство по созданию приложения, интеграции моделей и настройке пользовательского интерфейса.

Ключевые слова: Flask, веб-приложение, машинное обучение, прогнозирование цен, Ridge, XGBoost, LightGBM, Python, финансовые данные, интеграция моделей.

Building a flask web application for price prediction using machine learning models

Malik Aleksandr Igorevich

Sholom-Aleichem Priamursky State University

Student

Abstract

This article describes the process of creating a Flask web application that allows users to make price predictions based on trained machine learning models. The application provides an interface where users can input financial data and select a model for prediction. The models used, such as Ridge, XGBoost, and LightGBM, were pre-trained on Bitcoin price data. The article includes a step-by-step guide on building the application, integrating the models, and setting up the user interface.

Keywords: Flask, web application, machine learning, price prediction, Ridge, XGBoost, LightGBM, Python, financial data, model integration

1 Введение

1.1 Актуальность

В условиях быстрого развития технологий и роста объема данных, машинное обучение (ML) стало неотъемлемой частью современных аналитических инструментов, особенно в финансовой сфере.

Прогнозирование цен активов, таких как криптовалюты и акции, является одной из ключевых задач, решаемых с помощью ML. Однако доступ к этим мощным инструментам зачастую ограничен из-за сложности их применения и отсутствия удобных интерфейсов.

Создание веб-приложений на основе моделей машинного обучения позволяет сделать эти технологии доступными для широкой аудитории, включая аналитиков, трейдеров и инвесторов, не обладающих глубокими знаниями в программировании. Flask, как легковесный и гибкий фреймворк, предоставляет идеальную платформу для разработки таких приложений, объединяя мощь машинного обучения с удобством использования.

В текущих условиях финансовых рынков, где быстрый доступ к точной аналитике может стать решающим фактором, создание простого и функционального инструмента для прогнозирования цен становится не только актуальным, но и необходимым. Такой инструмент позволяет оперативно принимать информированные решения, минимизируя риски и увеличивая потенциальную прибыль.

1.2 Обзор исследований

Исследование под названием *«Прогнозирование диабета с использованием машинного обучения и Flask»* было проведено группой ученых из Университета VIT, включая Н. К. К. Раджу, К. Кришнамурти, Б. П. Бхагавада и Н. Шанкара. В работе рассматривается применение методов машинного обучения для предсказания вероятности развития диабета у пациентов. Диабет, оставаясь неизлечимым заболеванием, приводит к множеству долгосрочных осложнений, таких как повреждения сердца, сосудов, глаз, почек и нервов. Авторы использовали четыре алгоритма машинного обучения — метод опорных векторов (SVM), метод k-ближайших соседей (KNN), логистическую регрессию и классификатор случайного леса — для построения предсказательных моделей и сравнили их точность для выявления наиболее эффективного подхода. Лучшая модель была интегрирована в веб-приложение с использованием Flask, что позволило создать удобный инструмент для прогнозирования риска диабета [1]. Исследование под названием *«Визуализация кластеризации и предсказание классов с использованием Flask на эталонном наборе данных для неконтролируемых методов в машинном обучении»* было проведено А. Нат и Ш. Нема из Института технологий Уша Миттал. В работе рассматриваются передовые методы кластеризации и уменьшения размерности данных, такие как K-means, иерархическая кластеризация, метод главных компонент (PCA) и линейный дискриминантный анализ (LDA). Для анализа использовались три различных набора данных из эталонной базы данных UCI (IRIS, Wine и Seed). Основной целью исследования было формирование хороших кластеризационных паттернов для каждого набора данных и сравнение их точности с использованием различных классификаторов машинного обучения, таких как логистическая регрессия, K-ближайшие соседи (KNN), метод опорных векторов (SVM), наивный байесовский классификатор,

