

ПИиКТ

Системы искусственного интеллекта

Лабораторная работа №3

Выполнил: Балтабаев Дамир

Группа: P33121

Преподаватель: Авдюшина А.Е.

**Задание**

Для студентов с четным порядковым номером в группе – датасет с классификацией грибов, а нечетным – датасет с данными про оценки студентов инженерного и педагогического факультетов (для данного датасета нужно ввести метрику: студент успешный/неуспешный на основании грейда)

2. Отобрать **случайным** образом sqrt(n) признаков

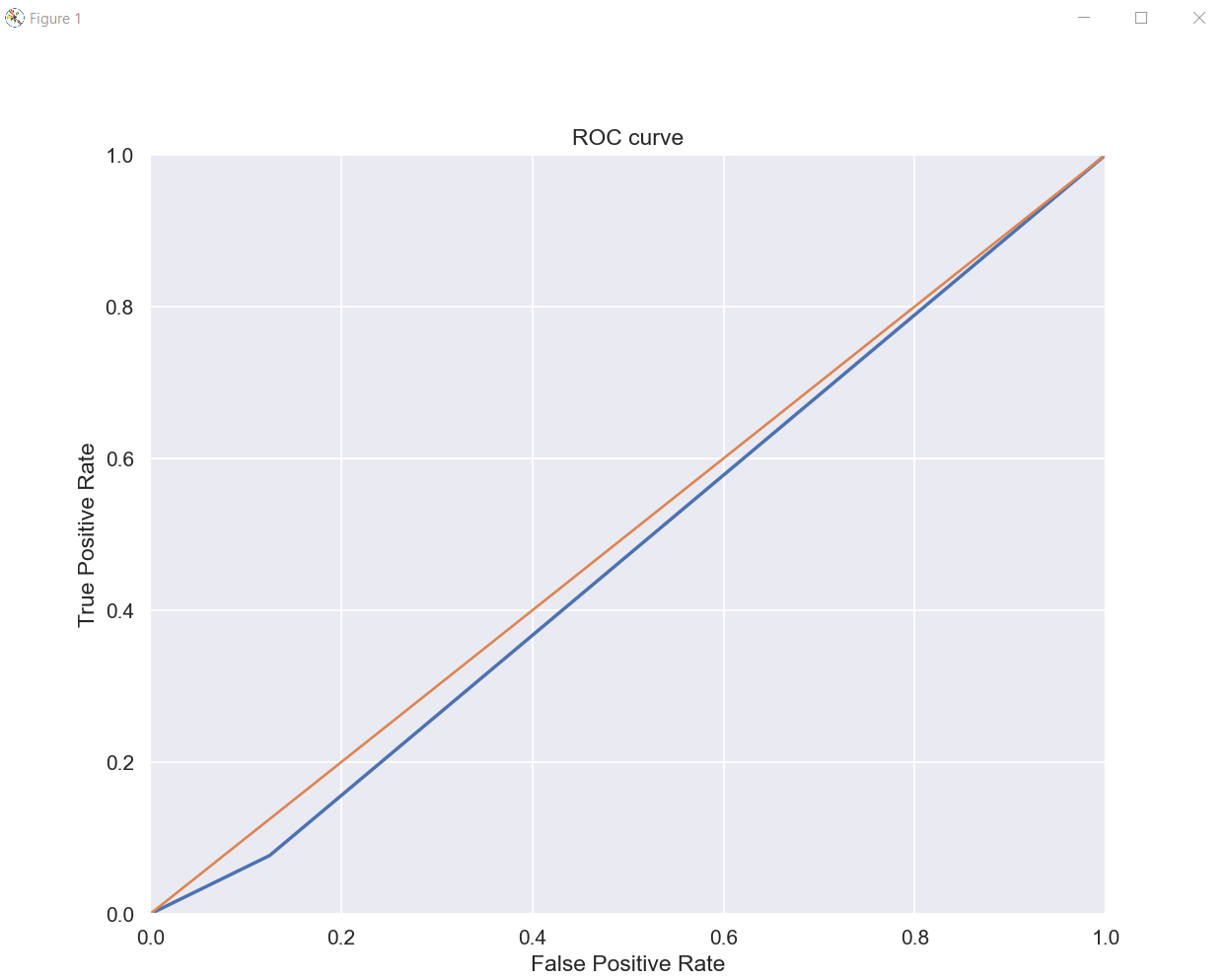
3. Реализовать без использования сторонних библиотек построение дерева решений (numpy и pandas использовать можно)

