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.

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.

Для разработки рекомендуется использовать следующие (или аналогичные) фреймворки:

- streamlit
- gradio
- dash

Текст программы:

```
import streamlit as st
import seaborn as sns
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score
from operator import itemgetter

@st.cache
def load_data():
    '''
    Загрузка данных
    '''
    dataskl = load_breast_cancer()
    data = pd.DataFrame(data= np.c_[dataskl['data'], dataskl['target']],
                        columns= list(dataskl['feature_names']) + ['target'])
    return data, dataskl


@st.cache
def preprocess_data(data_in, dataskl, TSize):
    temp_X = dataskl.data
    temp_y = dataskl.target
    # Чтобы в тесте получилось низкое качество используем только 0,5% данных для
обучения
    X_train, X_test, y_train, y_test = train_test_split(temp_X, temp_y, train_siz
e=TSize, random_state=1)
    return X_train, X_test, y_train, y_test



# Отрисовка ROC-кривой
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)
```

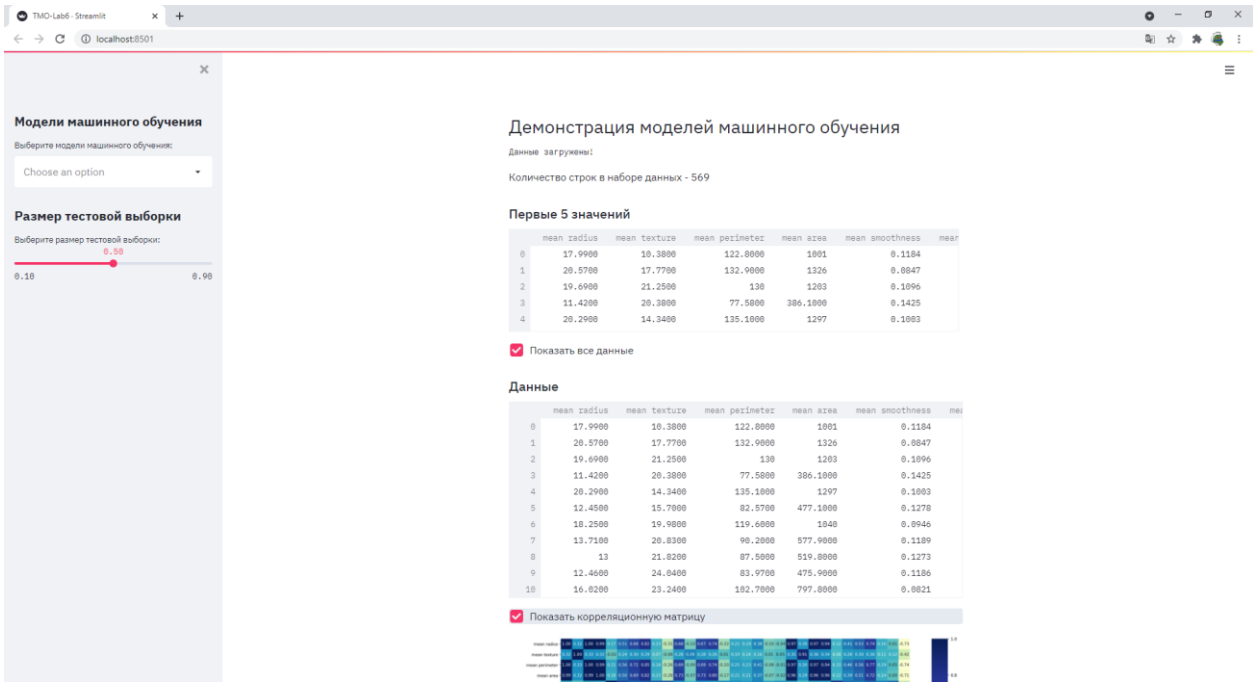
```
    roc_auc_value = roc_auc_score(y_true, y_score, average=average)
    #plt.figure()
    lw = 2
    ax.plot(fpr, tpr, color='darkorange',
             lw=lw, label='ROC curve (area = %0.2f)' % roc_auc_value)
    ax.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    ax.set_xlim([0.0, 1.0])
    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")


def vis_models_quality(array_metric, array_labels, str_header, ax1, i):
    pos = np.arange(len(array_metric))
    rects = ax1.barh(pos, array_metric,
                     align='center',
                     height=0.5,
                     tick_label=array_labels)
    ax1.set_title(str_header)
    for a,b in zip(pos, array_metric):
        plt.text(i, a-0.1, str(round(b,3)), color='black')

def draw_feature_importances(tree_model, X_dataset, figsize=(10,5)):
    """
    Вывод важности признаков в виде графика
    """
    # Сортировка значений важности признаков по убыванию
    list_to_sort = list(zip(X_dataset.columns.values, tree_model.feature_importan
ces_))
    sorted_list = sorted(list_to_sort, key=itemgetter(1), reverse = True)
    # Названия признаков
    labels = [x for x,_ in sorted_list]
    # Важности признаков
    data = [x for _,x in sorted_list]
    # Вывод графика
    fig, ax = plt.subplots(figsize=figsize)
    ind = np.arange(len(labels))
    plt.bar(ind, data)
    plt.xticks(ind, labels, rotation='vertical')
    # Вывод значений
    for a,b in zip(ind, data):
        plt.text(a-0.06, b+0.01, str(round(b,2)))
    st.pyplot(fig)
    return labels, data

models_list = ['LogR', 'KNN', 'SVC', 'Tree', 'RF', 'GB']

st.header('Демонстрация моделей машинного обучения')
st.sidebar.header('Модели машинного обучения')
```

