Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана национальный исследовательский университет)»
МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

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

Выполнила:
студент группы ИУ5-64Б
Подопригорова С. С.

Проверил:
Доцент кафедры ИУ5
Гапанюк Ю. Э.

Москва, 2021 г.

Описание задания

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

import math

from sklearn.preprocessing import MinMaxScaler

from sklearn.preprocessing import LabelEncoder

from sklearn.model_selection import cross_val_score

from sklearn.model_selection import GridSearchCV

# from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier

import matplotlib

matplotlib.use('TkAgg')

import matplotlib.pyplot as plt


from sklearn.model_selection import train_test_split
```

```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor

from heamy.estimator import Regressor
from heamy.pipeline import ModelsPipeline
from heamy.dataset import Dataset

from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from sklearn.metrics import median_absolute_error

from sklearn.tree import export_graphviz
from IPython.display import Image
from io import StringIO
import graphviz
import pydotplus

def get_png_tree(tree_model_param, feature_names_param):
    dot_data = StringIO()
    export_graphviz(tree_model_param, out_file=dot_data, feature_names=feature_names_param, filled=True,
rounded=True, special_characters=True)
    graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
    return graph.create_png()

@st.cache
def load_data():
    '''
    Загрузка данных
    '''
    data = pd.read_csv('data/SolarPrediction.csv', sep=',', nrows=500)
    return data

class MetricLogger:
```

```
def __init__(self):
    self.df = pd.DataFrame(
        {'metric': pd.Series([], dtype='str'),
        'alg': pd.Series([], dtype='str'),
        'value': pd.Series([], dtype='float')})

def add(self, metric, alg, value):
    """
    Добавление значения
    """
    # Удаление значения если оно уже было ранее добавлено
    self.df.drop(self.df[(self.df['metric']==metric)&(self.df['alg']==alg)].index, inplace = True)
    # Добавление нового значения
    temp = [{'metric':metric, 'alg':alg, 'value':value}]
    self.df = self.df.append(temp, ignore_index=True)

def get_data_for_metric(self, metric, ascending=True):
    """
    Формирование данных с фильтром по метрике
    """
    temp_data = self.df[self.df['metric']==metric]
    temp_data_2 = temp_data.sort_values(by='value', ascending=ascending)
    return temp_data_2['alg'].values, temp_data_2['value'].values

def plot(self, str_header, metric, loc=0.5, ascending=True, figsize=(5, 5)):
    """
    Вывод графика
    """
    array_labels, array_metric = self.get_data_for_metric(metric, ascending)
    fig, ax1 = plt.subplots(figsize=figsize)
    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(loc, a-0.05, str(round(b,3)), color='white')
        st.pyplot(fig)


@st.cache
def preprocess_data(data_in):
    '''
    Масштабирование признаков, функция возвращает X и y для кросс-валидации
    '''
    data_out = data_in.copy()

    data["Morning"] = (data["Time"] >= 6) & (data["Time"] <= 12)
    data["Afternoon"] = (data["Time"] >= 13) & (data["Time"] <= 19)
    data["Night"] = (data["Time"] >= 20) & (data["Time"] <= 23)
    data["EarlyMorning"] = (data["Time"] >= 0) & (data["Time"] <= 5)

    lab_enc = LabelEncoder()

    data['Morning'] = lab_enc.fit_transform(data['Morning'])
    data['Afternoon'] = lab_enc.fit_transform(data['Afternoon'])
    data['Night'] = lab_enc.fit_transform(data['Night'])
    data['EarlyMorning'] = lab_enc.fit_transform(data['EarlyMorning'])

    temp_y = data['Radiation']
    temp_X = data[['Temperature','Pressure', 'Humidity', 'WindDirection(Degrees)', 'Morning', 'Afternoon', 'Night',
'EarlyMorning']]

    X_train, X_test, y_train, y_test = train_test_split(temp_X, temp_y, train_size=0.25, random_state=1)
    return data[:], X_train, X_test, y_train, y_test


# Модели
models_list = ['LinR', 'SVR', 'Tree', 'RF', 'GB']
clas_models = {'LinR': LinearRegression(),
               'SVR': SVR(kernel='rbf', gamma=0.001, C=1000.0),
               'Tree': DecisionTreeRegressor(),
               'RF': RandomForestRegressor(n_estimators=15, oob_score=True, random_state=10),
               'GB':GradientBoostingRegressor()}
```

```
@st.cache(suppress_st_warning=True)
def print_models(models_select, metrics, X_train, X_test, y_train, y_test):
    current_models_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)

        RMSE = mean_squared_error(y_test, y_pred, squared=False)
        MAE = mean_absolute_error(y_test, y_pred)
        R2_Score = r2_score(y_test, y_pred)
        MedAE = median_absolute_error(y_test, y_pred)

        metrics.add('RMSE', model_name, RMSE)
        metrics.add('MAE', model_name, MAE)
        metrics.add('R2 Score', model_name, R2_Score)
        metrics.add('Median AE', model_name, MedAE)

        st.write('{} \t RMSE={}, MAE={}, R2={}, Median={}'.format(model_name, round(RMSE, 3), round(MAE, 3),
round(R2_Score, 3), round(MedAE, 3)))

    metrics.plot('Метрика: ' + 'RMSE', 'RMSE', ascending=False, figsize=(7, 3))
    metrics.plot('Метрика: ' + 'MAE', 'MAE', ascending=False, figsize=(7, 3))
    metrics.plot('Метрика: ' + 'R2 Score', 'R2 Score', loc = 0.05, ascending=True, figsize=(7, 3))
    metrics.plot('Метрика: ' + 'Median AE', 'Median AE', ascending=False, figsize=(7, 3))

@st.cache(suppress_st_warning=True)
def print_tree(X_train, y_train, columns):
    tree = clas_models['Tree']
    tree.fit(X_train, y_train)
    st.image(get_png_tree(tree, list(X_train.columns)))

st.sidebar.header('Модели машинного обучения')
models_select = st.sidebar.multiselect('Выберите модели', models_list)

data = load_data()
data, X_train, X_test, y_train, y_test = preprocess_data(data)
```

```
if st.checkbox('Показать корреляционную матрицу'):
    fig1, ax = plt.subplots(figsize=(10,5))
    sns.heatmap(data.corr(), annot=True, fmt='.2f')
    st.pyplot(fig1)

st.subheader('Оценка качества модели')
metrics = MetricLogger()
if len(models_select)>0:
    print_models(models_select, metrics, X_train, X_test, y_train, y_test)

if 'Tree' in models_select:
    if st.checkbox('Показать дерево решений'):
        print_tree(X_train, y_train, list(data.columns))
```

Экранные формы с примерами выполнения программы