классификатор на основе дерева решений и случайный лес. Результаты показали, что методы кластеризации K-means и иерархическая кластеризация обеспечивают лучшее качество кластеризации по сравнению с методами уменьшения размерности, а классификатор KNN показал наивысшую точность [2]. В исследовании *«GUI-ориентированный подход к прогнозированию сердечных заболеваний с использованием алгоритмов машинного обучения и Flask API»*, проведенном С. Кумаром Бозе, Ш. Гош и С. Дас под руководством специалистов из Технологического института Херитедж, предложена система для раннего выявления сердечных заболеваний. Основной целью работы было разработать систему обнаружения сердечных заболеваний с помощью алгоритмов машинного обучения. В рамках исследования была создана графическая пользовательская среда (GUI) с использованием HTML и CSS, а также интегрирован Flask для работы с серверной частью. Этот подход позволяет пользователям получать прогнозы относительно наличия сердечных заболеваний, на основе введенных данных [3]. В статье *«Модель предсказания множества заболеваний с использованием машинного обучения и Flask API»*, опубликованной на 5-й Международной конференции по коммуникационным и электронным системам (ICCES) в июне 2020 года и изданной IEEE, исследуются возможности интеграции нескольких моделей машинного обучения для предсказания различных заболеваний. Исследование, проведенное А. Ягантисваруду, направлено на создание системы, способной одновременно анализировать и прогнозировать риск нескольких заболеваний, включая диабет, диабетическую ретинопатию, сердечные заболевания и рак молочной железы. В статье описана реализация модели с использованием алгоритмов машинного обучения, TensorFlow и Flask API, а также метод сохранения и загрузки моделей с помощью Python pickling. Система позволяет пользователям вводить параметры болезни через API и получать прогнозы состояния пациента. Эта многозадачная система направлена на улучшение мониторинга состояния пациента и снижение уровня смертности, путем включения в анализ дополнительных параметров, таких как уровень креатинина, калия и другие, которые часто не учитываются в существующих системах [4].

1.3 Цель исследования

Целью данного исследования является разработка и внедрение веб-приложения на основе Flask, которое предоставляет пользователям возможность осуществлять прогнозирование цен финансовых активов, используя предобученные модели машинного обучения. В рамках исследования планируется интегрировать популярные модели, такие как Ridge, XGBoost и LightGBM, создать удобный пользовательский интерфейс для ввода данных и выбора модели, а также продемонстрировать эффективность такого подхода для решения задач прогнозирования в финансовой сфере.

2 Материалы и методы

Для реализации проекта по созданию веб-приложения для прогнозирования цен были использованы исторические данные о цене биткойна, включающие такие параметры, как цена открытия, цена закрытия, максимальная и минимальная цены, а также объем торгов. Эти данные, представленные в формате CSV, были загружены и обработаны с использованием библиотеки Pandas. В процессе обработки данных рассчитывались скользящие средние и стандартные отклонения для определенных периодов, что позволило улучшить качество прогнозирования. Пропущенные значения в данных были удалены для обеспечения корректности обучения моделей.

Основными моделями машинного обучения, использованными в проекте, стали Ridge, XGBoost и LightGBM. Ridge представляет собой регрессионную модель с регуляризацией, которая хорошо справляется с проблемой мультиколлинеарности. XGBoost и LightGBM, являясь моделями градиентного бустинга, были выбраны за их высокую производительность и способность эффективно обрабатывать большие объемы данных. Модели были обучены на исторических данных о цене биткойна, а для оценки их качества использовалось разделение данных на обучающую и тестовую выборки. Этот метод позволяет объективно оценить, как хорошо модель будет работать на новых данных, что критично для задач прогнозирования в финансовой сфере.

Для создания веб-приложения был выбран Flask — легковесный и гибкий фреймворк для разработки веб-приложений на Python. Приложение позволяет пользователям вводить финансовые данные через веб-интерфейс и выбирать модель для прогнозирования. Пользовательский интерфейс был разработан с использованием HTML и CSS, а для повышения его визуальной привлекательности и удобства был применен фреймворк Bootstrap.

Интеграция обученных моделей в веб-приложение была осуществлена с помощью библиотеки Joblib, которая используется для сохранения и загрузки моделей машинного обучения. Каждая модель была сохранена в отдельный файл, что позволило легко загружать их в Flask-приложение по запросу пользователя. В итоге, пользователи могут выбирать нужную модель, вводить данные и получать прогнозы непосредственно на веб-странице, что делает приложение удобным и доступным инструментом для анализа и прогнозирования цен на основе машинного обучения.