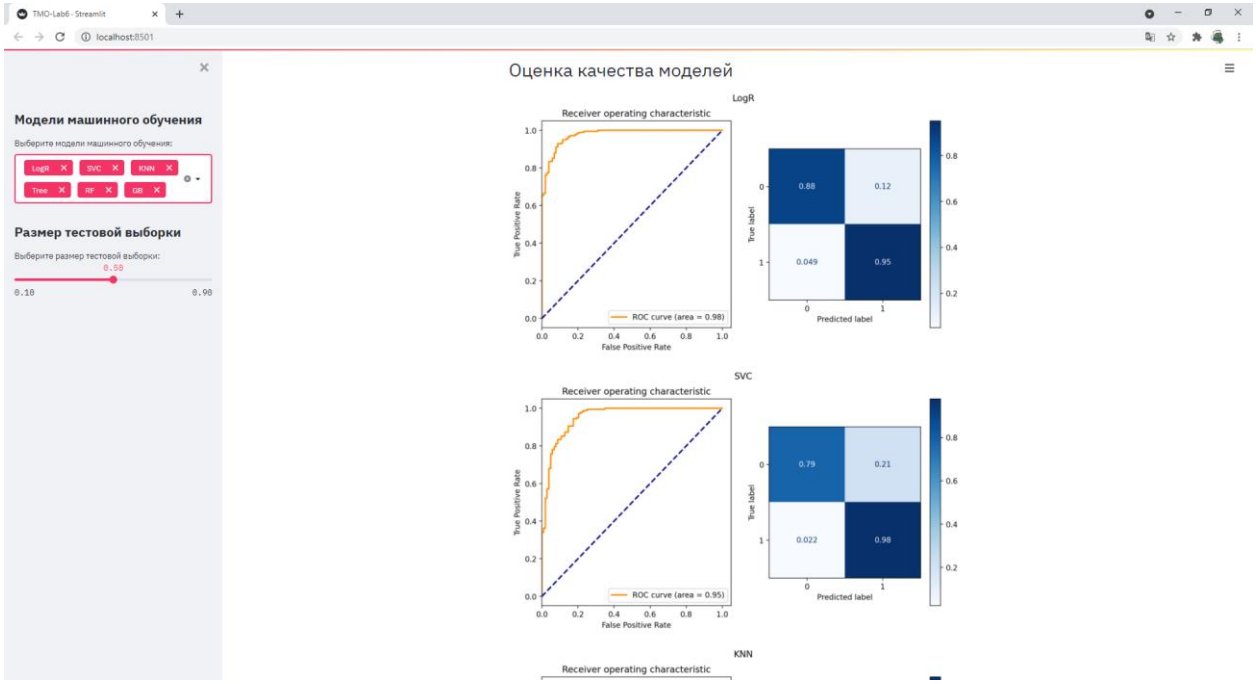
```
models_select = st.sidebar.multiselect('Выберите модели машинного обучения:', mod
els_list)
st.sidebar.header('Размер тестовой выборки')
TSize = st.sidebar.slider('Выберите размер тестовой выборки:', min_value=0.1, max
_value=0.9, value=0.5, step=0.1)

data_load_state = st.text('Загрузка данных...')
data, dataskl = load_data()
X_train, X_test, y_train, y_test = preprocess_data(data, dataskl, TSize)
data_load_state.text('Данные загружены!')
#Количество записей
data_len = data.shape[0]
st.write('Количество строк в наборе данных - {}'.format(data_len))

st.subheader('Первые 5 значений')
st.write(data.head())

if st.checkbox('Показать все данные'):
    st.subheader('Данные')
    st.write(data)

if st.checkbox('Показать корреляционную матрицу'):
    fig, ax = plt.subplots(figsize=(20,15))
    sns.heatmap(data.corr(method='pearson'), ax=ax, annot=True, fmt='.2f', cmap="
YlGnBu")
    st.pyplot(fig)

cv_knn = st.slider('Количество ближайших соседей:', min_value=1, max_value=data_l
en//2, value=5, step=1)

# Модели
clas_models = {'LogR': LogisticRegression(max_iter=10000),
               'KNN':KNeighborsClassifier(n_neighbors=cv_knn),
               'SVC':SVC(probability=True),
               'Tree':DecisionTreeClassifier(),
               'RF':RandomForestClassifier(),
               'GB':GradientBoostingClassifier()}


@st.cache(suppress_st_warning=True)
def print_models(models_select, X_train, X_test, y_train, y_test, dataskl):
    current_models_list = []
    roc_auc_list = []
    precision_list = []
    recall_list = []
    accuracy_list =[]
    for model_name in models_select:
        model = clas_models[model_name]
        model.fit(X_train, y_train)
        # Предсказание значений
        Y_pred = model.predict(X_test)
```

