

Московский государственный технический университет им. Н.Э. Баумана  
Факультет «Информатика и системы управления»  
Кафедра «Системы обработки информации и управления»



**Отчет**  
**Лабораторная работа №6**  
**По курсу Технологии машинного обучения»**

**ИСПОЛНИТЕЛЬ:**

Кожуро Б.Е.  
Группа ИУ5-65Б

**ПРЕПОДАВАТЕЛЬ:**

Гапанюк Ю.Е.

"\_\_" \_\_\_\_\_ 2021 г.

Москва 2021

---

## 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.

## 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, accuracy_score, roc_curve,
roc_auc_score, f1_score
from sklearn.preprocessing import MinMaxScaler
from catboost import Pool, CatBoostClassifier

def load():
    data = pd.read_csv("heart.csv")
    return data

TEST_SIZE = 0.3
RANDOM_STATE = 0

#Готовим данные к ML
def preprocess_data(data):
    scale_cols = ['age', 'trestbps', 'chol', 'thalach', 'ca', 'thal']
    new_cols = []
    scl = MinMaxScaler()
    scl_data = scl.fit_transform(data[scale_cols])
    for i in range(len(scale_cols)):
        data[scale_cols[i]] = scl_data[:, i]
    data_X = data.drop(columns='target')
    data_y = data['target']
    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

#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'],
                          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)

```

```

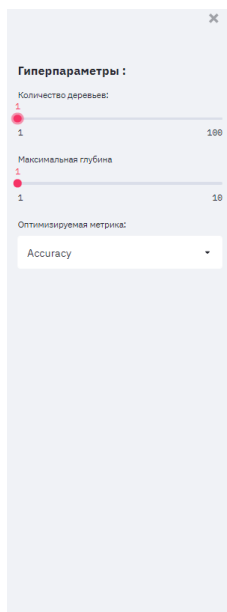
# Будем показывать матрицу только по запросу, чтобы не тормозить процесс
if st.checkbox('Показать корреляционную матрицу'):
    fig_corr, ax = plt.subplots(figsize=(20, 20))
    sns.heatmap(data.corr(), annot=True, fmt='.2f')
    st.pyplot(fig_corr)

