Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и технологий  
Высшая школа программной инженерии

Лабораторная работа №3

По дисциплине «Машинное обучение»

Выполнил студент гр 33534/5  Донцов А. Д.

Руководитель И. А. Селин

Санкт-Петербург  
2019 г.

# Задание

1) Загрузите набор данных Glass из файла glass.csv. Набор данных (признаки, классы) был изучен

в работе «Метод ближайших соседей». Постройте дерево классификации для модели,

предсказывающей тип (Type) по остальным признакам. Дайте интерпретацию полученным

результатам. Является ли построенное дерево избыточным? Исследуйте зависимость точности

классификации от критерия расщепления, максимальной глубины дерева и других параметров по

вашему усмотрению.

2) Загрузите набор данных Lenses Data Set из файла Lenses.txt:

3 класса (последний столбец):

1: пациенту следует носить жесткие контактные линзы,

2: пациенту следует носить мягкие контактные линзы,

3 : пациенту не следует носить контактные линзы.

Признаки (категориальные):

1. возраст пациента: (1) молодой, (2) предстарческая дальнозоркость, (3) старческая

дальнозоркость

2. состояние зрения: (1) близорукий, (2) дальнозоркий

3. астигматизм: (1) нет, (2) да

4. состояние слезы: (1) сокращенная, (2) нормальная

Постройте дерево решений. Какие линзы надо носить при предстарческой дальнозоркости,

близорукости, при наличии астигматизма и сокращенной слезы?

3) Загрузите набор данных spam7 из файла spam7.csv. Постройте оптимальное, по вашему

мнению, дерево классификации для параметра yesno. Объясните, как был осуществлён подбор

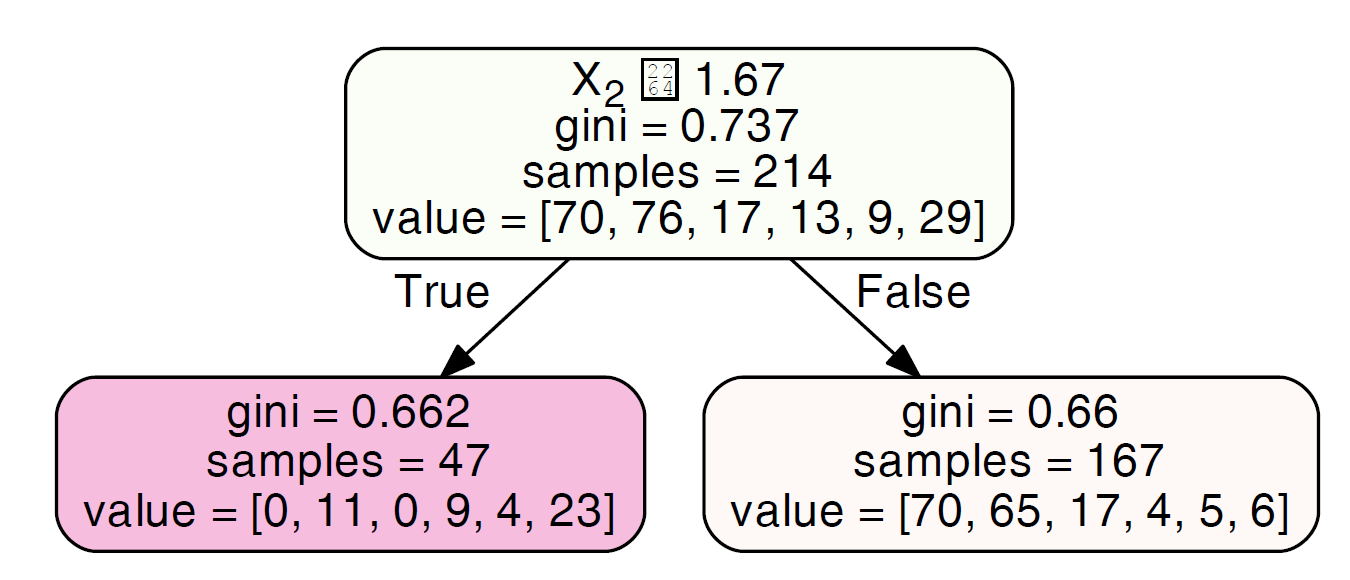
параметров.

