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?

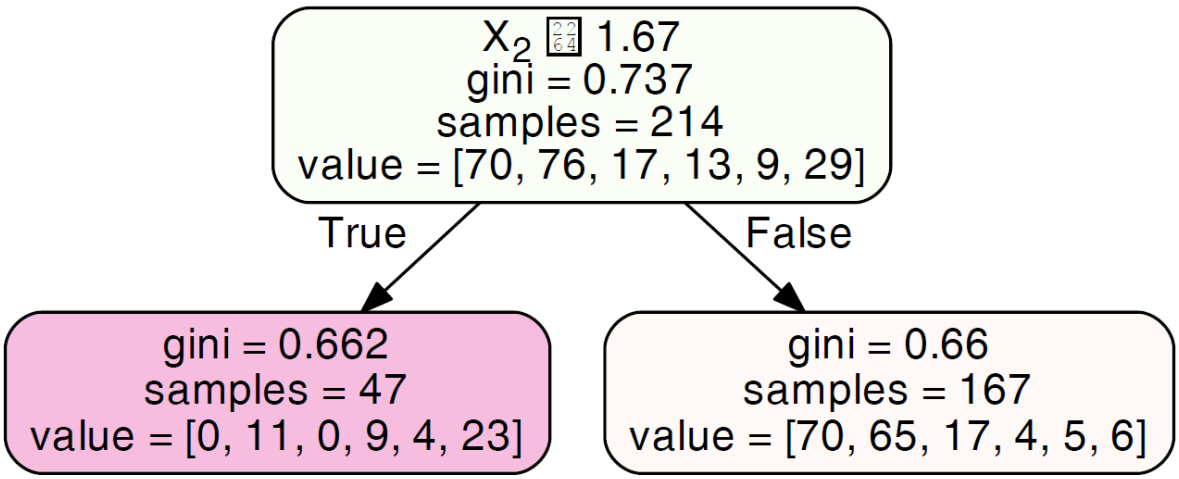
3) Загрузите набор данных spam7 из файла spam7.csv. Постройте оптимальное, по вашему мнению, дерево классификации для параметра yesno. Объясните, как был осуществлён подбор параметров.

Ход работы

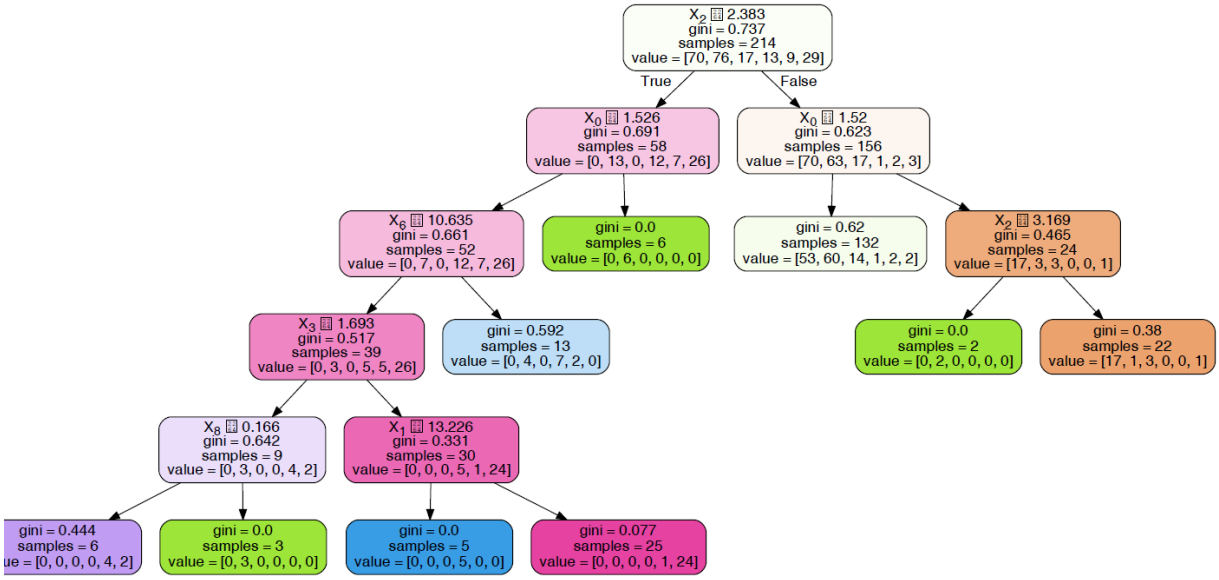
1. Для теста были выбраны классификаторы с двумя параметрами:
    a. Классификатор 1
        i. Max_depth – в диапазоне от 1 до 19
        ii. Max_leaf_nodes – в диапазоне от 2 до 20
        iii. Criterion - gini
        iv. Splitter – random
    b. Классификатор 2
        i. Max_depth – в диапазоне от 1 до 19
        ii. Max_leaf_nodes – в диапазоне от 2 до 20
        iii. Criterion - entropy
        iv. 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:

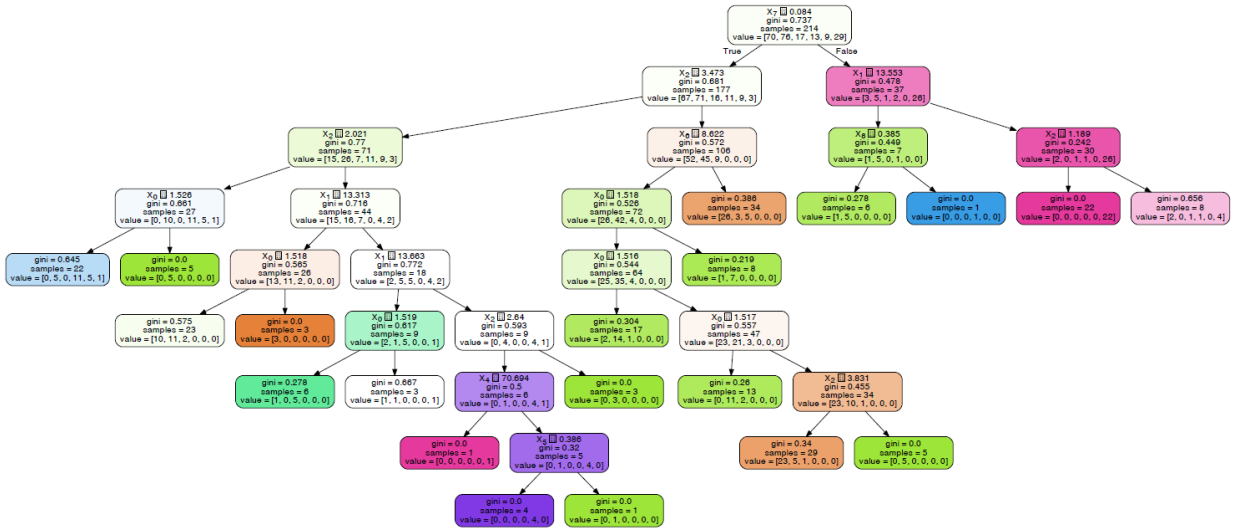
Дерево для п. 1, классификатор 1, max_depth = 1



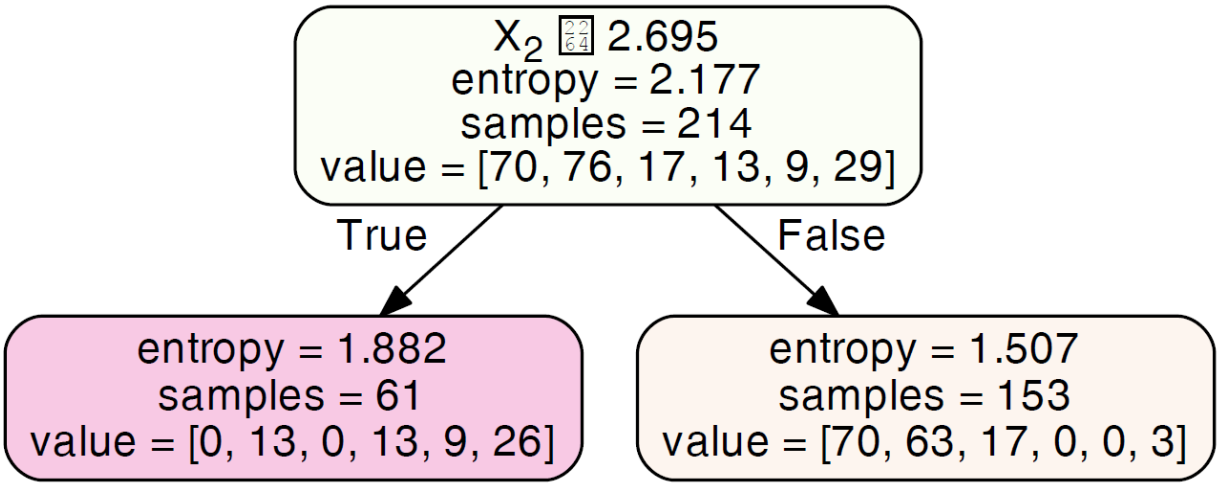
Дерево для п. 1, классификатор 1, max_depth = 8