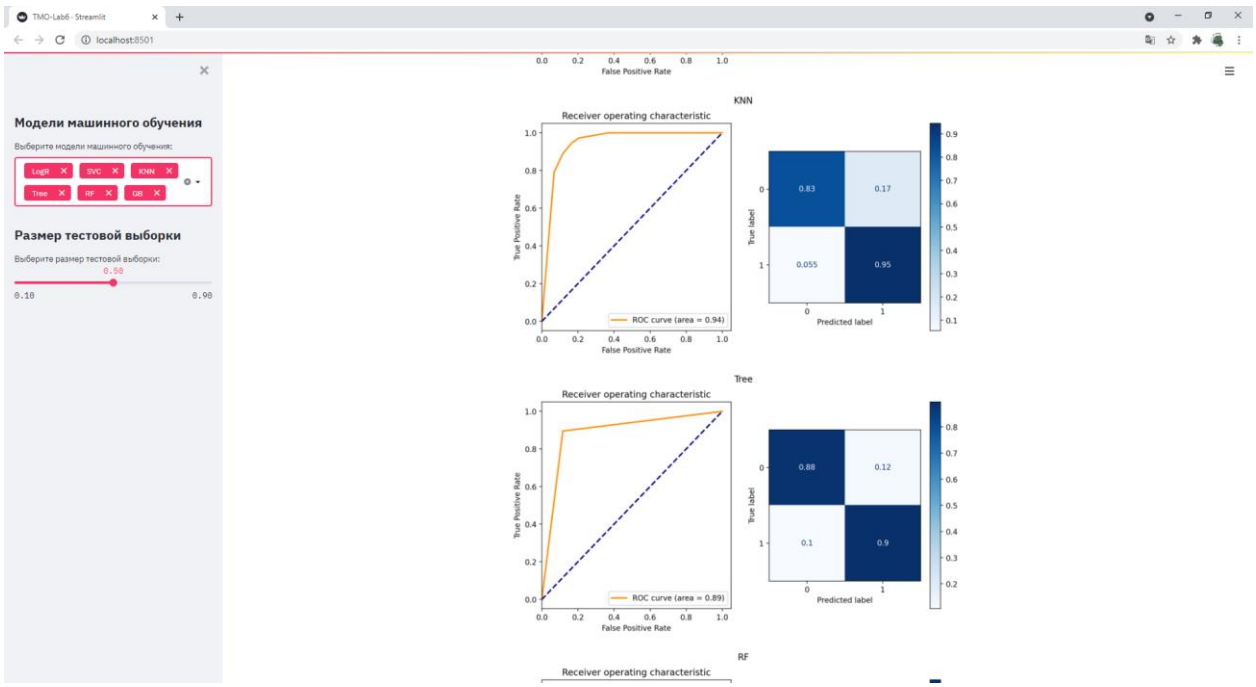
```
        # Предсказание вероятности класса "1" для roc auc
        Y_pred_proba_temp = model.predict_proba(X_test)
        Y_pred_proba = Y_pred_proba_temp[:,1]

        roc_auc = roc_auc_score(y_test, Y_pred)
        precision = precision_score(y_test, Y_pred)
        recall = recall_score(y_test, Y_pred)
        accuracy = accuracy_score(y_test, Y_pred)

        current_models_list.append(model_name)
        roc_auc_list.append(roc_auc)
        precision_list.append(precision)
        recall_list.append(recall)
        accuracy_list.append(accuracy)

        #Отрисовка ROC-кривых
        fig, ax = plt.subplots(ncols=2, figsize=(10,5))
        draw_roc_curve(y_test, Y_pred_proba, ax[0])
        plot_confusion_matrix(model, X_test, y_test, ax=ax[1],
                              display_labels=['0','1'],
                              cmap=plt.cm.Blues, normalize='true')
        fig.suptitle(model_name)
        st.pyplot(fig)

    if len(current_models_list)>0:
        fig, ax1 = plt.subplots(ncols=2, figsize=(8,2))
        vis_models_quality(accuracy_list, current_models_list, 'accuracy_score',
ax1[0], -0.7)
        vis_models_quality(precision_list, current_models_list, 'precision_score'
, ax1[1], 0.2)
        st.pyplot(fig)

        fig, ax1 = plt.subplots(ncols=2, figsize=(8,2))
        vis_models_quality(recall_list, current_models_list, 'recall_score', ax1[
0], -0.7)
        vis_models_quality(roc_auc_list, current_models_list, 'roc-
auc', ax1[1], 0.2)
        st.pyplot(fig)

        temp_d = {'accuracy_score': accuracy_list, 'precision_score': precision_l
ist, 'recall_score': recall_list, 'roc-auc': roc_auc_list}
        temp_df = pd.DataFrame(data=temp_d, index=current_models_list)
        st.write(temp_df)

    if "RF" in current_models_list:
        st.header('Важность признаков по RF')
        dataskl_x_ds = pd.DataFrame(data=dataskl['data'], columns=dataskl['featur
e_names'])
        dataskl_rf_cl = RandomForestClassifier(random_state=1)
        dataskl_rf_cl.fit(dataskl_x_ds, dataskl.target)
        _,_ = draw_feature_importances(dataskl_rf_cl, dataskl_x_ds)
```

