Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчет по лабораторной работе № 6
«Создание веб-приложения для демонстрации моделей машинного обучения»

Выполнил:

студент группы ИУ5-65Б

Герасименко А.В.

Подпись и дата:

Проверил:

преподаватель каф. ИУ5

Гапанюк Ю.Е.

Подпись и дата:

г. Москва, 2021 г.

Лабораторная работа №6

Создание веб-приложения для демонстрации моделей машинного обучения

Цель лабораторной работы

Изучение возможностей демонстрации моделей машинного обучения с помощью веб-приложений.

Задание

Разработайте макет веб-приложения, предназначенного для анализа данных.

Вариант 1. 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.

```
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, accuracy_score, roc_curve,
roc_auc_score, f1_score
from sklearn.preprocessing import MinMaxScaler
from catboost import Pool, CatBoostClassifier
# Запуск приложения streamlit run
C:/Users/User/Desktop/TMO_NST/Lab6/Lab6.py [ARGUMENTS]
def load():
    col_list = ['Pelvic_incidence',
                'Pelvic_tilt',
                'Lumbar_lordosis_angle',
                'Sacral_slope',
```

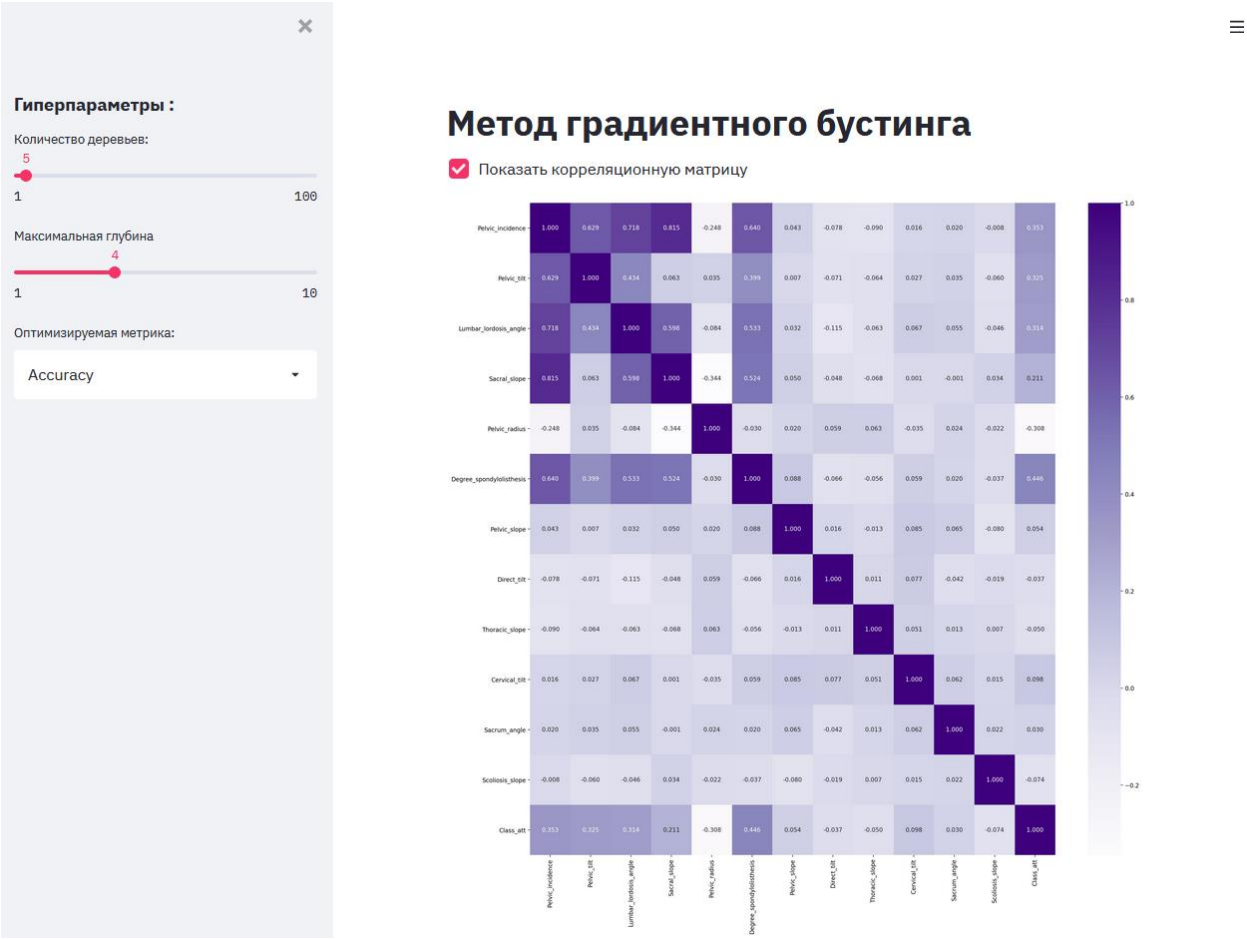
```
                'Pelvic_radius',
                'Degree_spondylolisthesis',
                'Pelvic_slope',
                'Direct_tilt',
                'Thoracic_slope',
                'Cervical_tilt',
                'Sacrum_angle',
                'Scoliosis_slope',
                'Class_att',
                'To_drop']
    data = pd.read_csv('Dataset_spine.csv', names=col_list, header=1, sep=",")
    data.drop('To_drop', axis=1, inplace=True)
    return data
# Готовим данные к ML
def preprocess_data(data):
    scale_cols = ['Pelvic_incidence',
                  'Pelvic_tilt',
                  'Lumbar_lordosis_angle',
                  'Sacral_slope',
                  'Pelvic_radius',
                  'Degree_spondylolisthesis',
                  'Pelvic_slope',
                  'Direct_tilt',
                  'Thoracic_slope',
                  'Cervical_tilt',
                  'Sacrum_angle',
                  'Scoliosis_slope']
    sc1 = MinMaxScaler()
    sc1_data = sc1.fit_transform(data[scale_cols])
    for i in range(len(scale_cols)):
```

```
        data[scale_cols[i]] = sc1_data[:, i]
    data['Class_att'] = data['Class_att'].map({'Abnormal': 1, 'Normal': 0})
    # Разделим данные на целевой столбец и признаки
    X = data.drop("Class_att", axis=1)
    Y = data["Class_att"]
    # С использованием метода train_test_split разделим выборку на
обучающую и тестовую
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25,
random_state=1)
    return X_train, X_test, Y_train, Y_test
# Отрисовка графика ROC_CURVE
def draw_roc_curve(y_true, y_score, ax, pos_label=1, average='micro'):
    fpr, tpr, thresholds = roc_curve(y_true, y_score,
                                     pos_label=pos_label)
    roc_auc_value = roc_auc_score(y_true, y_score, average=average)
    # plt.figure()
    lw = 2
    ax.plot(fpr, tpr, color='darkorange',
            lw=lw, label='ROC curve (area = %0.2f)' % roc_auc_value)
    ax.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    ax.set_xlim([0.0, 1.0])
    ax.set_xlim([0.0, 1.05])
    ax.set_xlabel('False Positive Rate')
    ax.set_ylabel('True Positive Rate')
    ax.set_title('Receiver operating characteristic')
    ax.legend(loc="lower right")
# Вывод метрик ML
def print_metrics(X_train, Y_train, X_test, Y_test, clf):
    clf.fit(X_train, Y_train)
    target = clf.predict(X_test)
```

```
    test_score = accuracy_score(Y_test, target)
    roc_res = clf.predict_proba(X_test)
    roc_auc = roc_auc_score(Y_test, roc_res[:, 1])
    f1_test_score = f1_score(Y_test, target)
    st.write(f"accuracy (точность): {test_score}")
    st.write(f"f1 метрика: {f1_test_score}")
    st.write(f"ROC AUC: {roc_auc}")
    fig1, ax1 = plt.subplots()
    draw_roc_curve(Y_test, roc_res[:, 1], ax1)
    st.pyplot(fig1)
    fig2, ax2 = plt.subplots(figsize=(10, 5))
    plot_confusion_matrix(clf, X_test, Y_test, ax=ax2, display_labels=['1', '0'],
cmap = 'Purples', normalize='true')
    ax2.set(title="Confusion matrix")
    st.pyplot(fig2)
    return test_score
# Вывод кривой обучения
def plot_learning_curve(data_X, data_y, clf, name='accuracy', scoring='accuracy'):
    train_sizes, train_scores, test_scores = learning_curve(estimator=clf,
scoring=scoring, 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 {name}-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 {name}-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(f'{name}-мера')
    st.pyplot(fig)
if __name__ == '__main__':
    st.title('Метод градиентного бустинга')
    data = load()
    data_X_train, data_X_test, data_y_train, data_y_test = preprocess_data(data)
    # Будем показывать матрицу только по запросу, чттобы не тормозить
процесс
    if st.checkbox('Показать корреляционную матрицу'):
        fig_corr, ax = plt.subplots(figsize=(20, 20))
        sns.heatmap(data.corr(), annot=True, cmap = 'Purples', fmt='.3f')
        st.pyplot(fig_corr)
    # Выбор гиперпараметров в сайдбаре
    st.sidebar.subheader('Гиперпараметры :')
    estimators = st.sidebar.slider('Количество деревьев: ', min_value=1,
max_value=100, value=5, step=1)
    max_depth = st.sidebar.slider('Максимальная глубина', min_value=1,
max_value=10, value=4, step=1)
    eval_metric = st.sidebar.selectbox('Оптимизируемая метрика:', ('Accuracy',
'F1', 'AUC'))
    # Вывод результатов
    translation_dict = {'Accuracy': 'accuracy', 'F1': 'f1', 'AUC': 'roc_auc'}
```

```
    gd = CatBoostClassifier(n_estimators=estimators, max_depth=max_depth,
eval_metric=eval_metric, random_state=1)
    result = print_metrics(data_X_train, data_y_train, data_X_test, data_y_test, gd)
    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, gd,
name=translation_dict.get(eval_metric), scoring=translation_dict.get(eval_metric))
    #  Показать данные
    if st.checkbox('Показать первые 10 строк датасета "Dataset_spine"'):
        st.write(data.head(10))
```



Гиперпараметры :

Количество деревьев:

5

1 100

Максимальная глубина

4

1 10

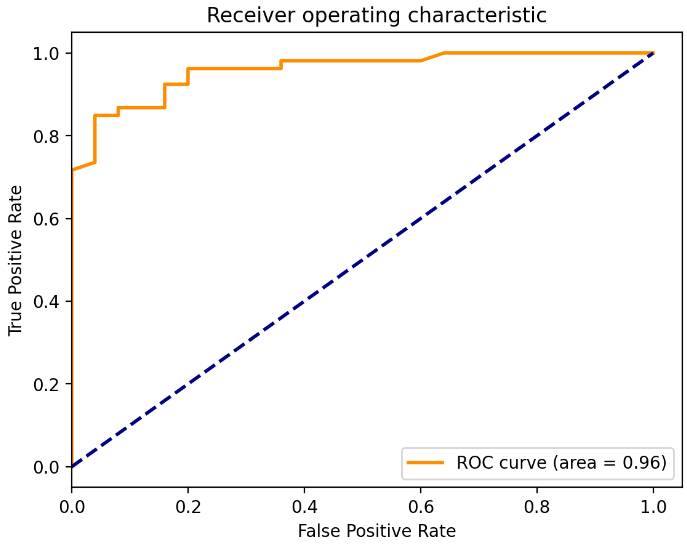
Оптимизируемая метрика:

Accuracy

accuracy (точность): 0.8846153846153846

f1 метрика: 0.9158878504672898

ROC AUC: 0.9584905660377359



Гиперпараметры :

Количество деревьев:

5

1 100

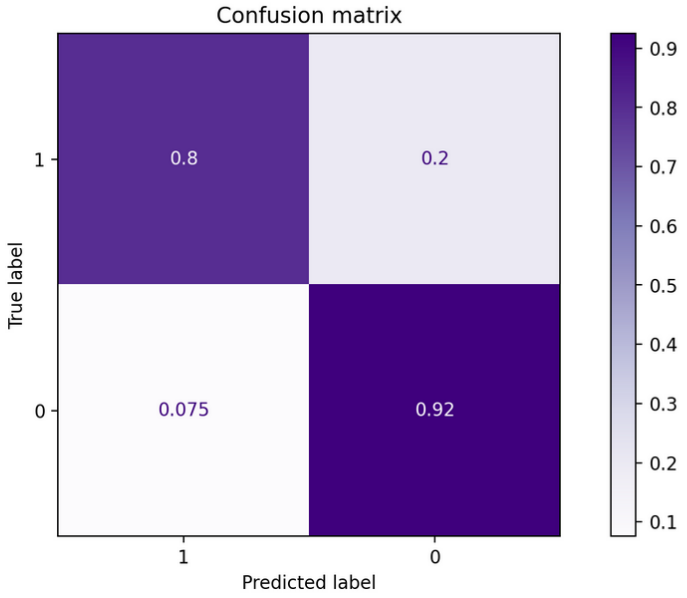
Максимальная глубина

4

1 10

Оптимизируемая метрика:

Accuracy



Гиперпараметры :

Количество деревьев:

5

1 100

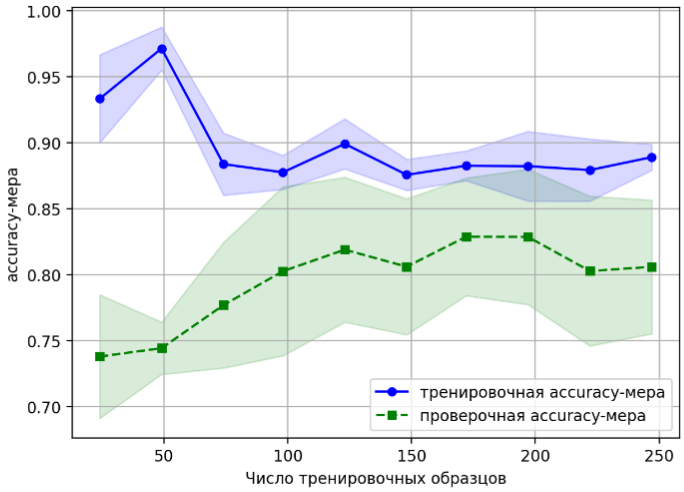
Максимальная глубина

4

1 10

Оптимизируемая метрика:

Accuracy



Показать первые 10 строк датасета "Dataset_spine"

	Pelvic_incidence	Pelvic_tilt	Lumbar_lordosis_angle	Sacral_slope	Pelvic_
0	0.1245	0.2968	0.0986	0.1446	
1	0.4117	0.5139	0.3230	0.3077	
2	0.4162	0.5574	0.2713	0.2894	
3	0.2273	0.2895	0.1281	0.2470	
4	0.1360	0.3657	0.0996	0.1199	
5	0.2632	0.4004	0.2073	0.2240	
6	0.1854	0.3092	0.1346	0.1966	
7	0.1702	0.3588	0.2568	0.1563	
8	0.1016	0.2066	0.2501	0.1694	
9	0.2272	0.3500	0.1551	0.2156	