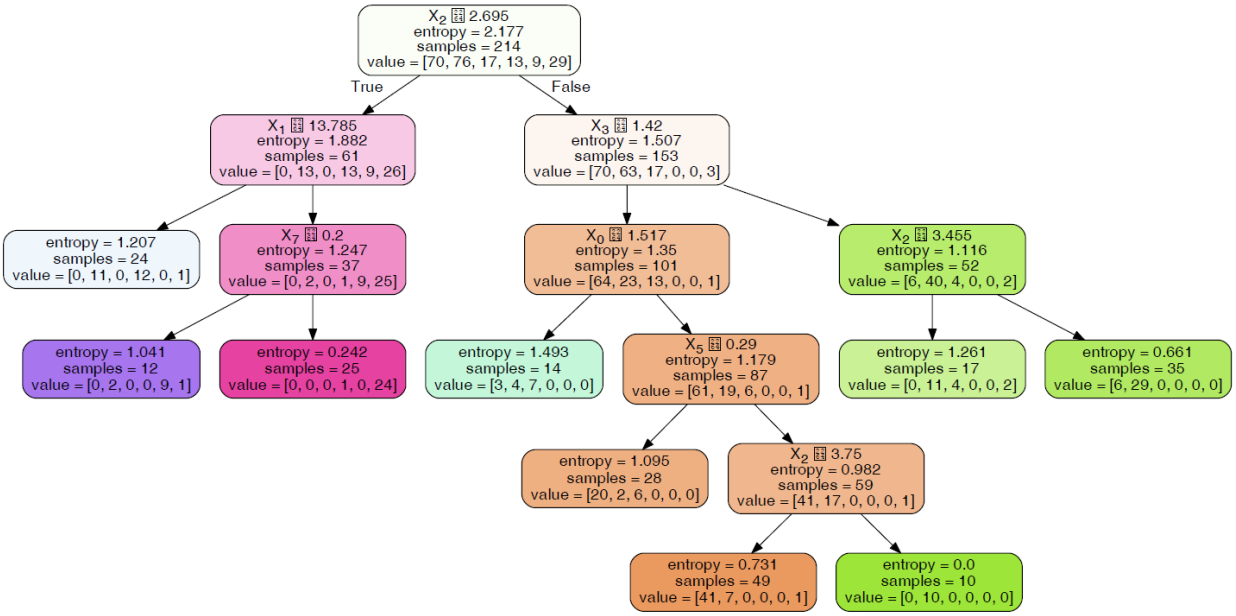
Дерево для п. 1, классификатор 1, max_depth = 19



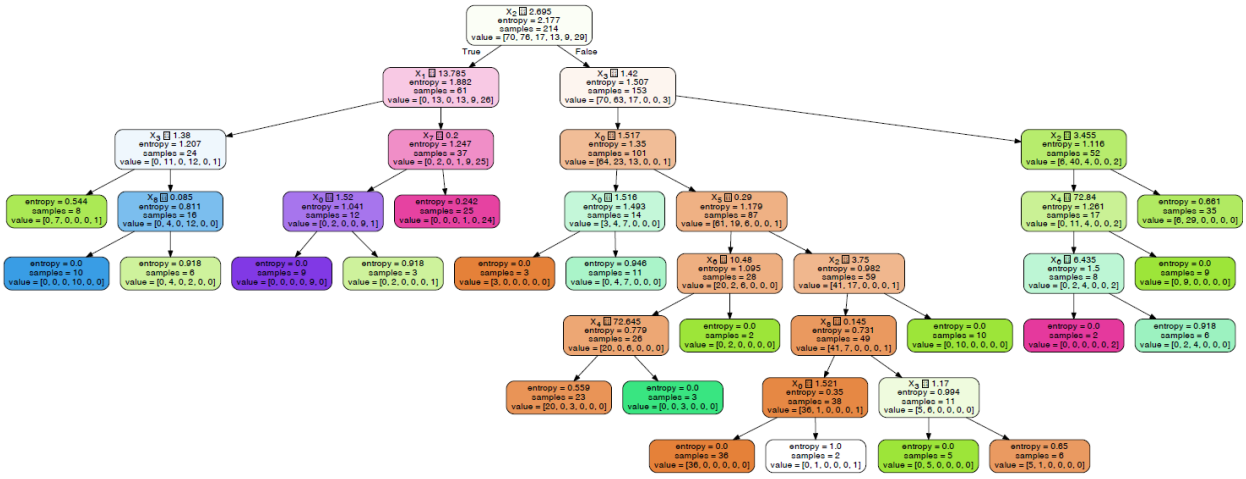
Дерево для п. 1, классификатор 2, max_depth = 1



