Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

Отчёт по лабораторной работе №2

Решающие деревья

Выполнил:

студент гр. ИП-111

Кузьменок Д.В.

Проверил:

Старший преподаватель кафедры ПМиК

Дементьева К.И.

Новосибирск, 2024 г.

**Задание**

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).

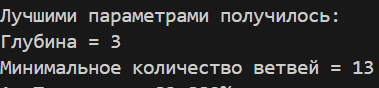
В качестве результата работы необходимо сделать не менее 10 случайных разбиений исходных данных на обучающую и тестовую выборки, для каждой построить дерево и протестировать, после чего построить таблицу, в которой указать процент правильно классифицированных данных. Полученную таблицу необходимо включить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

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

Имя файла: heart\_data.csv.

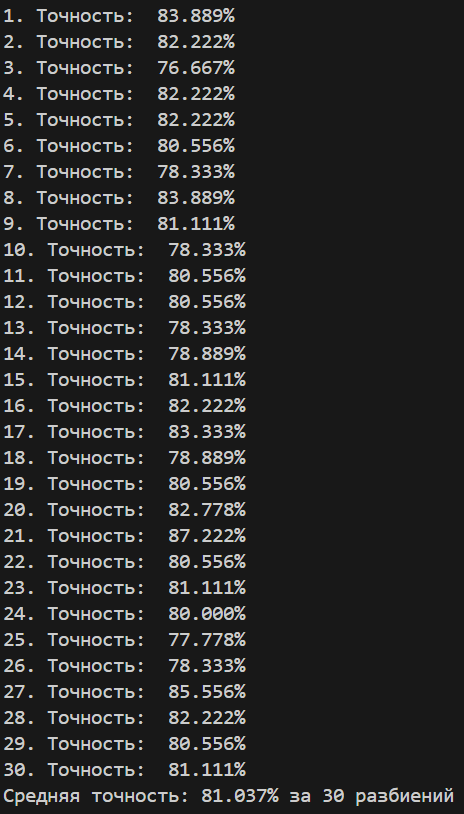
**Результаты**

Ход вычисления оптимальных значений максимальной глубины и минимального количества листьев:



Самый оптимальный вариант на обучающей модели получается при max\_depth= 3, min\_samples\_leaf = 13.

Для проверки полученных результатов я запускаю 30 раз подсчет точности для вычисления средней точности угадывания класса:



**Код программы**

**from** **sklearn.model\_selection** **import** train\_test\_split

**from** **sklearn.impute** **import** SimpleImputer

**from** **sklearn.tree** **import** DecisionTreeClassifier

**from** **sklearn.metrics** **import** accuracy\_score

**import** **numpy** **as** **np**

**import** **pandas** **as** **pd**

**def** **train\_and\_evaluate\_model**(file\_name, test\_size=**0.3**, n\_splits=**30**):

data = np.genfromtxt(file\_name, delimiter=',', skip\_header=**True**)

X = data[:, :-**1**]

y = data[:, -**1**]

best\_params = find\_best\_params(X, y, test\_size)

print(f"Лучшими параметрами получилось:**\n**Глубина = {best\_params["max\_depth"]}**\n**Минимальное количество ветвей = {best\_params["min\_samples\_leaf"]}")

average\_accuracy = cross\_validate\_model(X, y, best\_params, test\_size, n\_splits)

**return** average\_accuracy

**def** **find\_best\_params**(X, y, test\_size):

best\_accuracy = **0**

best\_params = {}

**for** max\_depth **in** range(**2**, **20**):

**for** min\_samples\_leaf **in** range(**2**, **20**):

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

**for** col **in** range(X\_train.shape[**1**]):

**if** "?" **in** X\_train[:, col]:

X\_train[:, col] = pd.to\_numeric(X\_train[:, col], errors='coerce')

X\_test[:, col] = pd.to\_numeric(X\_test[:, col], errors='coerce')

imp = SimpleImputer(missing\_values=np.nan, strategy='mean')

X\_train = imp.fit\_transform(X\_train)

X\_test = imp.transform(X\_test)

model = DecisionTreeClassifier(max\_depth=max\_depth, min\_samples\_leaf=min\_samples\_leaf)

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

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

**if** accuracy > best\_accuracy:

best\_accuracy = accuracy

best\_params = {'max\_depth': max\_depth, 'min\_samples\_leaf': min\_samples\_leaf}

**return** best\_params

**def** **cross\_validate\_model**(X, y, params, test\_size, n\_splits):

total\_accuracy = **0**

**for** i **in** range(n\_splits):

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

**for** col **in** range(X\_train.shape[**1**]):

**if** "?" **in** X\_train[:, col]:

X\_train[:, col] = pd.to\_numeric(X\_train[:, col], errors='coerce')

X\_test[:, col] = pd.to\_numeric(X\_test[:, col], errors='coerce')

imp = SimpleImputer(missing\_values=np.nan, strategy='mean')

X\_train = imp.fit\_transform(X\_train)

X\_test = imp.transform(X\_test)

model = DecisionTreeClassifier(\*\*params)

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

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

total\_accuracy += accuracy

print(f"{i+1}. Точность: {(accuracy \* 100):.3f}%")

average\_accuracy = total\_accuracy / n\_splits

print(f"Средняя точность: {(average\_accuracy \* 100):.3f}% за {n\_splits} разбиений")

**return** average\_accuracy

**if** \_\_name\_\_ == "\_\_main\_\_":

file\_name = "heart\_data.csv"

average\_accuracy = train\_and\_evaluate\_model(file\_name)