# 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 данные'):
    st.write(data.head(10))

```

## Экранные формы:



**Гиперпараметры :**

Количество деревьев: 1

Максимальная глубина: 1

Оптимизируемая метрика: Accuracy

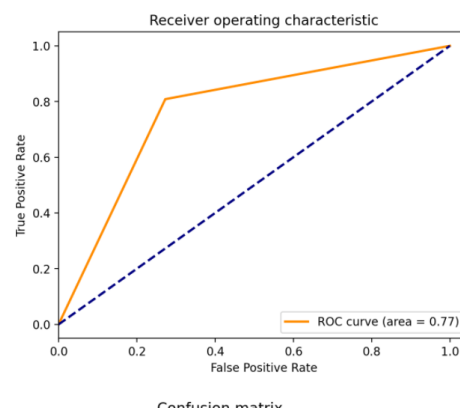
### Метод градиентного бустинга

☐ Показать корреляционную матрицу

accuracy (точность): 0.7692307692307693

f1 метрика: 0.7835051546391754

ROC AUC: 0.7678916827852998



Гиперпараметры :

Количество деревьев:

1

100

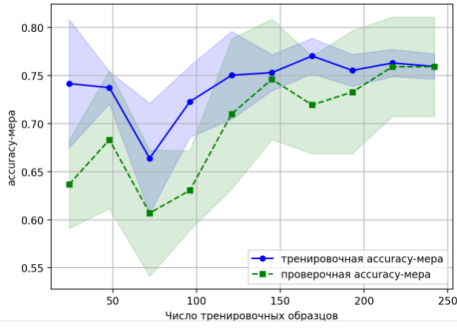
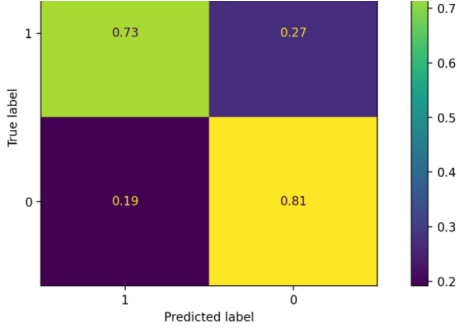
Максимальная глубина

1

10

Оптимизируемая метрика:

Ассигасу



Гиперпараметры :

Количество деревьев:

26

100

Максимальная глубина

4

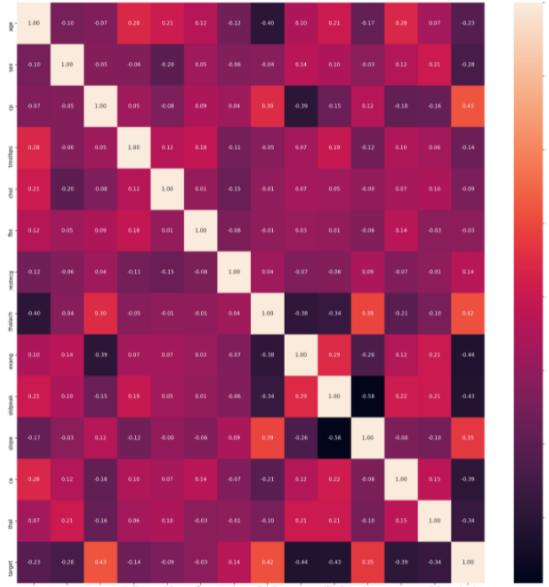
10

Оптимизируемая метрика:

Ассигасу

## Метод градиентного спуска

☒ Показать корреляционную матрицу



Гиперпараметры :

Количество деревьев:

26

100

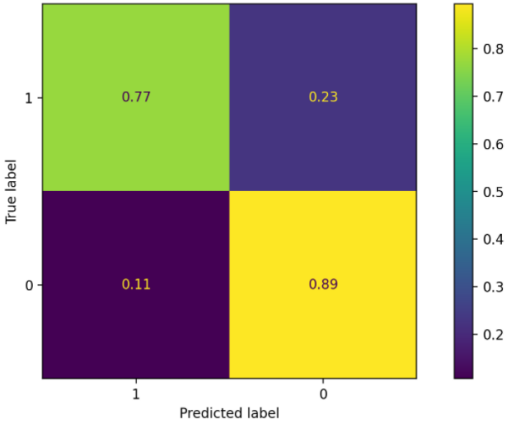
Максимальная глубина

4

10

Оптимизируемая метрика:

Ассигасу



Гиперпараметры :

Количество деревьев:

28

1100

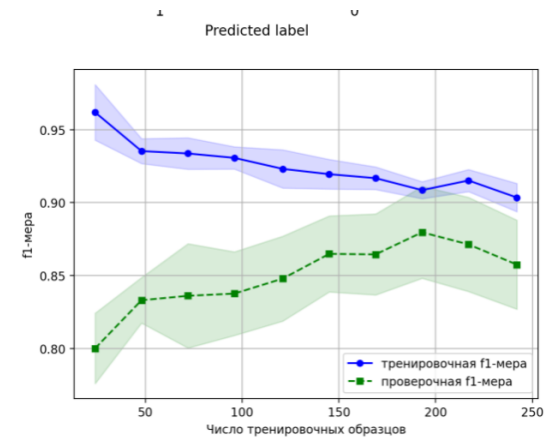
Максимальная глубина

4

110

Оптимизируемая метрика:

F1



☒ Показать данные

	age	sex	cp	trestbws	chol	fbs	restecg	thalach	exang	oldpeak
0	0.7083	1	3	0.4811	0.2443	1	0	0.6031	0	2.300
1	0.1667	1	2	0.3396	0.2831	0	1	0.8855	0	3.600
2	0.2500	0	1	0.3396	0.1781	0	0	0.7719	0	1.400
3	0.5625	1	1	0.2453	0.2511	0	1	0.8168	0	0.800
4	0.5833	0	0	0.2453	0.5205	0	1	0.7023	1	0.600
5	0.5033	1	0	0.4340	0.1507	0	1	0.5578	0	0.400
6	0.5625	0	1	0.4340	0.3836	0	0	0.6260	0	1.300
7	0.3125	1	1	0.2453	0.3128	0	1	0.7786	0	
8	0.4792	1	2	0.7358	0.1667	1	1	0.6947	0	0.500
9	0.5833	1	2	0.5283	0.0959	0	1	0.7863	0	1.600