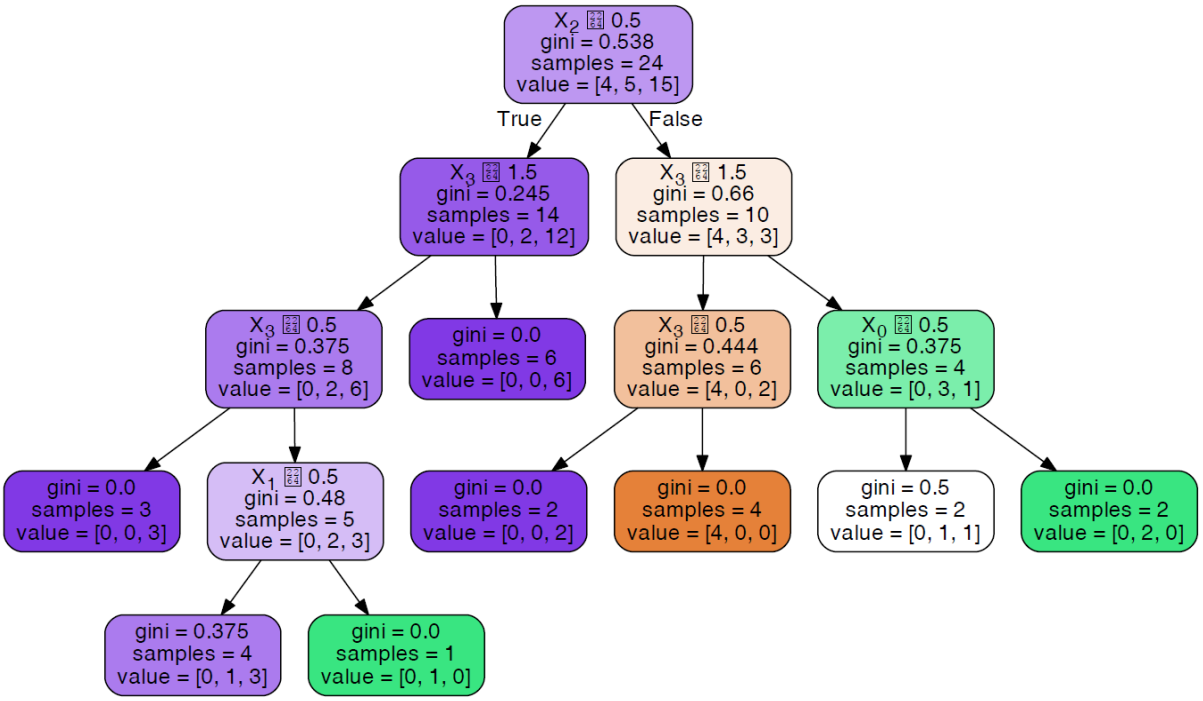
Дерево для п. 1, классификатор 2, max_depth = 8



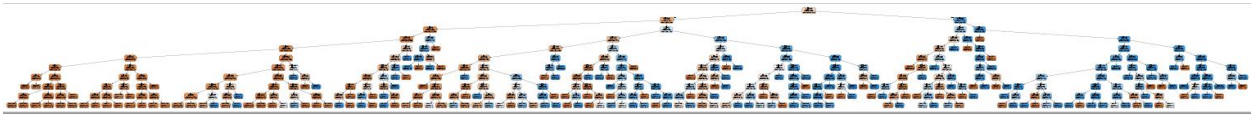
Дерево для п. 1, классификатор 2, max_depth = 19



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()
```