3 Результаты и обсуждения

3.1 Экспорт моделей

Известно, что все требуемые модели уже обучены и готовы для использования, но они расположены в облаке ноутбука Google Colab и требуется их экспортировать из него. Для этого, воспользуемся библиотекой joblib и для примера реализуем экспорт трех моделей: Ridge, XGBRegressor и LGBMRegressor (Рис. 1).

```
import joblib

for model in models:
    model_name = model.__class__.__name__
    if model_name == "Ridge" or model_name == "XGBRegressor" or model_name == "LGBMRegressor":
        joblib.dump(model, f'{model_name}.pkl')
```

Рисунок 1. Функция экспорта моделей

После чего, в проводнике ноутбука можно увидеть все экспортированные модели в виде файлов с расширением - **.pkl** (Рис. 2).

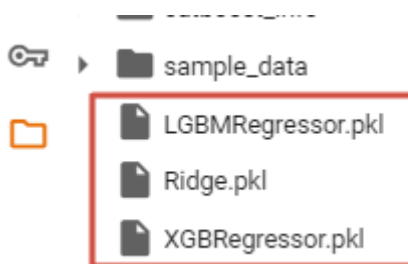


Рисунок 2. Экспортированные модели в панели проводника ноутбука

3.2 Создание Flask-приложения с интеграцией моделей

Следующим шагом будет создание Flask-приложения с последующей интеграцией экспортированных моделей.

Для начала создадим папку в любом удобном месте, в данном случае будет в качестве места был выбран рабочий стол. Внутри этой папки сформируем структуру, создав необходимые папки, файлы и переместив ранее экспортированные модели в папку `models`:

```
folder/
  models/
    LGBMRegressor.pkl
    Ridge.pkl
    XGBRegressor.pkl
  templates/index.html
  app.py
```

Вся работа выполнена в среде разработки PyCharm, в связи с этим требуется открыть папку приложения в качестве PyCharm-проекта. Далее в файле `index.html` создадим простую html-структуру, которая будет представлением будущего приложения, а также добавим простейший внешний вид, используя `bootstrap` (Рис. 3).

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Предсказание цены на основе модели</title>
  <!-- Подключение Bootstrap CSS -->
  <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
  <style>
    body {
      padding-top: 20px;
      padding-bottom: 20px;
    }
    .container {
      max-width: 600px;
    }
  </style>
</head>
<body>
  <div class="container">
    <center>
      <h1 class="mb-4">Выберите модель и введите данные для прогноза</h1>
    </center>
    {% if prediction %}
    <h2 class="mt-4">Результат: {{ prediction }}</h2>
    {% endif %}
  </div>