# Ход работы

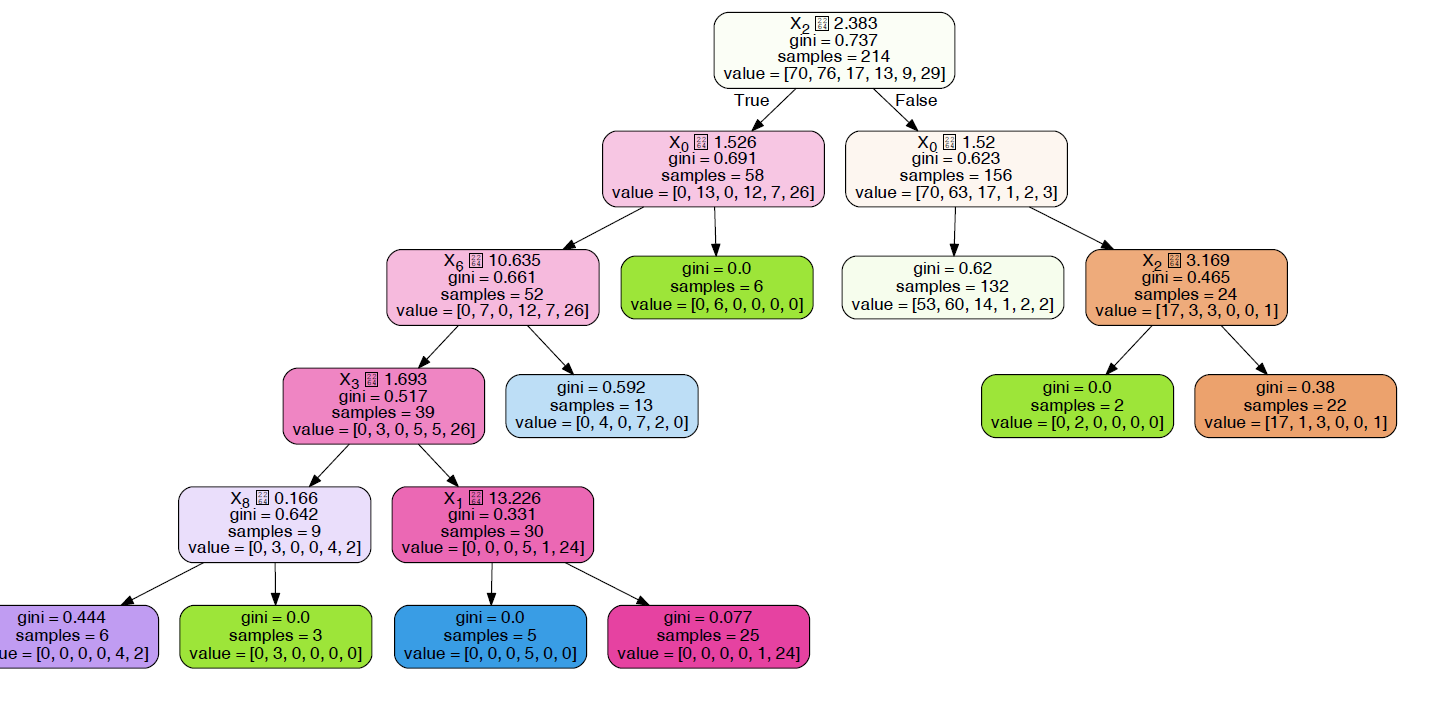
1. Для теста были выбраны классификаторы с двумя параметрами:
   1. Классификатор 1
      1. Max\_depth – в диапазоне от 1 до 19
      2. Max\_leaf\_nodes – в диапазоне от 2 до 20
      3. Criterion - gini
      4. Splitter – random
   2. Классификатор 2
      1. Max\_depth – в диапазоне от 1 до 19
      2. Max\_leaf\_nodes – в диапазоне от 2 до 20
      3. Criterion - entropy
      4. Splitter – best
2. Для набора данных lenses.csv был построено дерево, предсказанный тип – 3.
3. Для набора данных spam7.csv было построено решающее дерево. Параметры для классификатора были выбраны с помощью GridSearchCV. Параметры, передаваемые в GridSearchCV: parameters = {'criterion': ('gini', 'entropy'), 'splitter': ('best', 'random'), 'max\_depth': [1, 10]}  
   Таким образом, был выбран следующий набор параметров: DecisionTreeClassifier(criterion='gini', max\_depth=10, min\_samples\_leaf=1, min\_samples\_split=2, splitter='best')

# Результаты работы:

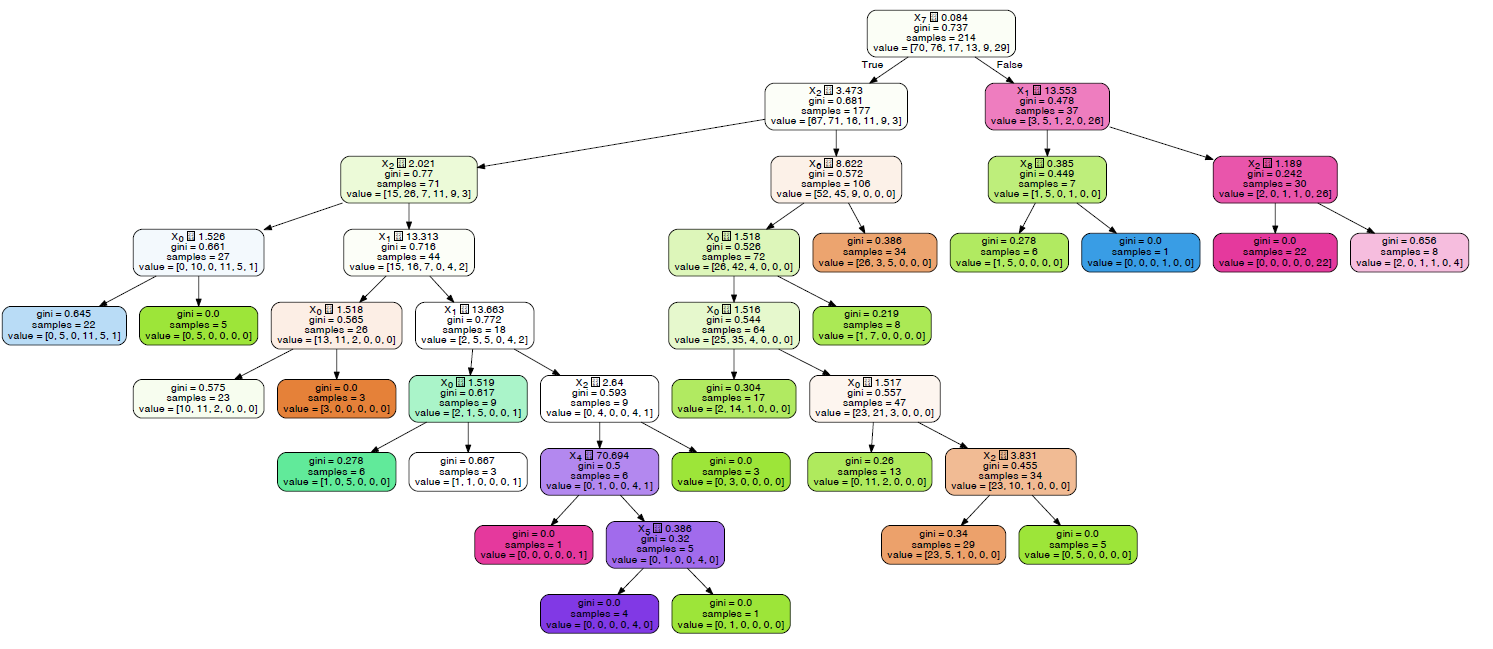
Дерево для п. 1, классификатор 1, max\_depth = 1



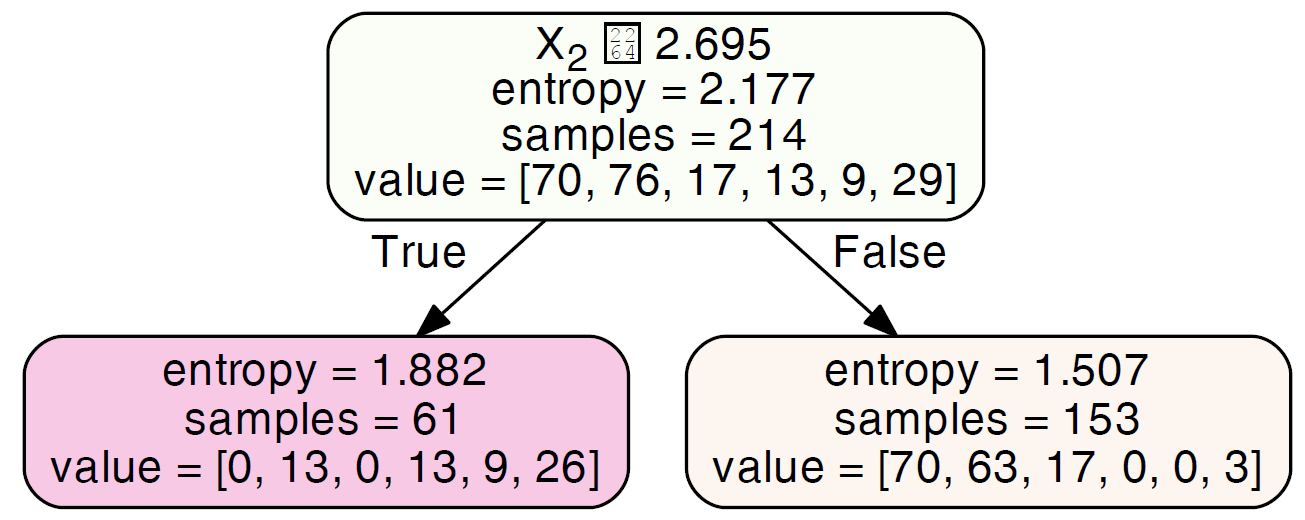
Дерево для п. 1, классификатор 1, max\_depth = 8



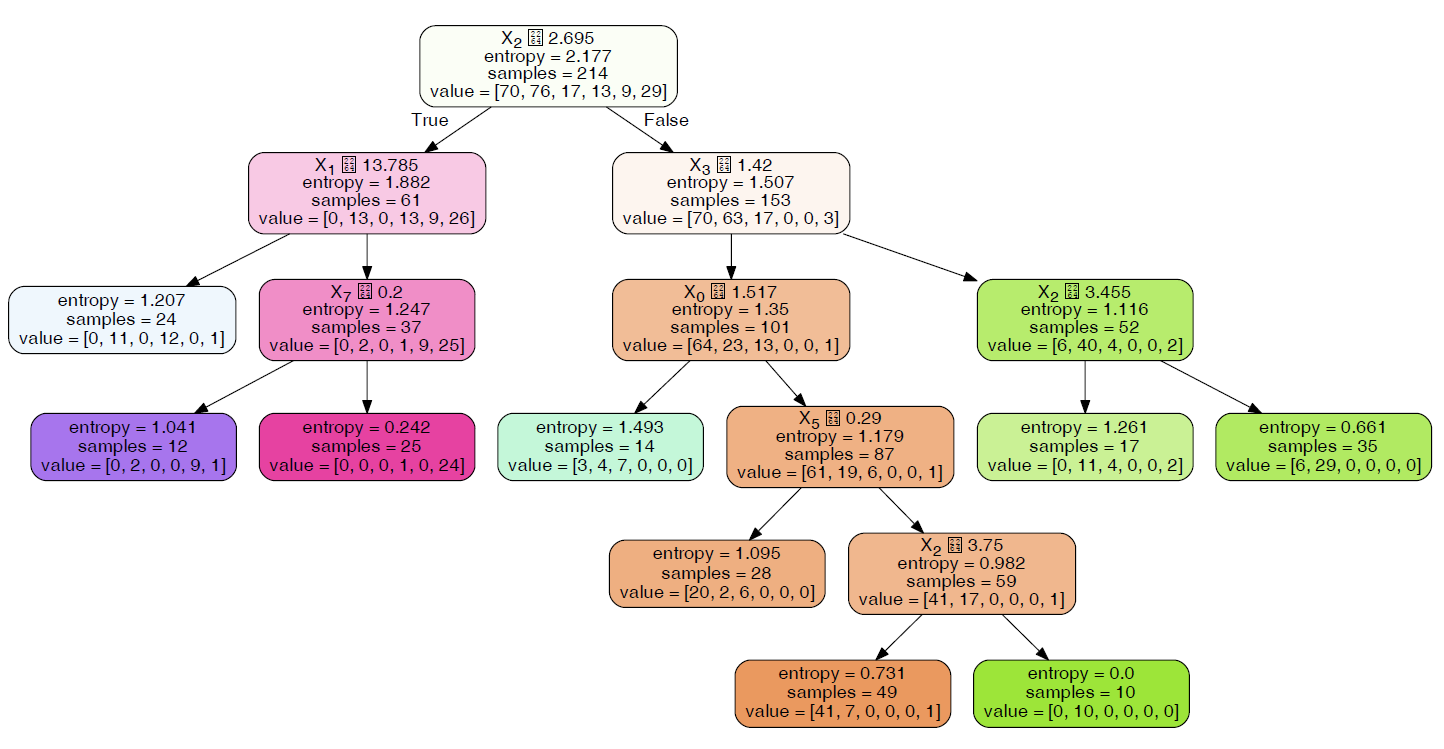
Дерево для п. 1, классификатор 1, max\_depth = 19



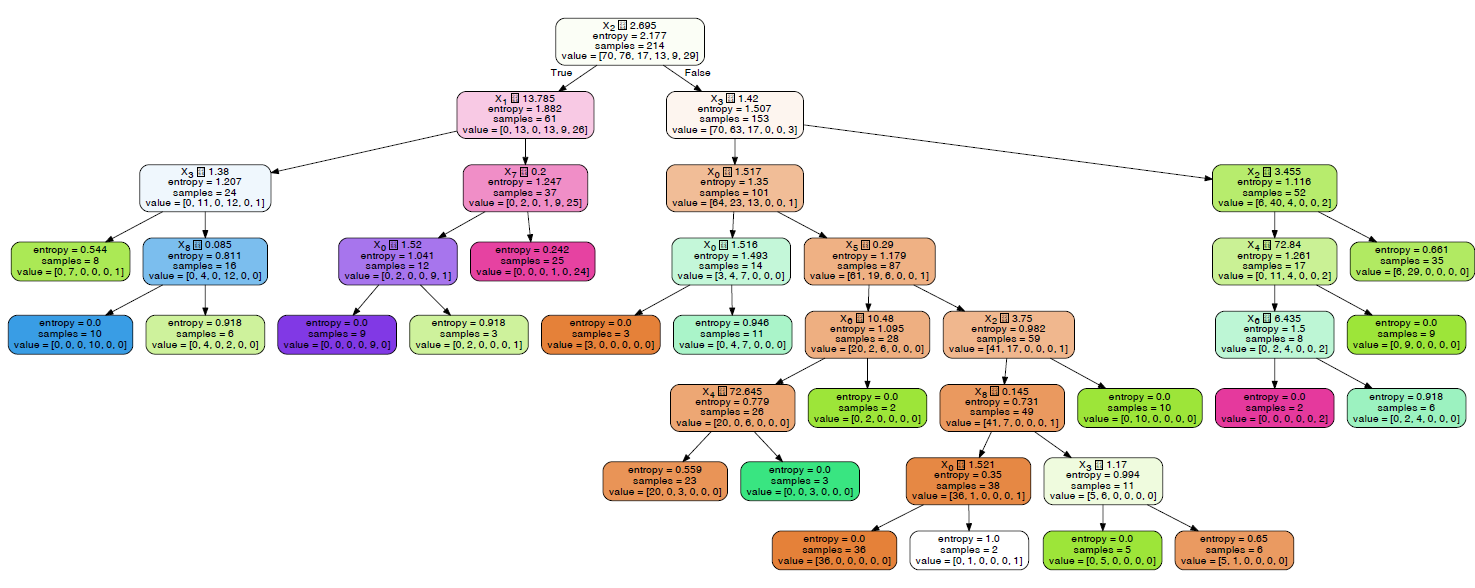
Дерево для п. 1, классификатор 2, max\_depth = 1



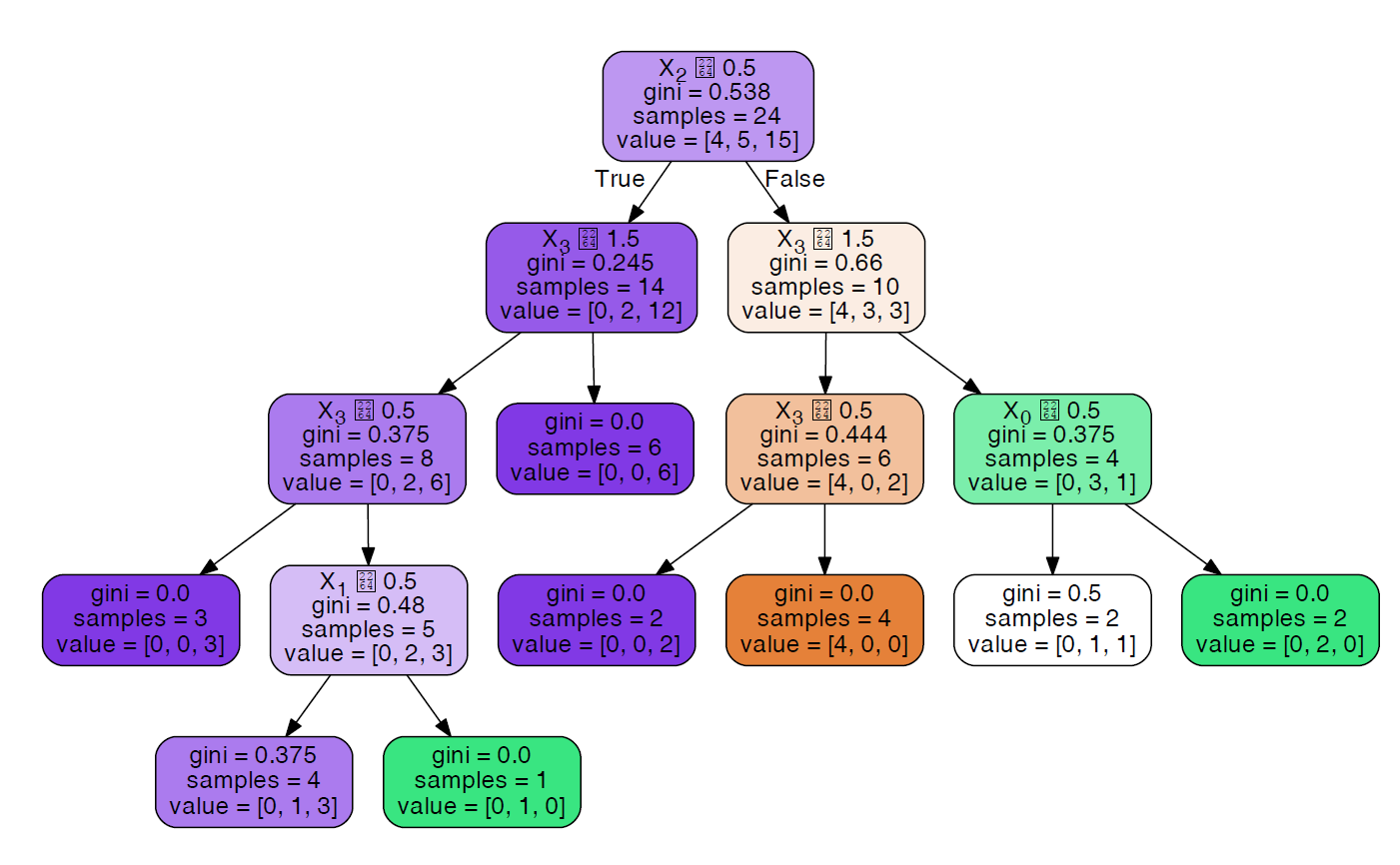
Дерево для п. 1, классификатор 2, max\_depth = 8



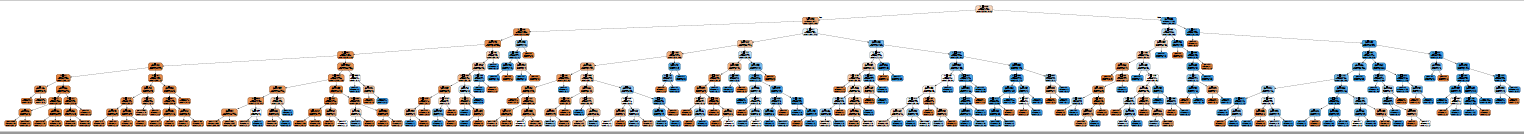
Дерево для п. 1, классификатор 2, max\_depth = 19



Дерево для п2, набор данных lenses



Дерево для п3



# Вывод

В п. 1 точность предсказания растет при увеличении параметра max\_depth. Максимальная точность достигается во 2 классификаторе при max\_depth = 19

В п. 2 был предсказан тип 3, при этом точность предсказания составляет около 90%

В п.3 оптимальное дерево было построено с помощью GridSearchCV.