4. Провести оценку реализованного алгоритма с использованием Accuracy, precision и recall

5. Построить AUC-ROC и AUC-PR

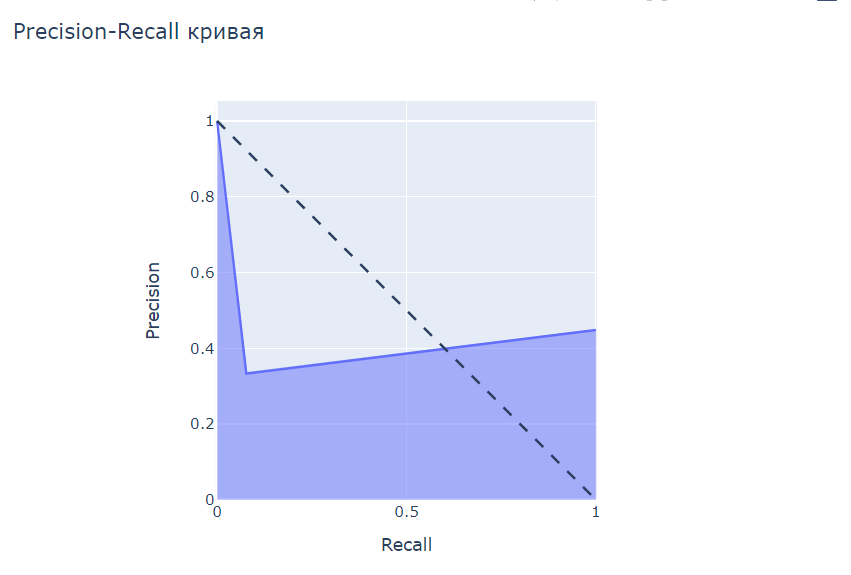
**Код**

from math import log  
import matplotlib  
import pandas as pd  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
import seaborn as sns  
import matplotlib.pyplot as plt  
from sklearn.metrics import roc\_curve, precision\_recall\_curve  
import warnings  
import plotly.express as px  
  
matplotlib.use("TkAgg")  
warnings.filterwarnings("ignore")  
  
  
# freq(Cj,T) - сколько элементов множества T принадлежат классу Cj  
def freq(data\_part, cj):  
 return (data\_part.GRADE == cj).sum()  
  
  
# энтропия для множества data\_part  
def info(data\_part):  
 sum = 0  
 for class\_value in data\_part.GRADE.unique():  
 p = freq(data\_part, class\_value) / data\_part.shape[0]  
 sum += p \* log(p, 2)  
 sum = sum \* (-1)  
 return sum  
  
  
# условная энтропия  
def info\_x(data\_part, partition\_field):  
 sum = 0  
 for field\_value in data\_part[partition\_field].unique():  
 subset = data\_part[data\_part[partition\_field] == field\_value]  
 sum += subset.shape[0] \* info(subset) / data\_part.shape[0]  
 return sum  
  
  
def split\_info\_x(data\_part, partition\_field):  
 sum = 0  
 for field\_value in data\_part[partition\_field].unique():  
 subset = data\_part[data\_part[partition\_field] == field\_value]  
 p = subset.shape[0] / data\_part.shape[0]  
 sum += p \* log(p, 2)  
 sum = sum \* (-1)  
 return sum  
  
  
# нормированный прирост информации  
def gain\_ratio(data\_part, partition\_field):  
 return (info(data\_part) - info\_x(data\_part, partition\_field)) / split\_info\_x(data\_part, partition\_field)  
  
  
class C4point5:  
  
 def \_\_init\_\_(self, data\_part):  
 subset\_fields = data\_part.drop(columns='GRADE').columns  
  
 subset\_fields\_gain\_ratios = pd.Series(  
 [gain\_ratio(data\_part, subset\_field) \* int(len(data\_part[subset\_field].unique()) != 1)  
 for subset\_field in subset\_fields]).fillna(0)  
  
