Национальный исследовательский университет информационных технологий, механики и оптики
Факультет ПИиКТ

Системы искусственного интеллекта

Лабораторная работа №3

Вариант Чётный

Работу выполнил:
Конаныхина А.А.

Группа:
P33102

Преподаватель:
Кугаевских А.В.

Санкт-Петербург
2022

Задание:

1. 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

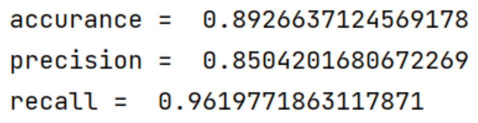
Код:

Файл tree.py:

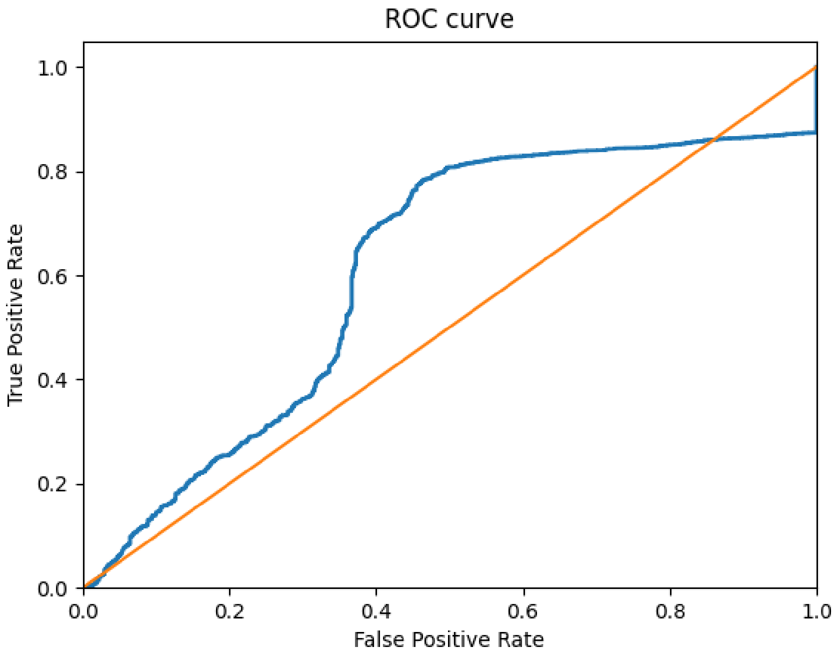
```
import math

class Chose:
    attr_num: int
    variants: list
    choses: list
    def __init__(self, attr_num, variants, choses):
        self.attr_num = attr_num
        self.variants = variants
        self.choses = choses

def calculate_possibles(keys, attributes):
    possible_values = []
    possible_keys = []
    for i in range(len(keys)):
        if keys[i] not in possible_keys:
            possible_keys.append(keys[i])
    for i in range(len(attributes)):
        possible_values_for_i = []
        for j in range(len(attributes[i])):
            if attributes[i][j] in possible_values_for_i:
                continue
            possible_values_for_i.append(attributes[i][j])
        possible_values.append(possible_values_for_i)
    return possible_keys, possible_values

def calculate_info_t(keys):
    possible_keys, possible_attributes = calculate_possibles(keys, [])
    info_t = 0
    for i in possible_keys:
        info_t += (keys.count(i) / len(keys)) * math.log(keys.count(i) /
len(keys), 2)
    info_t *= -1
    return info_t


def select_by_attribute_value(keys, attributes, attr, val):
    output_keys = []
    output_attributes = [[] for i in range(len(attributes))]
    for i in range(len(keys)):
        if attributes[attr][i] == val:
            output_keys.append(keys[i])
            for j in range(len(attributes)):
                output_attributes[j].append(attributes[j][i])
    return output_keys, output_attributes


def chose_attribute(keys, attributes):
    possible_keys, possible_attributes = calculate_possibles(keys,
attributes)

    raties_ = []
    for i in range(len(attributes)):
        info_x = 0
        split = 0
        for j in possible_attributes[i]:
            selected_keys, selected_attr = select_by_attribute_value(keys,
attributes, i, j)
            info_x += (len(selected_keys)/len(keys)) *
calculate_info_t(selected_keys)
            split += (len(selected_keys)/len(keys)) *
math.log(len(selected_keys)/len(keys), 2)
        split *= -1
        if split == 0:
            raties_.append(0)
        else:
            raties_.append((calculate_info_t(keys) - info_x)/split)
    for i in range(len(raties_)):
        if raties_[i] == max(raties_):
            return i


def create_chose(keys, attributes, max_depth, depth = 0):
    possible_keys, possible_attributes = calculate_possibles(keys,
attributes)

    if (depth >= max_depth):
        counts = []
        for i in possible_keys:
            counts.append(keys.count(i))
        for i in possible_keys:
            if keys.count(i) == max(counts):
                return i

    attribute = chose_attribute(keys, attributes)

    choses = []
    for i in range(len(possible_attributes[attribute])):
        selected_keys, selected_attributes = select_by_attribute_value(keys,
attributes, attribute, possible_attributes[attribute][i])
        possible_selected_keys, possible_selected_attributes =
calculate_possibles(selected_keys, selected_attributes)

        if len(possible_selected_keys) == 1:
            choses.append(possible_selected_keys[0])
        else:
            choses.append(create_chose(selected_keys, selected_attributes,
max_depth, depth + 1))

    return Chose(attribute, possible_attributes[attribute], choses)
```