# Текст программы

import os  
from matplotlib import pyplot as plt  
from sklearn import tree  
import graphviz  
import pandas as pd  
from sklearn import preprocessing  
from sklearn.model\_selection import GridSearchCV  
from sklearn import metrics  
  
  
def glass():  
 df = pd.read\_csv('glass.csv', sep=',')  
 df = df.drop('Id', 1)  
 data = df.drop('Type', 1).values  
 target = df['Type'].values  
 le = preprocessing.LabelEncoder()  
 targets\_encoded = le.fit\_transform(target)  
 for i in range(1, 20):  
 clf = tree.DecisionTreeClassifier(max\_depth=i, max\_leaf\_nodes=i+1, splitter='random')  
 clf = clf.fit(data, targets\_encoded)  
 dot\_data = tree.export\_graphviz(clf, out\_file=None,  
 filled=True, rounded=True,  
 special\_characters=True)  
 graph = graphviz.Source(dot\_data)  
 graph.render(os.getcwd() + '\\glasses\\Glass{0}'.format(i))  
 print('accuracy: {0}; max\_depth = {1}'.format(metrics.accuracy\_score(targets\_encoded, clf.predict(data)), i))  
 print('\*\*\*\*\*\*')  
 for i in range(1, 20):  
 clf = tree.DecisionTreeClassifier(max\_depth=i, max\_leaf\_nodes=i+1, criterion='entropy')  
 clf = clf.fit(data, targets\_encoded)  
 dot\_data = tree.export\_graphviz(clf, out\_file=None,  
 filled=True, rounded=True,  
 special\_characters=True)  
 graph = graphviz.Source(dot\_data)  
 graph.render(os.getcwd() + '\\glasses1\\Glass{0}'.format(i))  
 print('accuracy: {0}; max\_depth = {1}'.format(metrics.accuracy\_score(targets\_encoded, clf.predict(data)), i))  
  
  
  
def lenses():  
 clf = tree.DecisionTreeClassifier()  
 data = []  
 target = []  
 with open('Lenses.txt', 'r') as inp\_f:  
 for i in inp\_f:  
 data.append(i.split(' ')[1:5])  
 target.append(i[-2::].strip())  
 le = preprocessing.LabelEncoder()  
 data\_enc = [le.fit\_transform(i) for i in data]  
 target\_enc = le.fit\_transform(target)  
 clf = clf.fit(data\_enc, target\_enc)  
 dot\_data = tree.export\_graphviz(clf, out\_file=None,  
 filled=True, rounded=True,  
 special\_characters=True)  
 graph = graphviz.Source(dot\_data)  
 graph.render('Lenses')  
 example = [[2, 1, 1, 1]]  
 example\_enc = [le.fit\_transform(i) for i in example]  
 reslt = clf.predict(example\_enc)[0] + 1  
 print('Predicted\_type: {0}'.format(reslt))  
 print('accuracy: {0}'.format(metrics.accuracy\_score(target\_enc, clf.predict(data\_enc))))  
  
  
def spam7():  
 df = pd.read\_csv('spam7.csv', sep=',')  
 data = df.drop('yesno', 1).values  
 target = df['yesno'].values  
 le = preprocessing.LabelEncoder()  
 target\_enc = le.fit\_transform(target)  
 #parameters = {'criterion': ('gini', 'entropy'), 'splitter': ('best', 'random'), 'max\_depth': [1, 10]}  
 #tree\_clf = tree.DecisionTreeClassifier()  
 #clf = GridSearchCV(tree\_clf, parameters, cv=5)  
 clf = tree.DecisionTreeClassifier(criterion='gini', max\_depth=10, min\_samples\_leaf=1, min\_samples\_split=2, splitter='best')  
 clf.fit(data, target\_enc)  
 #print(clf.best\_estimator)  
 dot\_data = tree.export\_graphviz(clf, out\_file=None,  
 filled=True, rounded=True,  
 special\_characters=True)  
 graph = graphviz.Source(dot\_data)  
 graph.render('spam7')  
  
  
  
os.environ["PATH"] += os.pathsep + os.getcwd() + '\\release\\bin\\'  
glass()  
lenses()  
spam7()