 self.most\_popular\_class = data\_part.GRADE.mode()[0]  
 self.subtrees = {}  
 self.partition\_field = None  
 self.leaf\_class = data\_part.GRADE.unique()  
  
 if not any(subset\_fields\_gain\_ratios) or len(data\_part.GRADE.unique()) == 1:  
 self.leaf = True  
 return;  
  
 self.partition\_field = subset\_fields[np.argmax(  
 subset\_fields\_gain\_ratios)]  
  
 self.leaf = False  
 for partition\_field\_value in data\_part[self.partition\_field].unique():  
 self.subtrees[partition\_field\_value] = C4point5(  
 data\_part[data\_part[  
 self.partition\_field] == partition\_field\_value]  
 .drop(columns=self.partition\_field))  
  
 def predictMethod(self, data\_item):  
 if not self.leaf:  
 partition\_value = data\_item[self.partition\_field]  
 if partition\_value in self.subtrees:  
 return self.subtrees[partition\_value].predictMethod(data\_item)  
 return self.most\_popular\_class  
 final\_class = self.leaf\_class  
 if len(final\_class) == 1:  
 return final\_class[0]  
 return self.most\_popular\_class  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 data\_from\_csv = pd.read\_csv('DATA.csv', index\_col='STUDENT ID', sep=';')  
  
 prop\_count = int(data\_from\_csv.shape[1] \*\* 0.5)  
 data = data\_from\_csv.sample(prop\_count, axis=1)  
 data['GRADE'] = (data\_from\_csv.GRADE >= 4).astype(int)  
  
 train, test = train\_test\_split(data, test\_size=0.2)  
  
 tree = C4point5(train)  
  
 test\_results = []  
 predict\_positive = set()  
 real\_positive = set()  
 predict\_negative = set()  
 real\_negative = set()  
  
 for i in range(test.shape[0]):  
 predict = tree.predictMethod(test.drop(columns='GRADE').iloc[i])  
 if test.iloc[i].GRADE == 1:  
 real\_positive.add(i)  
 else:  
 real\_negative.add(i)  
 test\_results.append(predict)  
 if predict == 1:  
 predict\_positive.add(i)  
 else:  
 predict\_negative.add(i)  
  
 true\_positive = predict\_positive & real\_positive  
 true\_negative = predict\_negative & real\_negative  
  
 Accuracy = (len(true\_positive) + len(true\_negative)) / len(test\_results)  
  
 print('Accuracy :', Accuracy)  
  
 print(f'Precision для успевающих: {len(true\_positive) / len(predict\_positive)}')  
 print(f'Recall для успевающих: {len(true\_positive) / len(real\_positive)}')  
  
 print(f'Precision для неуспевающих: {len(true\_negative) / len(predict\_negative)}')  
 print(f'Recall для неуспевающих: {len(true\_negative) / len(real\_negative)}')  
  
 # sns.set(font\_scale=1.1)  
 # sns.set\_color\_codes("deep")  
 #  
 # plt.figure(figsize=(10, 8))  
 # fpr, tpr, thresholds = roc\_curve(test.GRADE.tolist(), test\_results)  
 # plt.plot(fpr, tpr, lw=2, label='ROC кривая')  
 # plt.plot([0, 1], [0, 1])  
 # plt.xlim([0.0, 1.0])  
 # plt.ylim([0.0, 1.0])  
 # plt.xlabel('False Positive Rate')  
 # plt.ylabel('True Positive Rate')  
 # plt.title('ROC curve')  
 # plt.show()  
 #  
 # precision, recall, thresholds = precision\_recall\_curve(test.GRADE.tolist(), test\_results)  
 #  
 # fig = px.area(  
 # x=recall, y=precision,  
 # title=f'Precision-Recall кривая',  
 # labels=dict(x='Recall', y='Precision'),  
 # width=700, height=500  
 # )  
 # fig.add\_shape(  
 # type='line', line=dict(dash='dash'),  
 # x0=0, x1=1, y0=1, y1=0  
 # )  
 # fig.update\_yaxes(scaleanchor="x", scaleratio=1)  
 # fig.update\_xaxes(constrain='domain')  
 #  
 # fig.show()

****

**Изображение выглядит как текст

Автоматически созданное описание**



**Вывод:**

В данной лабораторной работе я узнал что такое дерево решений и разобрался с его устройством, написав код на Python.