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

Лабораторная работа №3

по дисциплине «Машинное обучение»

Выплолнил студент
гр. 33534/5 Стойкоски Н.С.

Руководитель И.А. Селин

Санкт-Петербург
2019 г.

Оглавление

Постановка задачи

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. Был создан набор функций которые исследуют зависимостей точности классификации от параметров:

Функция train_test_check принимает исходная выборка, параметр который хочем исследовать, и значения который он принимает. Многократно строим классификаторы с варированием требуемого параметра, при этом точность оценивается с использованием функции sklearn.model_selection.cross_validate, которая даёт точность на тестирующую и тренировочную выборку. По данным строится график зависимости.

Функция check2d так же исследует заданный параметр классификатора, при этом строются две отдельние функции с разных критерии расщепления (gini, entropy).

Функция check3d исследует взаимное влияние на точности классификации двух заданных параметров, при многократного их варирование. Результат выводитеся в трехмерном пространстве с использованием функции matplotlib.plot_trisurf.

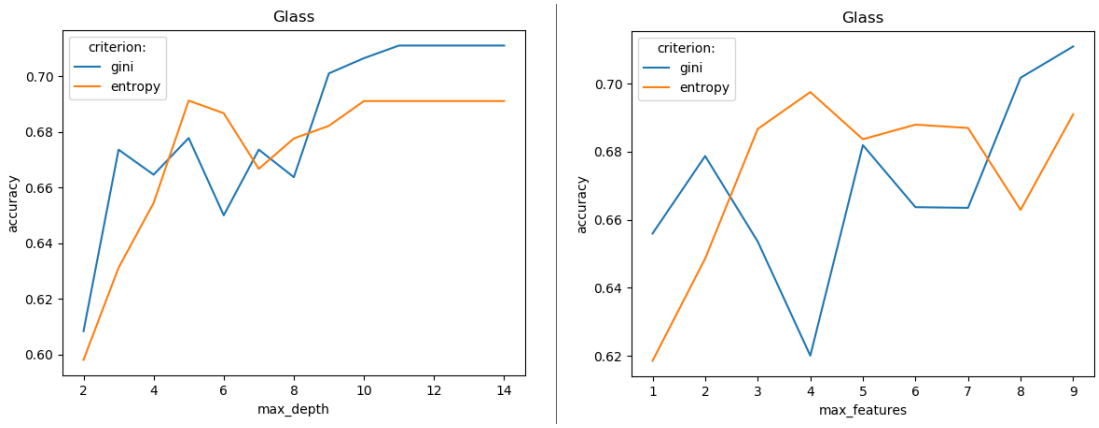
Эти функции используются для исследовние влияния на точностью параметров, такие как критерии расщепления, максимальная глубина дерева, максимальное количество рассматриваемых признаков при поиска лучшего расщепления, минимальное количество образцов для расщепления, минимальное количество образцов которые могут быть в листовой узел. Для выбора найлучшых параметров удобно использовать функцию check3d.

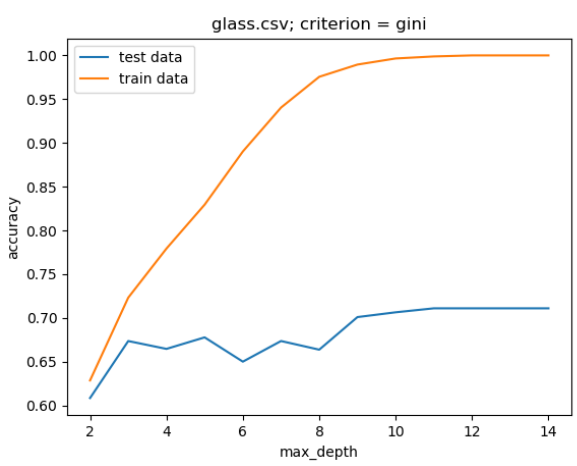
Можно сделать вывод что на наборе glass.csv дерево классификации, построенное со всех параметров по умолчанию (glass.pdf), содержащее более 100 узлов (точность = 0.710) является избыточное, так как мы можем получить одна и та же точность классификации при установление параметров: criterion='gini', max_depth=8, max_leaf_nodes=15 в результате чего имеем дерево (glass_md8_ml15.pdf) содержащее 30 узлов (точность = 0.710). Максимальная точность классификации (точность = 0.733) была достигнута при параметры: criterion='entropy', max_features=7, max_depth=6.

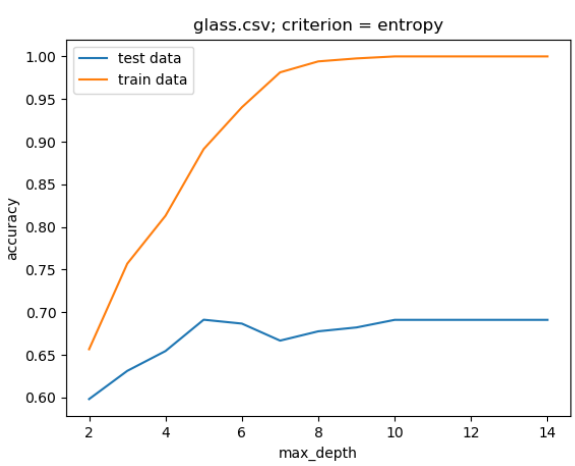
2. На наборе Lenses было построено дерево классификации, при этом для работы с категориальными даннымы (возможность более четкой интерпретации построенного дерева) была использована функция sklearn.preprocessing.OneHotEncoder которая заменяет категориальных признаков на несколько новые бинарные признаки, каждый из которых отвечает за наличия одной из возможных категории.
С использованием построенного дерево классификации предсказуем, что при предстарческой дальнозоркости, близорукости, при наличии астигматизма и сокращенной слезы пациенту не следует носить контактные линзы.