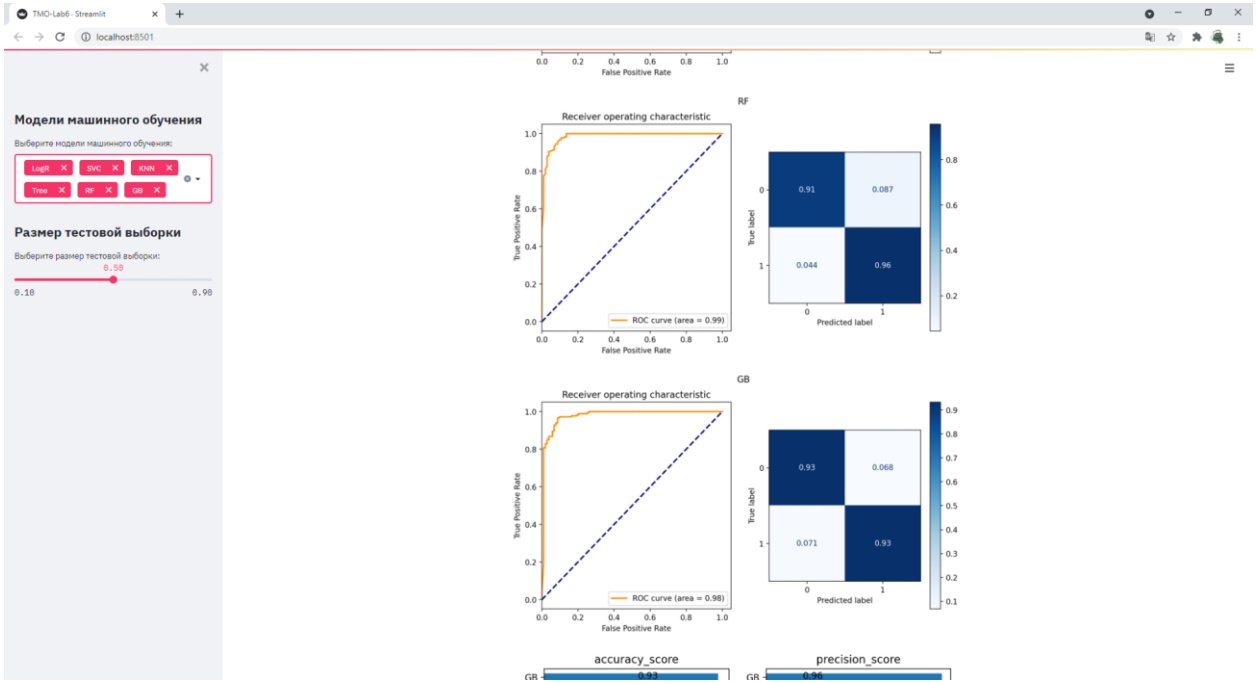
```
    if "GB" in current_models_list:
        st.header('Важность признаков по GB')
        dataskl_x_ds = pd.DataFrame(data=dataskl['data'], columns=dataskl['featur
e_names'])
        dataskl_gb_cl = GradientBoostingClassifier(random_state=1)
        dataskl_gb_cl.fit(dataskl_x_ds, dataskl.target)
        _,_ = draw_feature_importances(dataskl_gb_cl, dataskl_x_ds)

    if "Tree" in current_models_list:
        st.header('Важность признаков по Tree')
        dataskl_x_ds = pd.DataFrame(data=dataskl['data'], columns=dataskl['featur
e_names'])
        dataskl_tree_cl = DecisionTreeClassifier(random_state=1)
        dataskl_tree_cl.fit(dataskl_x_ds, dataskl.target)
        _,_ = draw_feature_importances(dataskl_tree_cl, dataskl_x_ds)


st.header('Оценка качества моделей')
print_models(models_select, X_train, X_test, y_train, y_test, dataskl)
```