  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

Рисунок 3. Структура представления приложения

Следом за этим, добавим в представление форму с методом запроса: POST (Рис. 4). Форма будет отправлять данные непосредственно в сам обработчик и после получения, который после обработки – обновит представление с результатом просчетов. В качестве данных будут подаваться основные признаки, по которым модель обучалась.

```

<form method="POST">
  <div class="form-group">
    <label for="model">Выберите модель:</label>
    <select name="model" id="model" class="form-control">
      <option value="Ridge.pkl">Ridge</option>
      <option value="XGBRegressor.pkl">XGBoost</option>
      <option value="LGBMRegressor.pkl">LightGBM</option>
    </select>
  </div>
  <div class="form-group">
    <label for="volume">Объем торгов:</label>
    <input type="text" id="volume" name="volume" class="form-control" placeholder="Введите объем торгов">
  </div>
  <div class="form-group">
    <label for="open_price">Цена открытия:</label>
    <input type="text" id="open_price" name="open_price" class="form-control" placeholder="Введите цену открытия">
  </div>
  <div class="form-group">
    <label for="close_price">Цена закрытия:</label>
    <input type="text" id="close_price" name="close_price" class="form-control" placeholder="Введите цену закрытия">
  </div>
  <div class="form-group">
    <label for="high_price">Самая высокая цена:</label>
    <input type="text" id="high_price" name="high_price" class="form-control" placeholder="Введите самую высокую цену">
  </div>
  <div class="form-group">
    <label for="low_price">Самая низкая цена:</label>
    <input type="text" id="low_price" name="low_price" class="form-control" placeholder="Введите самую низкую цену">
  </div>
  <div class="form-group">
    <label for="moving_average_7">Скользящее среднее за 7 дней:</label>
    <input type="text" id="moving_average_7" name="moving_average_7" class="form-control" placeholder="Введите скользящее среднее за 7 дней">
  </div>
  <div class="form-group">
    <label for="moving_average_14">Скользящее среднее за 14 дней:</label>
    <input type="text" id="moving_average_14" name="moving_average_14" class="form-control" placeholder="Введите скользящее среднее за 14 дней">
  </div>
  <div class="form-group">
    <label for="std_dev_7">Стандартное отклонение за 7 дней:</label>
    <input type="text" id="std_dev_7" name="std_dev_7" class="form-control" placeholder="Введите стандартное отклонение за 7 дней">
  </div>
  <button type="submit" class="btn btn-primary">Предсказать</button>
</form>

```

Рисунок 4. Структура основной формы приложения

Для удобства пользования и проверки – добавим автозаполнение значений формы при помощи JS-скрипта, добавленного после подключения bootstrap (Рис. 5).

```

<script>
    document.addEventListener('DOMContentLoaded', (event) => {
        const defaultValues = {
            volume: '4500.25',
            open_price: '29000.0',
            close_price: '29500.0',
            high_price: '30000.0',
            low_price: '28000.0',
            moving_average_7: '29000.0',
            moving_average_14: '29500.0',
            std_dev_7: '300.5'
        };

        for (const [key, value] of Object.entries(defaultValues)) {
            const input = document.getElementById(key);
            if (!input.value) {
                input.value = value;
            }
        }
    });
</script>

```

Рисунок 5. JS-скрипт, заполняющий поля форм значениями

После того, как шаблон представления был создан, приступим к созданию основы приложения, для этого установим в интерпретатор все необходимые для работы библиотеки в терминале проекта командой: **pip install flask joblib pandas scikit-learn xgboost lightgbm** (Рис. 6).

```

PS C:\Users\tseve\Desktop\MLPrediction> pip install flask joblib pandas scikit-learn xgboost lightgbm
Collecting flask
  Using cached flask-3.0.3-py3-none-any.whl.metadata (3.2 kB)
Collecting joblib
  Using cached joblib-1.4.2-py3-none-any.whl.metadata (5.4 kB)
Collecting pandas
  Using cached pandas-2.2.2-cp312-cp312-win_amd64.whl.metadata (19 kB)
Collecting scikit-learn
  Using cached scikit_learn-1.5.1-cp312-cp312-win_amd64.whl.metadata (12 kB)
Collecting xgboost
  Using cached xgboost-2.1.1-py3-none-win_amd64.whl.metadata (2.1 kB)
Collecting lightgbm
  Using cached lightgbm-4.5.0-py3-none-win_amd64.whl.metadata (17 kB)
Requirement already satisfied: Werkzeug>=3.0.0 in c:\users\tseve\appdata\local\programs\python\python312\lib\site-packages (from flask) (3.0.1)
Requirement already satisfied: Jinja2>=3.1.2 in c:\users\tseve\appdata\local\programs\python\python312\lib\site-packages (from flask) (3.1.3)
Requirement already satisfied: itsdangerous>=2.1.2 in c:\users\tseve\appdata\local\programs\python\python312\lib\site-packages (from flask) (2.1.2)
Requirement already satisfied: click>=8.1.3 in c:\users\tseve\appdata\local\programs\python\python312\lib\site-packages (from flask) (8.1.7)
Requirement already satisfied: blinker>=1.6.2 in c:\users\tseve\appdata\local\programs\python\python312\lib\site-packages (from flask) (1.7.0)
Requirement already satisfied: numpy>=1.26.0 in c:\users\tseve\appdata\local\programs\python\python312\lib\site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\tseve\appdata\local\programs\python\python312\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\tseve\appdata\local\programs\python\python312\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\tseve\appdata\local\programs\python\python312\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: scipy>=1.6.0 in c:\users\tseve\appdata\local\programs\python\python312\lib\site-packages (from scikit-learn) (1.12.0)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\tseve\appdata\local\programs\python\python312\lib\site-packages (from scikit-learn) (3.4.0)
Requirement already satisfied: colorama in c:\users\tseve\appdata\local\programs\python\python312\lib\site-packages (from click>=8.1.3->flask) (0.4.6)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\tseve\appdata\local\programs\python\python312\lib\site-packages (from Jinja2>=3.1.2->flask) (2.1.5)
Requirement already satisfied: six>=1.5 in c:\users\tseve\appdata\local\programs\python\python312\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Using cached flask-3.0.3-py3-none-any.whl (101 kB)
Using cached joblib-1.4.2-py3-none-any.whl (301 kB)
Using cached pandas-2.2.2-cp312-cp312-win_amd64.whl (11.5 MB)
Using cached scikit_learn-1.5.1-cp312-cp312-win_amd64.whl (10.9 MB)
Using cached xgboost-2.1.1-py3-none-win_amd64.whl (124.9 MB)
Using cached lightgbm-4.5.0-py3-none-win_amd64.whl (1.4 MB)
Installing collected packages: joblib, xgboost, scikit-learn, pandas, lightgbm, flask