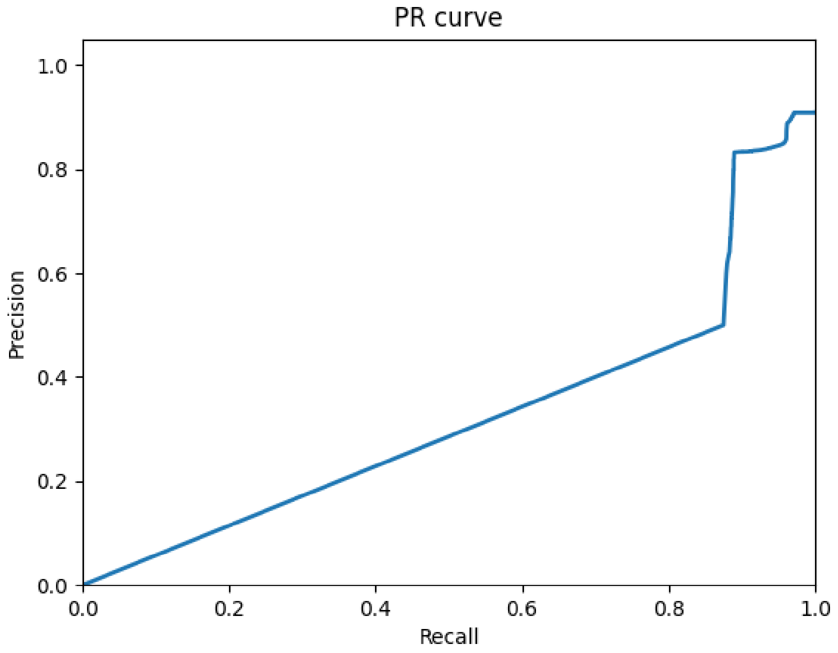
```python
def createDisTree(keys, attributes):
    return create_chose(keys, attributes, len(attributes))

def select(tree: Chose, value):
    for i in range(len(tree.choses)):
        if value == tree.variants[i]:
            return tree.choses[i]


def calculate_result(tree: Chose, attributes):
    result = ""
    while result == "":
        select_result = select(tree, attributes[tree.attr_num])
        if type(select_result) == Chose:
            result = calculate_result(select_result, attributes)
        else:
            result = select_result
    return result

def calculate_metrics(tree: Chose, keys, attributes, positive_class = "e"):
    predicted_keys = []
    for i in range(len(keys)):
        predict_attributes = []
        for j in range(len(attributes)):
            predict_attributes.append(attributes[j][i])
        predicted_keys.append(calculate_result(tree, predict_attributes))

    TP = 0
    TN = 0
    FN = 0
    FP = 0
    ROC_TP = [0]
    ROC_FP = [0]
    PR_PR = [0]
    PR_RC = [0]
    for i in range(len(keys)):
        if keys[i] == predicted_keys[i] == positive_class:
            TP += 1
        elif keys[i] == predicted_keys[i]:
            TN += 1
        elif predicted_keys[i] == positive_class:
            FP += 1
        else:
            FN += 1
        ROC_TP.append(TP)
        ROC_FP.append(FP)
        PR_PR.append(0 if TP + FP == 0 else (TP)/(TP + FP))
        PR_RC.append(0 if TP + FN == 0 else  (TP)/(TP + FN))

    accurancy = (TP + TN)/(TP + TN + FP + FN)
    precision = 0 if TP + FP == 0 else (TP)/(TP + FP)
    recall = 0 if TP + FN == 0 else  (TP)/(TP + FN)

    ROC_TP = [x / TP for x in ROC_TP]
    ROC_FP = [x / FP for x in ROC_FP]

    PR_RC.sort()
    PR_PR.sort()

    return accurancy, precision, recall, ROC_TP, ROC_FP, PR_PR, PR_RC
```

Вывод программы:



```
accurance =  0.8926637124569178
precision =  0.8504201680672269
recall =  0.9619771863117871
```





Вывод:

В ходе выполнения данной лабораторной работы были изучены алгоритмы построения дерева решений и решения задачи классификации.