Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Кафедра прикладной математики и кибернетики

Отчет

По лабораторной работе №2

Логические классификаторы.

Решающие деревья.

Выполнил:

студент гр. ИП-712

Алексеев С.В.

Работу проверил: Ассистент кафедры  
Морозова К.И.

Новосибирск 2020 г.

# Задание

Реализовать классификатор на основе решающих деревьев.

# Процесс выполнения работы

1. Данные были разбиты на обучающий и тестовый наборы с соотношением 7:3
2. На обучающем наборе данных было создано решающее дерево.
3. На тестовом наборе данных было протестировано созданное решающее дерево.

* Точность на 10 разбиениях по умолчанию составила:  
  0.76, 0.77, 0.71, 0.76, 0.74, 0.76, 0.78, 0.72, 0.72, 0.72.
* Точность на 5 разбиениях с максимальной глубиной дерева равной 5 и максимальным количеством листьев 10:  
  0.79, 0.77, 0.76, 0.82, 0.81
* Точность на 5 разбиениях с максимальной глубиной дерева равной 10 и максимальным количеством листьев 5:  
  0.77, 0.84, 0.76, 0.8, 0.78

# Вывод

В лабораторной работе был реализован классификатор на основе решающих деревьев. При изменении глубины дерева и количества листьев можно наблюдать небольшое улучшение результатов. Классификатор на основе решающих деревьев показывает точность в данной задаче в районе 0.75.

# Листинг

import numpy as np  
import pandas as pd  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn import metrics  
from sklearn.impute import SimpleImputer  
  
dataset = pd.read\_csv('heart\_data.csv')  
dataset.head()  
X = dataset.loc[:, 'age':'thal']  
y = dataset['goal']  
X = X.replace('?', np.nan)  
y = y.replace('?', np.nan)  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3)  
imputer = SimpleImputer()  
X\_train = imputer.fit\_transform(X\_train)  
  
i = 1  
while i < 20:  
 clf = DecisionTreeClassifier(max\_depth=i, min\_samples\_leaf=1)  
 clf.fit(X\_train, y\_train)  
 X\_test = imputer.transform(X\_test)  
 y\_pred = clf.predict(X\_test)  
 print(clf.score(X\_train, y\_train))  
 print(clf.score(X\_test, y\_test))  
 print('')  
 i = i + 1  
  
clf.fit(X\_train, y\_train)  
X\_test = imputer.transform(X\_test)  
y\_pred = clf.predict(X\_test)  
print(clf.score(X\_train, y\_train))  
print(clf.score(X\_test, y\_test))  
# print(y\_pred)  
  
print(clf.max\_depth)  
print(clf.min\_samples\_leaf)