```

Рисунок 6. Выполнение установки библиотек в терминале проекта

Далее, напишем само приложение, в основе которого лежат: вывод представления и обработчик формы, принимающий все признаки, по которым модель предсказывает ответ (Рис. 7).

```
from flask import Flask, request, render_template
import joblib

# Инициализация Flask приложения
app = Flask(__name__)

# Главный маршрут (домашняя страница)

@app.route(rule: '/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        # Получение данных из формы
        model_name = request.form['model']
        feature_values = [float(request.form['volume']),
                           float(request.form['open_price']),
                           float(request.form['close_price']),
                           float(request.form['high_price']),
                           float(request.form['low_price']),
                           float(request.form['moving_average_7']),
                           float(request.form['moving_average_14']),
                           float(request.form['std_dev_7'])]

        # Загрузка модели
        model = joblib.load(f'models/{model_name}')

        # Прогнозирование
        prediction = model.predict([feature_values])

        # Отображение результата на веб-странице
        return render_template(template_name_or_list: 'index.html', prediction=prediction[0])

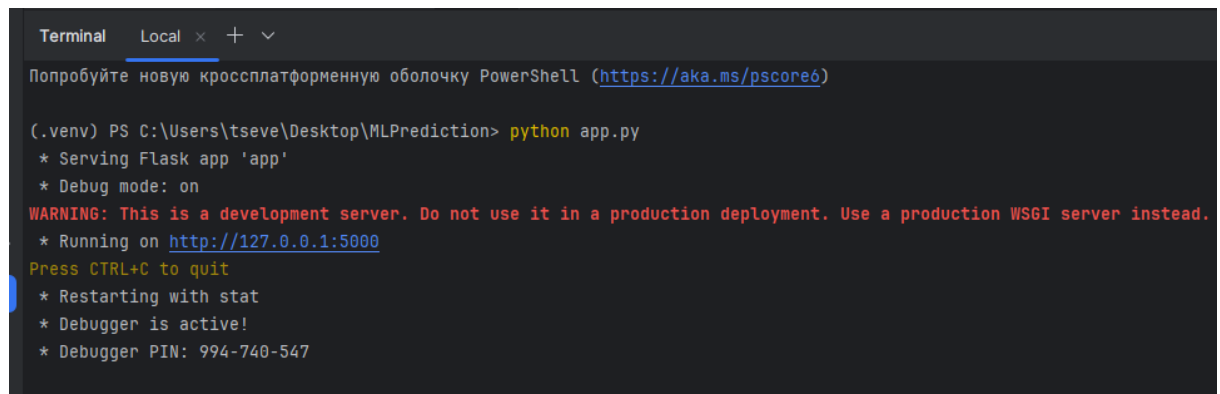
    # Если метод GET, просто отобразить форму
    return render_template('index.html')

# Основная точка входа для запуска приложения
if __name__ == '__main__':
    # Запуск Flask сервера
    app.run(debug=True)
```

