ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

ЛАБОРАТОРНАЯ РАБОТА № 2
Решающие деревья

Выполнил:
Студент группы: ИП-715
Винтер Антон

Проверил: ассистент кафедры ПМиК
Морозова К.И.

Новосибирск – 2020

Оглавление

Текст задания.

Данная работа носит творческий характер и призвана показать, насколько студент подготовлен к реальному применению полученных знаний на практике. Как известно, в реальной работе никаких вводных данных не предоставляется, тем не менее, мы слегка пренебрегли данным правилом и предоставили теорию и предпочтительный метод для применения.

В приложенном файле (heart_data.csv) располагаются реальные данные по сердечной заболеваемости, собранные различными медицинскими учреждениями. Каждый человек представлен 13-ю характеристиками и полем goal, которое показывает наличие болезни сердца, поле принимает значение 0 или 1 (0 – нет болезни, 1 - есть). Символ ‘?’ в каком-либо поле означает, что для конкретного человека отсутствуют данные в этом поле (либо не производились замеры, либо не записывались в базу).

Требуется имеющиеся данные разбить на обучающую и тестовую выборки в процентном соотношении 70 к 30. После чего по обучающей выборке необходимо построить решающее дерево. Для построения дерева можно пользоваться любыми существующими средствами. Кроме того, для построения дерева необходимо будет решить задачу выделения информативных решающих правил относительно имеющихся числовых признаков.

Разрешается использовать уже реализованные решающие деревья из известных библиотек (например, scikit-learn для Python), либо реализовывать алгоритм построения дерева самостоятельно (все необходимые алгоритмы представлены в теории по ссылке).

В качестве результата работы необходимо сделать не менее 10 случайных разбиений исходных данных на обучающую и тестовую выборки, для каждой построить дерево и протестировать, после чего построить таблицу, в которой указать процент правильно классифицированных данных. Полученную таблицу необходимо включить в отчёт по лабораторной работе.

В отчёте следует отразить следующие изменяемые параметры: глубина дерева и количество деревьев для каждого тестируемого случая.

Описание основных функций

pandas - это Python библиотека для анализа и обработки данных. Она действительно быстрая и позволяет вам легко исследовать данные.

PyDotPlus - это улучшенная версия старого проекта pydot, которая предоставляет интерфейс Python для языка Dot Graphviz.

В Python sklearn — это пакет, который содержит все необходимые пакеты для реализации алгоритма машинного обучения.

Деревья решений (Dts) — это непараметрический контролируемый метод обучения, используемый для классификации, в данном случаем. Цель состоит в том, чтобы создать модель, которая предсказывает значение, а целевая переменная путем изучения простых правил принятия решения

DecisionTreeClassifier принимает в качестве входных данных два массива: массив X, содержащий обучающие выборки, и массив Y целочисленных значений, содержащий метки классов для обучающих выборок

#чем больше depth тем больше значение на обучающей выборке
#чем меньше samples тем обущающая ближе к 1

min_samples_leafint или float - по умолчанию = 1.
Минимальное количество образцов, требуемых для нахождения в листовом узле

Чем больше значение min_samples_leafint, тем значение обучающей выборки будет уменьшаться. При min_samples_leafint = 1, значение обучающей выборки может достигаться 1.0

max_depthint, по умолчанию = None
Максимальная глубина дерева. Если нет, то узлы расширяются до тех пор, пока все листья не станут чистыми или

При увеличении max_depthint, значение на обучающей выборке будет так же увеличиваться, в конечном итоге приблизиться к единице, но не будет ей равно

функция: train_test_split(x, y, test_size=0.3)

test_size : float, int или None, необязательно (по умолчанию=None), если float, должно быть между 0.0 и 1.0 и представлять долю набора данных, включаемого в тестовое разделение

функция dt.fit(train_x, train_y):

Экземпляр оценки dt (для классификатора) сначала устанавливается в модель; то есть он должен учиться у модели. Это делается путем передачи нашего тренировочного набора в метод fit. Для обучающего набора мы будем использовать все изображения из нашего набора данных, за исключением последнего изображения, которое мы оставим для нашего прогнозирования. Мы выбираем обучающий набор с синтаксисом Python (train_x), который создает новый массив, содержащий все, кроме последнего элемента (train_y)

dt.score(train_x, train_y)):

В многоуровневой классификации эта функция вычисляет точность подмножества: набор меток, предсказанных для выборки, должен точно соответствовать соответствующему набору меток в y_true.

Результат работы программы

```
[[28  1  2 ...  0  0  0]
 [29  1  2 ...  0  0  0]
 [29  1  2 ...  0  0  0]
 ...
 [56  1  4 ...  2  0  0]
 [58  0  2 ...  2  0  7]
 [65  1  4 ...  2  0  0]]

0.На обучающей выборке:  0.8975609756097561
0.На тестовой выборке:  0.7640449438202247

1.На обучающей выборке:  0.8878048780487805
1.На тестовой выборке:  0.7415730337078652

2.На обучающей выборке:  0.8975609756097561
2.На тестовой выборке:  0.7752808988764045

3.На обучающей выборке:  0.8731707317073171
3.На тестовой выборке:  0.7415730337078652

4.На обучающей выборке:  0.8829268292682927
4.На тестовой выборке:  0.8314606741573034

5.На обучающей выборке:  0.8731707317073171
5.На тестовой выборке:  0.797752808988764

6.На обучающей выборке:  0.8878048780487805
6.На тестовой выборке:  0.7528089887640449

7.На обучающей выборке:  0.8975609756097561
7.На тестовой выборке:  0.7528089887640449

8.На обучающей выборке:  0.9024390243902439
8.На тестовой выборке:  0.7191011235955056

9.На обучающей выборке:  0.8682926829268293
9.На тестовой выборке:  0.8202247191011236

True
```

Код программы

```
import pandas as pd

import pydotplus
from sklearn import tree
from sklearn.model_selection import train_test_split

df = pd.read_csv('data.csv', header=0)
df = df.replace('?', 0.)
dt = tree.DecisionTreeClassifier(min_samples_leaf=5, max_depth = 10)

x = df.values[:, 0:13]
print(x)
print()

y = df.values[:, 13]
y = y.astype('int')

for i in range(10):
  train_x, test_x, train_y, test_y = train_test_split(x, y, test_size=0.3)
  dt.fit(train_x, train_y)
  print(str(i) + '.На обучающей выборке: ', dt.score(train_x, train_y))
  print(str(i) + '.На тестовой выборке: ', dt.score(test_x, test_y))
  print()


gv_str = tree.export_graphviz(dt, out_file=None, feature_names=df.head()
[0:0].columns[:-1])
graph = pydotplus.graph_from_dot_data(gv_str)
graph.write_png("tree.png")
```