МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э.Баумана

Отчет по лабораторной работе №6
по курсу «Технологии машинного обучения»

Создание веб-приложения для демонстрации моделей машинного обучения.

Подготовил
Ионов С.А.
ИУ5-62Б

1) Описание задания

Разработайте макет веб-приложения, предназначенного для анализа данных.

Макет должен быть реализован для одной модели машинного обучения. Макет должен позволять:

- задавать гиперпараметры алгоритма,
- производить обучение,
- осуществлять просмотр результатов обучения, в том числе в виде графиков.

2) Текст программы

```
 import streamlit as st
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, learning_curve
from sklearn.metrics import plot_confusion_matrix, classification_report, plot_roc_curve,
f1_score
from sklearn.ensemble import RandomForestClassifier


def load_data():
    '''
    Загрузка данных
    '''
    data = pd.read_csv("data\data.csv")
    return data


TEST_SIZE = 0.3
RANDOM_STATE = 0


def preprocess_data(data):
    data.drop(columns=['id', 'Unnamed: 32'], inplace=True)
    data['diagnosis'] = data['diagnosis'].map({'M': 1, 'B': 0})
    data = data.drop(columns=['fractal_dimension_mean', 'texture_se', 'symmetry_se'])
    data_X = data.drop(columns='diagnosis')
    data_y = data['diagnosis']
    data_X_train, data_X_test, data_y_train, data_y_test = train_test_split(data_X, \
                                                          data_y, test_size=TEST_SIZE, \
                                                          random_state=RANDOM_STATE)
    return data_X_train, data_X_test, data_y_train, data_y_test
```

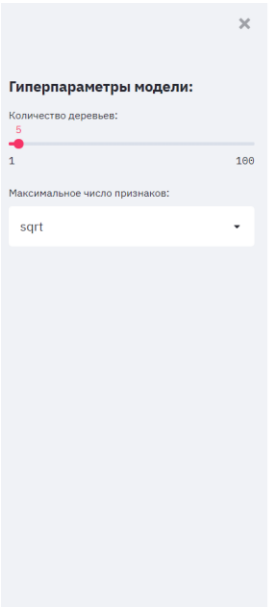
```
def print_metrics(X_train, Y_train, X_test, Y_test, clf):
    clf.fit(X_train, Y_train)
    target = clf.predict(X_test)
    ret = f1_score(Y_test, target)
    st.write(f'F1-мера: {ret}')
    fig1, ax1 = plt.subplots()
    plot_roc_curve(clf, X_test, Y_test, ax = ax1)
    ax1.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
            label='Chance', alpha=.8)
    ax1.set(xlim=[-0.05, 1.05], ylim=[-0.05, 1.05],
           title="Receiver operating characteristic")
    ax1.legend(loc="lower right")
    st.pyplot(fig1)
    fig2, ax2 = plt.subplots(figsize=(10, 5))
    plot_confusion_matrix(clf, X_test, Y_test, ax=ax2,
                          display_labels=['B', 'M'],
                          cmap=plt.cm.Blues,
                          normalize='true')
    ax2.set(title="Confusion matrix")
    st.pyplot(fig2)
    rep = classification_report(Y_test, target, target_names=['B', 'M'], output_dict=True)
    df = pd.DataFrame(rep).transpose()
    st.markdown("<h2 style='text-align: center; font-weight: bold'>Report</h1>",
unsafe_allow_html=True)
    st.table(df)
    return ret


def plot_learning_curve(data_X, data_y, clf):
    train_sizes, train_scores, test_scores = learning_curve(estimator=clf, scoring='f1', X=data_X,
y=data_y,
                                                            train_sizes=np.linspace(0.1, 1.0, 10), cv=5)
    train_mean = np.mean(train_scores, axis=1)
    train_std = np.std(train_scores, axis=1)
    test_mean = np.mean(test_scores, axis=1)
    test_std = np.std(test_scores, axis=1)
    fig = plt.figure(figsize=(7,5))
    plt.plot(train_sizes, train_mean, color='blue', marker='o', markersize=5,
label=f'тренировочная f1-мера')
    plt.fill_between(train_sizes, train_mean+train_std, train_mean-train_std, alpha=0.15,
color='blue')
    plt.plot(train_sizes, test_mean, color='green', linestyle='--', marker='s', markersize=5,
             label=f'проверочная f1-мера')
    plt.fill_between(train_sizes, test_mean+test_std, test_mean-test_std, alpha=0.15,
color='green')
    plt.grid()
    plt.legend(loc='lower right')
    plt.xlabel('Число тренировочных образцов')
```