Рисунок 7. Код основы Flask-приложения

3.3 Запуск приложения и его проверка

Для запуска приложения достаточно запустить файл `app.py` интерпретатором, сделать это можно прямо в IDE, либо через командную строку, для этого нужно перейти в папку проекта и ввести **python app.py** (Рис. 8).



```
Terminal Local x + v
Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/pscore6)

(.venv) PS C:\Users\tseve\Desktop\MLPrediction> python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 994-740-547
```

Рисунок 8. Выполнение запуска приложения в терминале проекта

В результате, после проделанных действий можно увидеть соответствующие надписи в консоли интерпретатора, в которых указаны параметры приложения и что самое главное – адрес приложения, развернутого в локальной сети, как правило - <http://127.0.0.1:5000>.

Перейдя по указанной ссылке, можно увидеть рабочее веб-приложение с уже заполненными данными для проверки (Рис. 9).

Выберите модель и введите
данные для прогноза

Выберите модель:

Ridge

Объем торгов:

4500.25

Цена открытия:

29000.0

Цена закрытия:

29500.0

Самая высокая цена:

30000.0

Самая низкая цена:

28000.0

Скользящее среднее за 7 дней:

29000.0

Скользящее среднее за 14 дней:

29500.0

Стандартное отклонение за 7 дней:

300.5

Предсказать

Рисунок 9. Вид на главную форму приложения

После выбора модели и ввода данных, можем получить результат, нажав на соответствующую кнопку «Предсказать» (Рис. 10).

00000

Предсказать

Результат: 27383.115586739936

Рисунок 10. Вывод результата после успешного обращения к выбранной модели

Выводы

В ходе данного исследования было разработано функциональное веб-приложение на базе Flask, позволяющее прогнозировать цены финансовых активов с использованием различных моделей машинного обучения, таких как Ridge, XGBoost и LightGBM. Приложение успешно интегрировало предобученные модели и предоставило пользователям интуитивно понятный интерфейс для ввода данных и получения прогнозов.

Разработанное приложение представляет собой полезный инструмент для анализа и прогнозирования цен, который может быть полезен аналитикам, трейдерам и инвесторам. Оно демонстрирует потенциал машинного обучения в финансовых приложениях и может быть расширено и улучшено в будущем для еще более точного и удобного прогнозирования на основе новых данных и моделей.

Библиографический список

1. Raju N.K.K. et al. Diabetes Prediction Using Machine Learning and Flask // Biomedical & Pharmacology Journal. 2024. Т. 17. № 2. С. 1307-1316.
2. Nath A., Nema S. Clustering Visualization and Class Prediction using Flask of Benchmark Dataset for Unsupervised Techniques in Machine learning // International Journal of Innovative Technology and Exploring Engineering. 2020. Т. 9. № 7. С. 1297-1302.
3. Bose S. K., Ghosh S., Das S. A GUI-Based Approach to Predict Heart Disease Using Machine Learning Algorithms and Flask API // Computational Intelligence in Communications and Business Analytics. 2023. С. 301-309. DOI: 10.1007/978-3-031-48876-4_23.
4. Yaganteeswarudu A. Multi Disease Prediction Model by using Machine Learning and Flask API // 2020 5th International Conference on Communication and Electronics Systems (ICCES). 2020. С. 1-6. DOI: 10.1109/ICCES48766.2020.9137896.
5. Lakshmanarao A., Raja Babu M., Bala Krishna M.M. Malicious URL Detection using NLP, Machine Learning and FLASK // 2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES). 2021. С. 1-6. DOI: 10.1109/ICSES52305.2021.9633889.