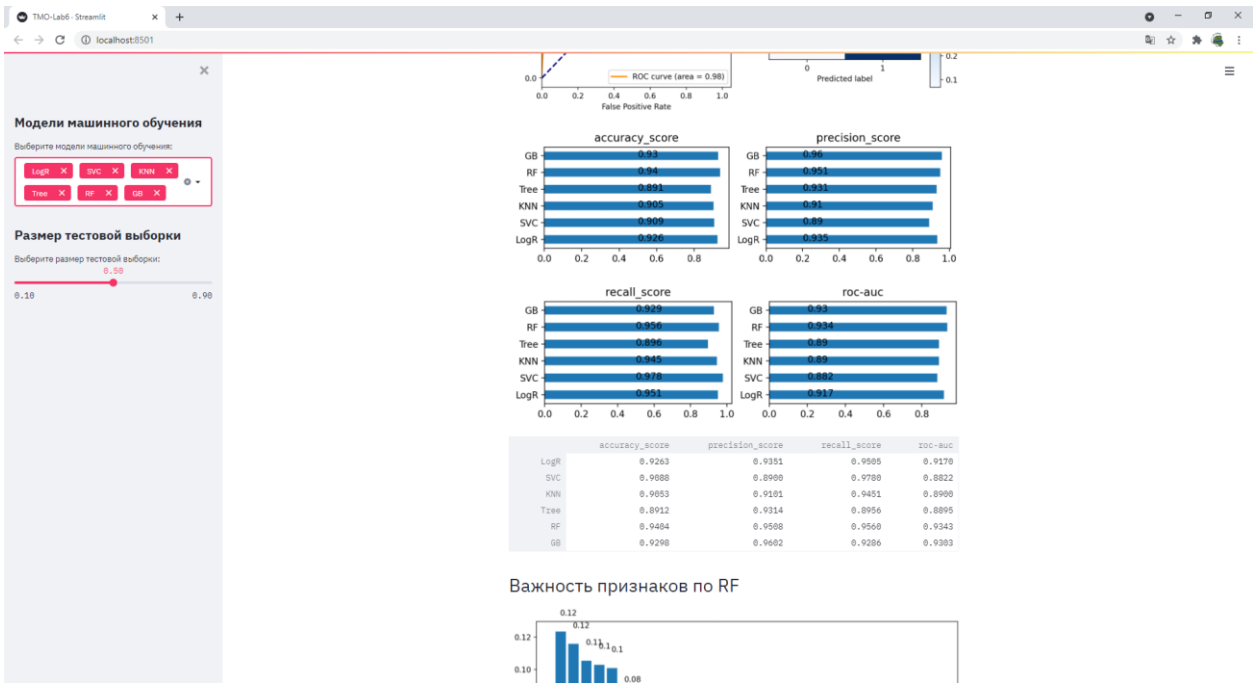
Экранные формы с примерами выполнения программы:




TMO-Lab6 - Streamlit
localhost:8501
Модели машинного обучения
Выберите модели машинного обучения:
LogR
SVC
KNN
Tree
RF
GB
Размер тестовой выборки
Выберите размер тестовой выборки:
0.50
0.10
0.90
Оценка качества моделей
LogR
Receiver operating characteristic
True Positive Rate
False Positive Rate
ROC curve (area = 0.98)
True label
Predicted label
0.88
0.12
0.049
0.95
SVC
Receiver operating characteristic
ROC curve (area = 0.95)
0.79
0.21
0.022
0.98
KNN
Receiver operating characteristic

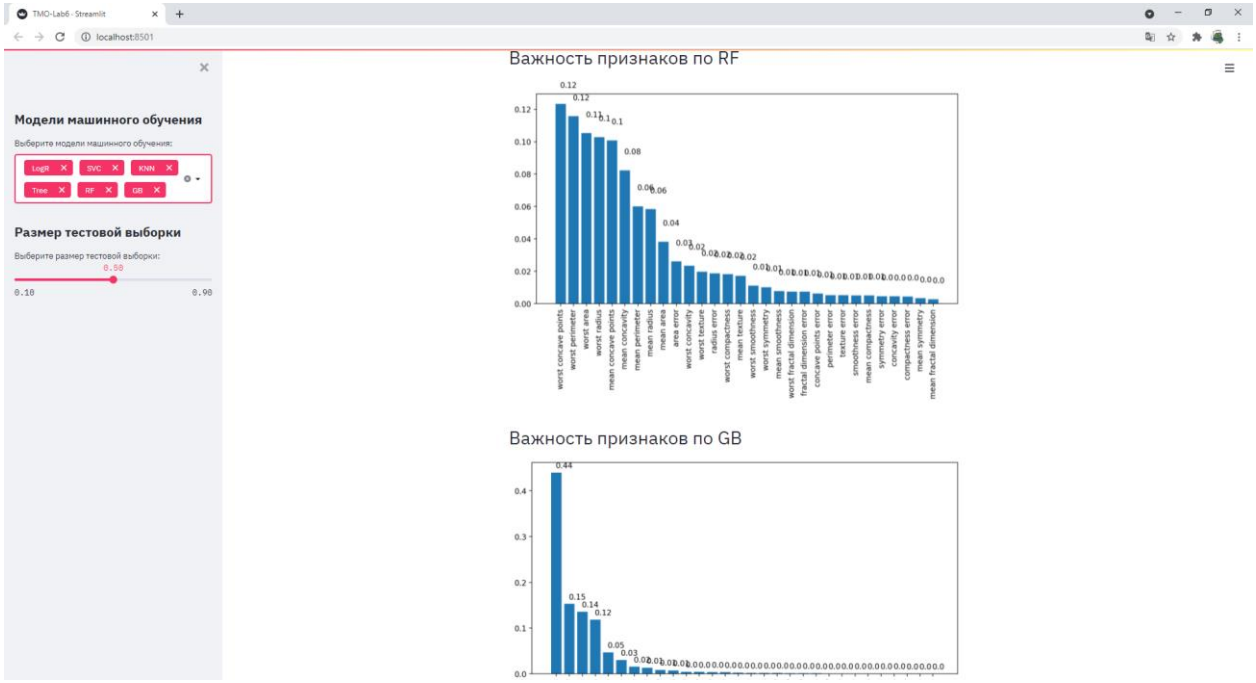

TMO-Lab6 - Streamlit
localhost:8501
Модели машинного обучения
Выберите модели машинного обучения:
LogR
SVC
KNN
Tree
RF
GB
Размер тестовой выборки
Выберите размер тестовой выборки:
0.50
0.10
0.90
False Positive Rate
KNN
Receiver operating characteristic
True Positive Rate
ROC curve (area = 0.94)
True label
Predicted label
0.83
0.17
0.055
0.95
Tree
Receiver operating characteristic
ROC curve (area = 0.89)
0.88
0.12
0.1
0.9
RF
Receiver operating characteristic

