

# загрузка библиотек

import pandas as pd

import numpy as np

from sklearn.preprocessing import LabelEncoder # кодирование категориальных

from sklearn.feature\_selection import SelectKBest # выбор признаков

from sklearn.feature\_selection import chi2 # выбор по Хи квадрат

from sklearn.model\_selection import train\_test\_split # деление на тест и обучение

# 0.2 тестовой к 0.8, 0.3 к 0.7 тренировочной

!pip install catboost

from catboost import CatBoostClassifier #ставим дополнительно через pip или conda

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score

# критерий качества, точность accuracy

# precision\_score, recall\_score, f1\_score (confusion\_matrix)

# работа с моделью в браузере (подключение)

# \* реализовать интерфейсную часть (API)

from flask import Flask, request, jsonify

# Набор констант по программе

MODEL\_FILE\_EFF = 'people\_university\_model\_eff.cbm' # модель с 3 лучшими признаками

MODEL\_FILE = 'people\_university\_model.cbm' # модель

DF\_COUNT = 2001  # число строк

SUBJ\_COUNT = 6  # число предметов

MIN\_GRADE = 0  # мин балл

MAX\_GRADE = 99  # макс балл

FINAL\_RESULT\_LIST = ['удовлетворительно', 'хорошо', 'отлично']  # результат теста

# Набор def и классов

def generate\_students():

    '''

    Создайте датасет из не менее чем 2000 записей, содержащий данные о

    среднем балле студентов (от 0 до 99) по 6 предметам и оценка итоговой

    лабораторной работы (удовлетворительно, хорошо, отлично).

    Строки датасета:

    id # идентификатор студента

    subjects = [{'math': score1}, {'reading': score2}, {'ml': score3}, {'physics': score4}, {'optimisation': score4}, {'sports': score5}] # список предметов

    final\_lab = ['удовлетворительно', 'хорошо', 'отлично'] # итоговая работа

    Создать из датасета DataFrame

    '''

    def generate\_final(row):

        # TODO поменять random

        # Балл ниже 30 - удовл

        cond\_c = row['mean\_score'] < 35

        # Балл 35 - 60 - хор

        cond\_b = row['mean\_score'] >= 35

        cond\_b\_1 = row['mean\_score'] < 60

        # Балл 60 - 100 - отл

        cond\_a = row['mean\_score'] >= 60

        val = ''

        if(cond\_c):

            val = FINAL\_RESULT\_LIST[0]

        elif(cond\_b and cond\_b\_1):

            val = FINAL\_RESULT\_LIST[1]

        else:

            val = FINAL\_RESULT\_LIST[2]

        # return FINAL\_RESULT\_LIST[0] if cond\_c else (FINAL\_RESULT\_LIST[1] if cond\_b else FINAL\_RESULT\_LIST[2])

        return val

    subjects = [f"subject\_{num + 1}" for num in range(SUBJ\_COUNT)]

    grades = np.random.randint(MIN\_GRADE, MAX\_GRADE + 1, (DF\_COUNT, SUBJ\_COUNT))

    student\_scores = pd.DataFrame (grades, columns = subjects)

    student\_scores.sum(axis=0) # axis 1 - по столбцам, axis 0 - построчно, сумма

    student\_scores['mean\_score'] = np.round((student\_scores.sum(axis=1)/SUBJ\_COUNT), 3)

    print(student\_scores.head(5))

    # apply - применить функцию к датасету

    student\_scores['final\_lab'] = student\_scores.apply(generate\_final, axis=1)

    return student\_scores

def classifyStudent(new\_student, new\_columns, model\_file, categories=[0, 1, 2]):

    '''

    Функция для классификации студентов

    @param new\_student - массив ключ-значение или строка в датасете

    '''

    # Чтение обученной модели из файла

    model = CatBoostClassifier()

    model.load\_model(model\_file)

