**Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»**

**Факультет программной инженерии и компьютерной техники**

**Системы искусственного интеллекта**

**Лабораторная работа №3**

**Вариант 11**

Студент:

Кустарев Иван Павлович P33121

Преподаватель:

Авдюшина Анна Евгеньевна

Оценка: \_\_\_\_\_\_\_\_\_\_

Подпись преподавателя: \_\_\_\_\_\_\_\_\_\_

Санкт-Петербург 2022

1. **Задание**
2. 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)
3. Отобрать случайным образом sqrt(n) признаков
4. Реализовать без использования сторонних библиотек построение дерева решений (numpy и pandas использовать можно)
5. Провести оценку реализованного алгоритма с использованием Accuracy, precision и recall
6. Построить AUC-ROC и AUC-PR
7. **Программа**

import math  
import random  
  
import matplotlib.pyplot as plt  
  
PARAMS\_NUMBER = 6  
  
  
def read\_data():  
 T = []  
 selected\_columns = random.sample(list(range(1, 32)), PARAMS\_NUMBER)  
 with open('lab3.csv') as file:  
 for line in file.readlines():  
 dataset\_seperated\_columns = line.split(';')  
 T.append(  
 (  
 [int(dataset\_seperated\_columns[c]) for c in selected\_columns],  
 int(dataset\_seperated\_columns[32]),  
 )  
 )  
 return T  
  
  
def select\_all\_classes(T):  
 classes = []  
 for \_, C in T:  
 classes.append(C)  
 return classes  
  
  
def freq(C, T):  
 filtered = []  
 for i in T:  
 if i[1] == C:  
 filtered.append(i)  
 return len(filtered)  
  
  
def split\_by\_X(T, X):  
 T\_parts = {}  
 for params, clazz in T:  
 parameter = params[X]  
 if parameter not in T\_parts:  
 T\_parts[parameter] = []  
 T\_parts[parameter].append((params, clazz))  
 return T\_parts  
  
  
def info(T):  
 return -sum([freq(C, T) / len(T) \* math.log2(freq(C, T) / len(T)) for C in select\_all\_classes(T)])  
  
  
def info\_x(T, X):  
 return sum([len(Ti) / len(T) \* info(Ti) for Ti in split\_by\_X(T, X).values()])  
  
  
def split\_info\_x(T, X):  
 return -sum([len(Ti) / len(T) \* math.log2(len(Ti) / len(T)) for Ti in split\_by\_X(T, X).values()])  
  
  
def gain\_ratio(T, X):  
 divider = split\_info\_x(T, X)  
 if divider == 0:  
 return -math.inf  
 return (info(T) - info\_x(T, X)) / divider  
  
  
def select\_best\_x(T):  
 best\_X = 0  
 for X in range(PARAMS\_NUMBER):  
 if gain\_ratio(T, X) > gain\_ratio(T, best\_X):  
 best\_X = X  
 return best\_X  
  
  
def build\_tree\_node(T):  
 X = select\_best\_x(T)  
 T\_parts = split\_by\_X(T, X)  
 if len(T\_parts.keys()) == 1:  
 return select\_all\_classes(T)[0]  
 return {X: {clazz: build\_tree\_node(T\_parts[clazz]) for clazz in T\_parts.keys()}}  
  
  
def check(tree, params):  
 X = list(tree.keys())[0]  
 C = tree[X][params[X]]  
 if type(C) == int:  
 return C  
 else:  
 return check(C, params)  
  
  
def check\_all(T):  
 results = []  
 for params, real\_clazz in T:  
 check\_res = check(tree, params)  
 print(f'By check: {check\_res} and true is: {real\_clazz}')  
 results.append((check\_res, real\_clazz))  
 return results  
  
  
def apr(results, separator=3):  
 tp = 0  
 tn = 0  
 fp = 0  
 fn = 0  
 for checked\_class, real\_class in results:  
 if (checked\_class >= separator) == (real\_class >= 3):  
 if checked\_class >= separator:  
 tp += 1  
 else:  
 tn += 1  
 else:  
 if checked\_class >= separator:  
 fp += 1  
 else:  
 fn += 1  
 accuracy = (tp + tn) / (tp + fp + fn + tn)  
 precision = tp / (tp + fp)  
 recall = tp / (tp + fn)  
 return accuracy, precision, recall  
  
  
def print\_AUC\_ROC(results):  
 results.sort(key=lambda item: -item[0])  
 x = [0]  
 y = [0]  
 for i in range(0, len(results)):  
 if results[i][1] >= PARAMS\_NUMBER:  
 x.append(x[-1])  
 y.append(y[-1] + 1)  
 else:  
 x.append(x[-1] + 1)  
 y.append(y[-1])  
 plt.plot(x, y)  
 # plt.plot([x[0], x[len(x) - 1]], [y[0], y[len(y) - 1]])  
 plt.title("AUC-ROC")  
 plt.show()  
  
  
def print\_AUC\_PR(results):  
 x = []  
 y = []  
 last\_recall = None  
 for separator in range(1, 7):  
 \_, precision, recall = apr(results, separator)  
 if last\_recall is not None:  
 x.append(last\_recall)  
 y.append(precision)  
 x.append(recall)  
 y.append(precision)  
 last\_recall = recall  
 plt.plot(x, y)  
 plt.title("AUC-PR")  
 plt.show()  
  
  
T = read\_data()  
tree = build\_tree\_node(T)  
results = check\_all(T)  
accuracy, precision, recall = apr(results)  
print(f'Accuracy: {accuracy}')  
print(f'Precision: {precision}')  
print(f'Recall: {recall}')  
print\_AUC\_ROC(results)  
print\_AUC\_PR(results)

1. **Вывод**

В ходе выполнения лабораторной работы был изучен алгоритм построения дерева решений c4.5. Знания были практически применены при реализации алгоритма на языке python. Были изучены такие понятия как accuracy, precision, recall, а так же AUC\_ROC и AUC\_PR диаграммы.