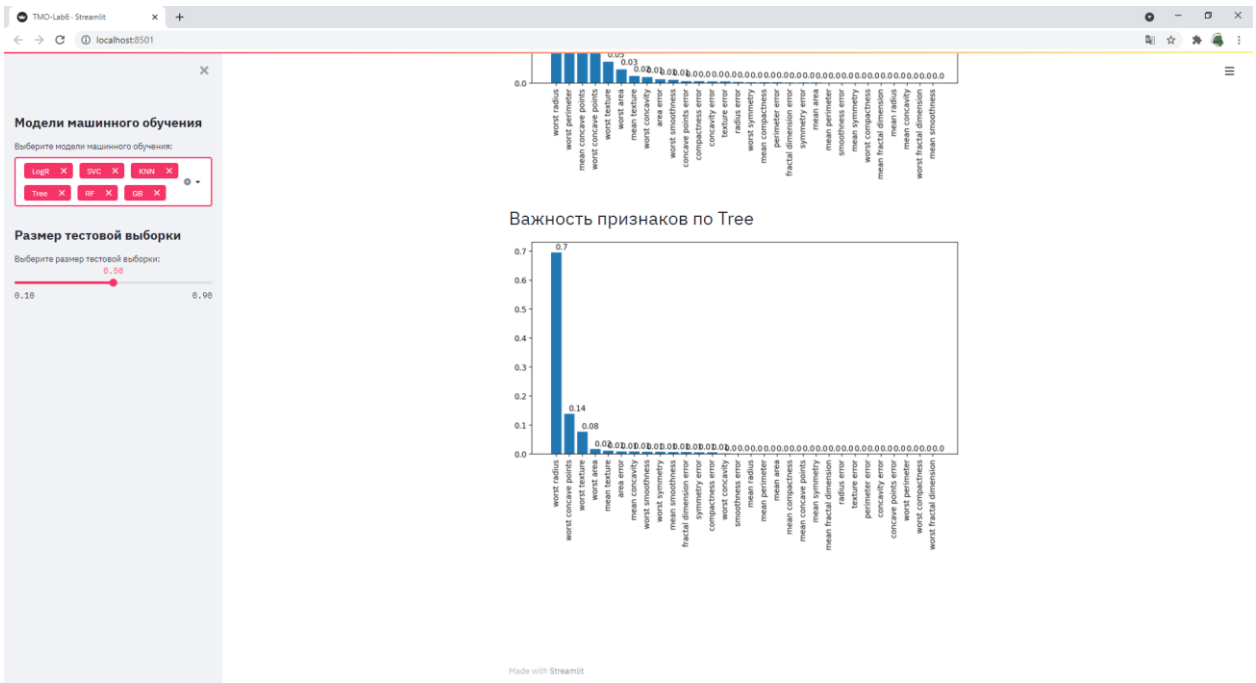




	accuracy_score	precision_score	recall_score	roc-auc
LogR	0.9263	0.9351	0.9505	0.9170
SVC	0.9088	0.8900	0.9780	0.8822
KNN	0.9053	0.9101	0.9451	0.8900
Tree	0.8912	0.9314	0.8956	0.8895
RF	0.9484	0.9508	0.9560	0.9343
GB	0.9298	0.9602	0.9286	0.9303

Важность признаков по RF


TMO-Lab6 - Streamlit
localhost:8501
Модели машинного обучения
Выберите модели машинного обучения:
LogR
SVC
KNN
Tree
RF
GB
Размер тестовой выборки
Выберите размер тестовой выборки:
0.50
0.10
0.90
Важность признаков по RF
Важность признаков по GB


TMO-Lab6 - Streamlit
localhost:8501
Модели машинного обучения
Выберите модели машинного обучения:
LogR
SVC
KNN
Tree
RF
GB
Размер тестовой выборки
Выберите размер тестовой выборки:
0.50
0.10
0.90
Важность признаков по Tree
Made with Streamlit