    # Преобразование данных студента в DataFrame

    new\_data = pd.DataFrame([new\_student], columns=new\_columns)

    # Использование модели для предсказания

    predicted\_category = model.predict(new\_data)[0]

    # Преобразование обратно в текстовую категорию

    categories = [...] #категориальная по студентам

    predicted\_category = categories[predicted\_category]

    return predicted\_category

def create\_new\_dataset():

    df = generate\_students()

    df.to\_csv('ml/student.csv', index=False)

def preprocess\_dataset(

            file='ml/student.csv',

            to\_drop=['final\_lab', 'mean\_score'],

            y\_name='final\_lab',

            need\_to\_encode\_y=True,

            need\_best\_features=True,

            test\_size=0.2,

            random\_state=1234

        ):

    '''

    Разделить на X и y

    X - все предметы + \*студент\*

    id студента убрать из X

    y - результирующий столбец, Final lab <- LabelEncoder()

    '''

    df = pd.read\_csv(file)

    X = df.drop(columns=to\_drop, axis=1)

    print(X)

    y = df[y\_name] #Series

    print(y)

    if (need\_to\_encode\_y):

        label\_encoder = LabelEncoder()

        temp = label\_encoder.fit\_transform(y)

        new\_result = pd.DataFrame([y, temp], columns=['y', 'encoded\_y'])

        new\_result.to\_csv('ml/result\_labels.csv', index=False)

        y = temp

        print(y)

    if (need\_best\_features):

        # 2. Выберите лучшие 3 признака для обучения

        selector = SelectKBest(chi2, k=3)

        X = selector.fit\_transform(X, y)

    # 3. Разбейте датасет на тестовую и обучающую выборку train\_test\_split (для лучших признаков)

    # 80% обучающей выборки, 20% тестовой

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=test\_size, random\_state=random\_state)

    return X\_train, X\_test, y\_train, y\_test

def train\_model(X\_train, y\_train, iterations=100, depth=4, learning\_rate=0.1, file\_to\_save=MODEL\_FILE):

    # # 4. Проведите обучение модели, результатом должна быть сохраненная модель.

    # # Обучаем модель CatBoostClassifier

    model = CatBoostClassifier(iterations=iterations, depth=depth, learning\_rate=learning\_rate)

    model.fit(X\_train, y\_train)

    # Сохранение модели в файл

    model.save\_model(file\_to\_save)

def model\_report(model, X\_test, y\_test, average='weighted'):

    # 5. Проведите тестирование модели, результатом должно быть число - точность модели (accuracy) на тестовой выборке.

    # Делаем предсказания на тестовом наборе

    y\_pred = model.predict(X\_test)

    # Оцениваем точность модели

    accuracy\_eff = accuracy\_score(y\_test, y\_pred)

    print(f"Правильность (accuracy) модели: {accuracy\_eff}")

    precision\_eff = precision\_score(y\_test, y\_pred, average=average)

    print(f"Точность (precision) модели: {precision\_eff}")

    recall\_eff = recall\_score(y\_test, y\_pred, average=average)

    print(f"Полнота (recall) модели: {recall\_eff}")

    f1\_eff = f1\_score(y\_test, y\_pred, average=average)

    print(f"F1 мера модели: {f1\_eff}")

# #Натренировать модель

# X\_train, X\_test, y\_train, y\_test = preprocess\_dataset('ml/result\_labels.csv')

#train\_model(X\_train, y\_train)

new\_model = CatBoostClassifier()

#new\_model.load\_model(MODEL\_FILE)

#model\_report(new\_model, X\_test, y\_test, average='weighted')

# # Сохранить модель

# # 6. Реализуйте функцию, которая на вход принимает оценку студента по 6 предметам и возвращает прогноз оценки.

new\_grades = np.random.randint(MIN\_GRADE, MAX\_GRADE + 1, SUBJ\_COUNT)

print(new\_grades)

[74 99 51 70 53 73]