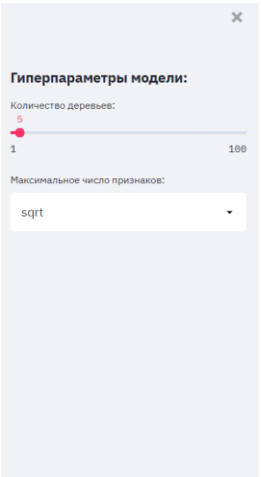
```
    plt.ylabel('f1-мера')
    st.pyplot(fig)


if __name__ == '__main__':

    st.title('Демонстрация метода случайного леса')
    data = load_data()
    data_X_train, data_X_test, data_y_train, data_y_test = preprocess_data(data)
    if st.checkbox('Показать данные'):
        st.write(data.head())
    if st.checkbox('Показать корреляционную матрицу'):
        fig_corr, ax = plt.subplots(figsize=(20, 20))
        sns.heatmap(data.corr(), annot=True, fmt='.2f')
        st.pyplot(fig_corr)
    st.sidebar.subheader('Гиперпараметры модели:')
    n_est = st.sidebar.slider('Количество деревьев: ', min_value=1, max_value=100, value=5,
step=1)
    max_features = st.sidebar.selectbox('Максимальное число признаков:', ('sqrt', 'log2'))
    rfc = RandomForestClassifier(n_estimators=n_est, max_features=max_features,
random_state=RANDOM_STATE)
    res_rfc = print_metrics(data_X_train, data_y_train, data_X_test, data_y_test, rfc)
    data_X = pd.concat([data_X_train, data_X_test])
    data_y = pd.concat([data_y_train, data_y_test])
    plot_learning_curve(data_X, data_y, rfc)
```

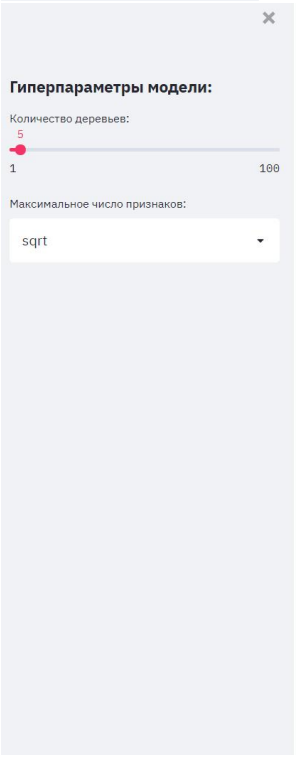
3) Экранные формы с примерами выполнения программы

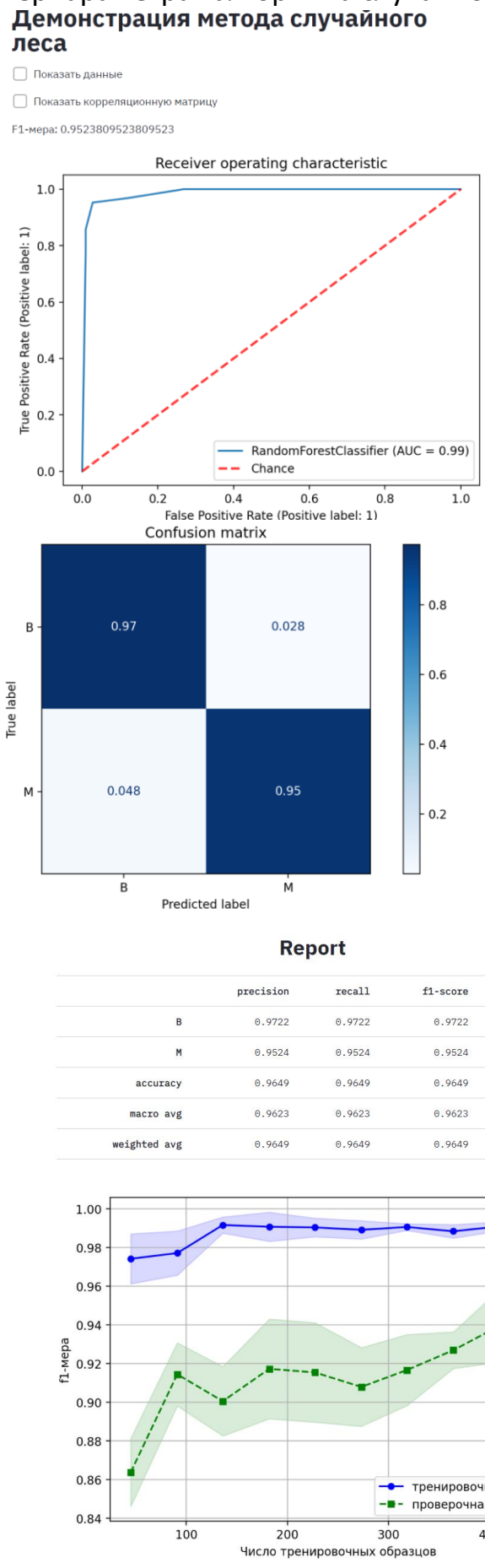
- Для лабораторной работы был выбран датасет Breast Cancer Wisconsin, который был использован в предыдущей лабораторной работе. Будем решать задачу классификации для предсказания наличия рака молочной железы

- Данный макет веб-приложения позволяет производить обучение модели при разных гиперпараметрах алгоритма случайного леса:









- При изменении гиперпараметров получаем следующий вывод: