**Московский государственный технический   
университет им. Н. Э. Баумана**

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №7

Выполнил: Проверил:  
Фень Н.Т. Гапанюк Ю.Е.  
группа ИУ5Ц-82Б

Дата: 31.05.25 Дата:

Подпись: Подпись:

Москва, 2025 г.

**Цель лабораторной работы:** изучение возможностей демонстрации моделей машинного обучения с помощью веб-приложений.

**Задание:**

Разработайте макет веб-приложения, предназначенного для анализа данных.

Вариант 2. Макет должен быть реализован для нескольких моделей машинного обучения. Макет должен позволять:

* выбирать модели для обучения,
* производить обучение,
* осуществлять просмотр результатов обучения, в том числе в виде графиков.

**Ход выполнения:**

**App.py**

import streamlit as st

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import (

    accuracy\_score, confusion\_matrix,

    roc\_curve, auc, precision\_recall\_curve,

    RocCurveDisplay

)

from sklearn.preprocessing import label\_binarize

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.datasets import load\_iris, load\_breast\_cancer

from sklearn.multiclass import OneVsRestClassifier

# Настройка страницы

st.set\_page\_config(

    page\_title="ML Model Explorer",

    page\_icon="🧠",

    layout="wide"

)

# Заголовок приложения

st.title("🧠 Демонстратор моделей машинного обучения")

st.write("""

Веб-приложение для обучения и визуализации работы различных алгоритмов машинного обучения.

Загрузите свои данные или используйте встроенные наборы данных.

""")

# Боковая панель для настроек

with st.sidebar:

    st.header("⚙️ Настройки эксперимента")

    dataset\_type = st.radio("Источник данных:",

                           ["Встроенные данные", "Загрузить CSV"])

    # Выбор набора данных

    if dataset\_type == "Встроенные данные":

        dataset\_name = st.selectbox(

            "Выберите набор данных:",

            ["Ирисы Фишера", "Рак молочной железы"]

        )

    else:

        data\_file = st.file\_uploader("Загрузите CSV файл", type="csv")

    # Выбор модели

    model\_name = st.selectbox(

        "Выберите модель:",

        ["Логистическая регрессия", "Случайный лес",

         "Градиентный бустинг", "SVM"]

    )

    # Параметры модели

    st.subheader("Параметры модели:")

    test\_size = st.slider("Доля тестовых данных:", 0.1, 0.5, 0.2)

    random\_state = st.number\_input("Random State:", 0, 100, 42)

    # Кнопка обучения

    train\_button = st.button("Обучить модель")

# Основная область

if dataset\_type == "Встроенные данные":

    if dataset\_name == "Ирисы Фишера":

        data = load\_iris()

        is\_multiclass = True

    else:

        data = load\_breast\_cancer()

        is\_multiclass = False

    X = pd.DataFrame(data.data, columns=data.feature\_names)

    y = pd.Series(data.target, name='target')

    class\_names = data.target\_names if hasattr(data, 'target\_names') else None

else:

    if data\_file is not None:

        data = pd.read\_csv(data\_file)

        X = data.iloc[:, :-1]

        y = data.iloc[:, -1]

        is\_multiclass = len(y.unique()) > 2

        class\_names = None

    else:

        st.info("⏳ Ожидаю загрузки данных...")

        st.stop()

# Отображение данных

st.header("🔍 Исследование данных")

st.subheader("Первые 5 строк данных:")

st.write(X.head())

st.subheader("Статистика данных:")

st.write(X.describe())

# Выбор модели

models = {

    "Логистическая регрессия": LogisticRegression(max\_iter=10000),

    "Случайный лес": RandomForestClassifier(n\_estimators=100),

    "Градиентный бустинг": GradientBoostingClassifier(),

    "SVM": SVC(probability=True)

}

if train\_button:

    st.header("📊 Результаты обучения")

    # Разделение данных

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(

        X, y, test\_size=test\_size, random\_state=random\_state

    )

    # Обучение модели

    model = models[model\_name]

    with st.spinner(f"Обучение модели {model\_name}..."):

        model.fit(X\_train, y\_train)

        y\_pred = model.predict(X\_test)

        y\_proba = model.predict\_proba(X\_test) if hasattr(model, "predict\_proba") else None

    # Оценка модели

    accuracy = accuracy\_score(y\_test, y\_pred)

    st.success(f"Точность модели: \*\*{accuracy:.4f}\*\*")

    # Визуализация результатов

    col1, col2 = st.columns(2)

    with col1:

        st.subheader("Матрица ошибок")

        cm = confusion\_matrix(y\_test, y\_pred)

        fig, ax = plt.subplots(figsize=(8, 6))

        sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', ax=ax)

        if class\_names is not None:

            ax.set\_xticklabels(class\_names)

            ax.set\_yticklabels(class\_names)

        ax.set\_xlabel('Предсказанный класс')

        ax.set\_ylabel('Истинный класс')

        st.pyplot(fig)

    with col2:

        if y\_proba is not None:

            st.subheader("ROC-кривая")

            fig, ax = plt.subplots(figsize=(8, 6))

            if is\_multiclass:

                # Многоклассовая ROC-кривая

                y\_test\_bin = label\_binarize(y\_test, classes=np.unique(y))

                n\_classes = y\_test\_bin.shape[1]

                for i in range(n\_classes):

                    fpr, tpr, \_ = roc\_curve(y\_test\_bin[:, i], y\_proba[:, i])

                    roc\_auc = auc(fpr, tpr)

                    ax.plot(fpr, tpr, lw=2,

                            label=f'Класс {i} (AUC = {roc\_auc:.2f})')

                ax.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

                ax.set\_xlim([0.0, 1.0])

                ax.set\_ylim([0.0, 1.05])

                ax.set\_xlabel('False Positive Rate')

                ax.set\_ylabel('True Positive Rate')

                ax.legend(loc="lower right")

                ax.set\_title('Многоклассовая ROC-кривая')

            else:

                # Бинарная классификация

                fpr, tpr, \_ = roc\_curve(y\_test, y\_proba[:, 1])

                roc\_auc = auc(fpr, tpr)

                ax.plot(fpr, tpr, color='darkorange', lw=2,

                        label=f'ROC кривая (AUC = {roc\_auc:.2f})')

                ax.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

                ax.set\_xlim([0.0, 1.0])

                ax.set\_ylim([0.0, 1.05])

                ax.set\_xlabel('False Positive Rate')

                ax.set\_ylabel('True Positive Rate')

                ax.legend(loc="lower right")

                ax.set\_title('ROC-кривая')

            st.pyplot(fig)

        else:

            st.warning("ROC-кривая недоступна: модель не возвращает вероятности")

    # Важность признаков (если доступно)

    if hasattr(model, 'feature\_importances\_'):

        st.subheader("Важность признаков")

        importances = pd.Series(

            model.feature\_importances\_,

            index=X.columns

        ).sort\_values(ascending=False)

        fig, ax = plt.subplots(figsize=(10, 6))

        importances.plot.bar(ax=ax)

        ax.set\_title("Важность признаков")

        ax.set\_ylabel("Важность")

        plt.xticks(rotation=45, ha='right')

        st.pyplot(fig)

# Инструкция по использованию

st.sidebar.header("ℹ️ Инструкция")

st.sidebar.info("""

1. Выберите или загрузите данные

2. Выберите модель для обучения

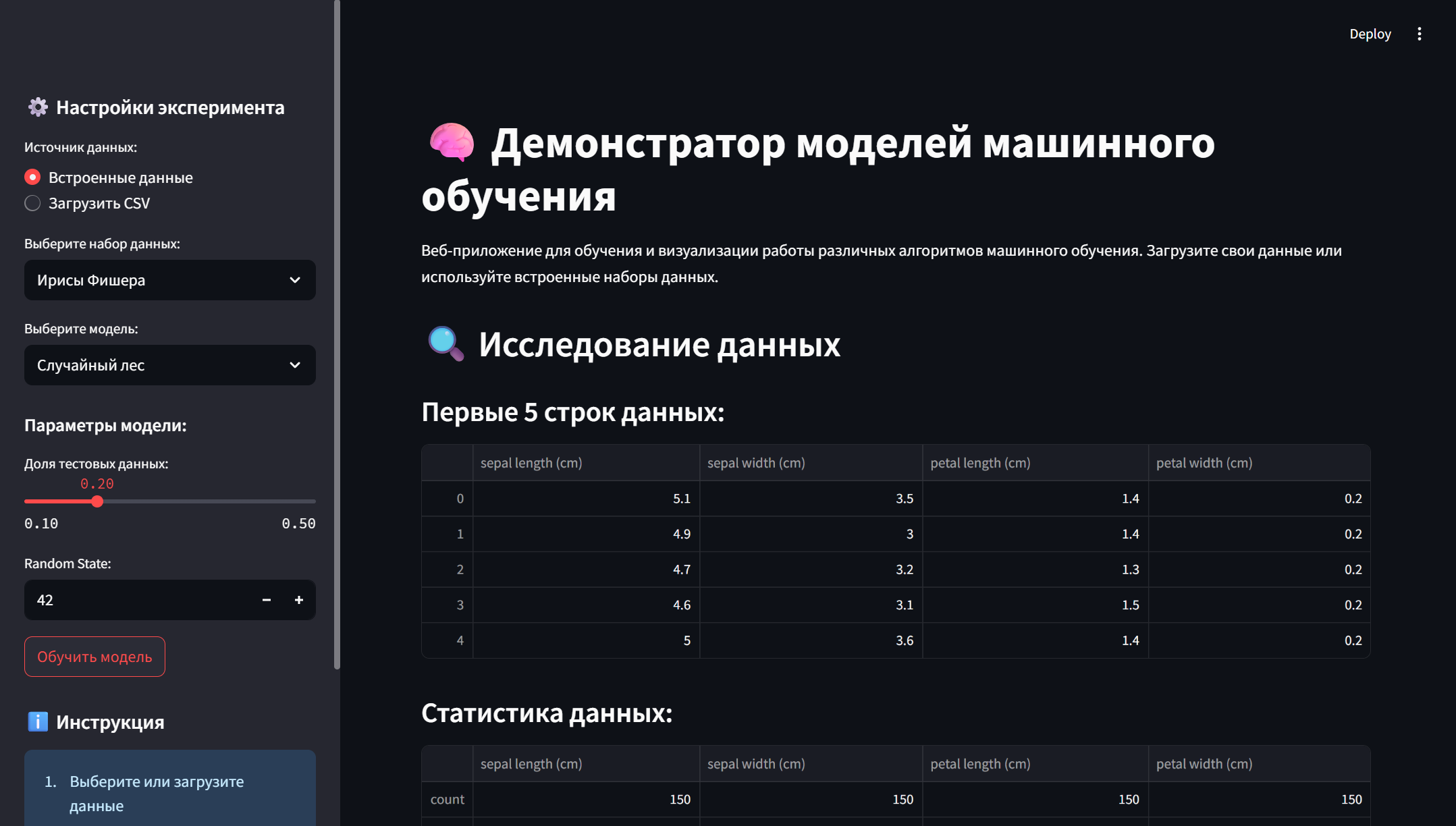
3. Настройте параметры

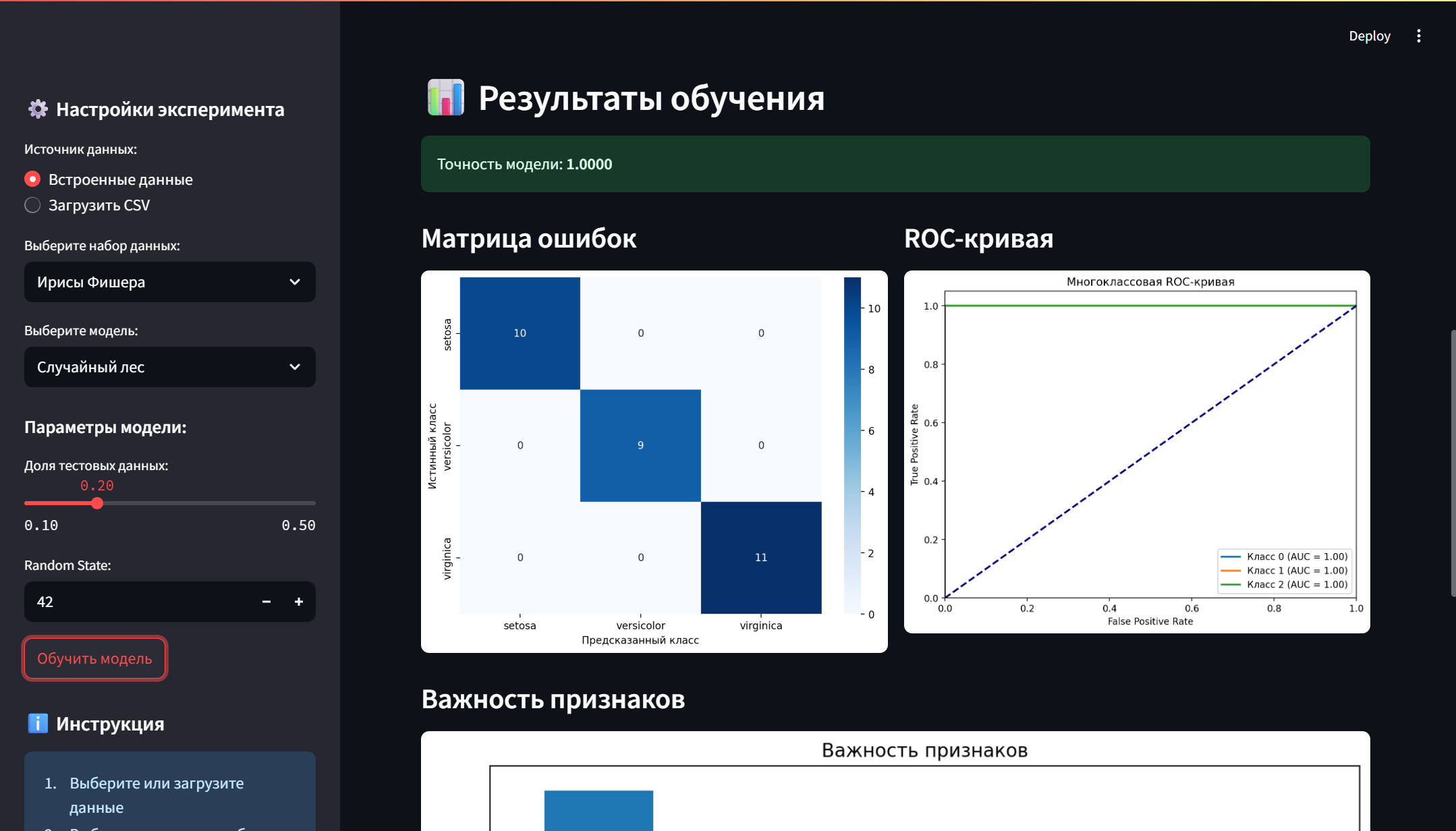
4. Нажмите "Обучить модель"

5. Анализируйте результаты

""")

**Результат работы программы**

****

****