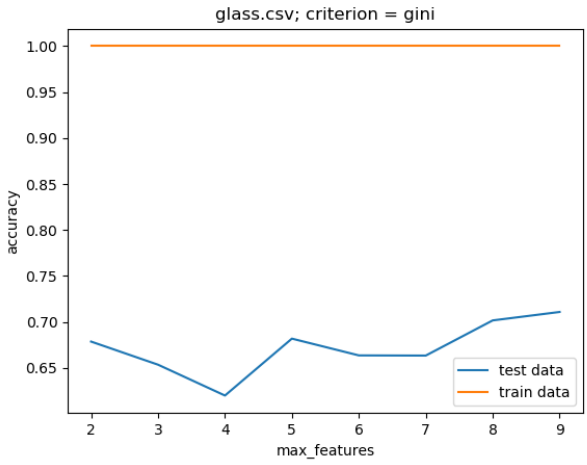
3. На наборе данных spam7 был осуществлен подбор лучших параметров с использованием функции sklearn.model_selection.GridSearchCV.
В результате получено:
параметры: {'criterion': 'gini', 'max_depth': 7, 'max_features': 4}
точность классификации: 0.876

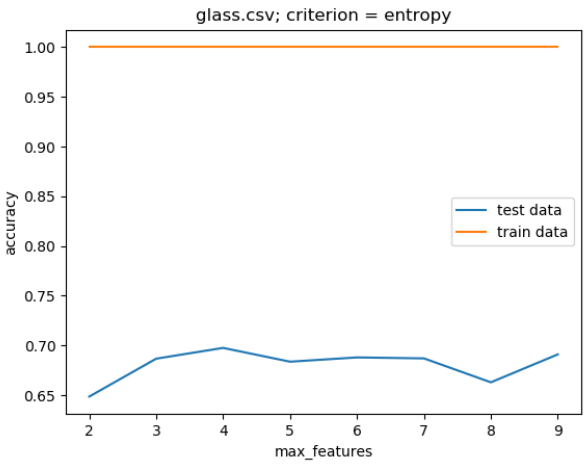
Результаты

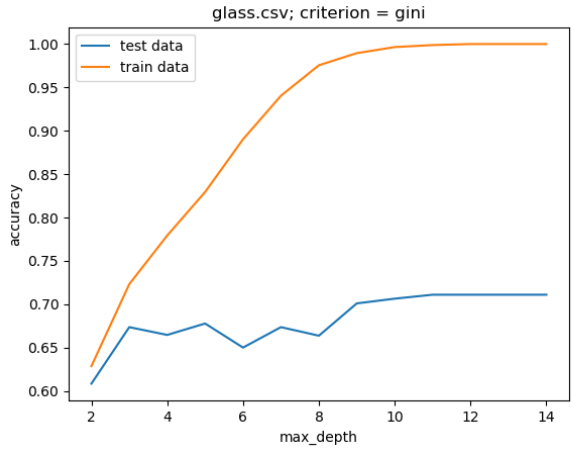


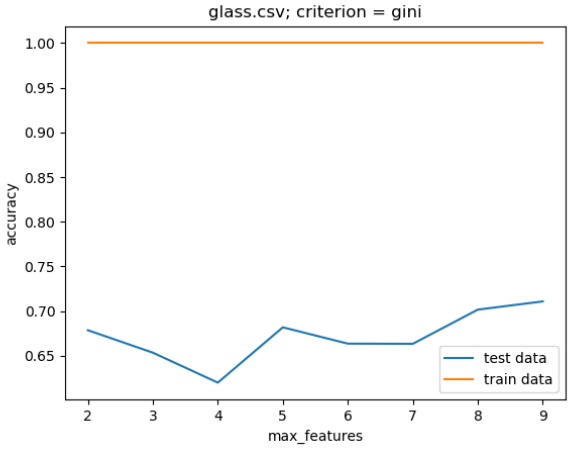

glass.csv; criterion = gini
test data
train data
accuracy
max_depth

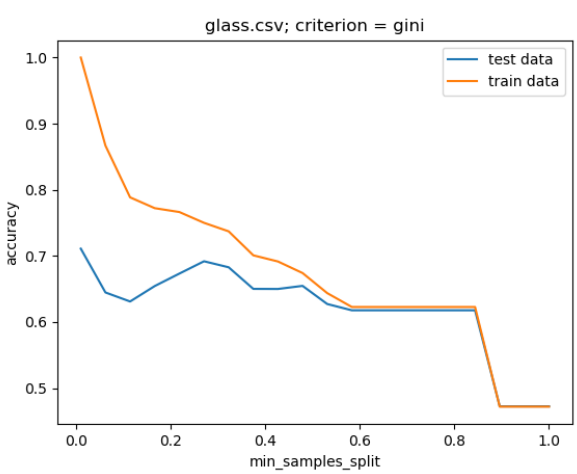

glass.csv; criterion = entropy
test data
train data
accuracy
max_depth

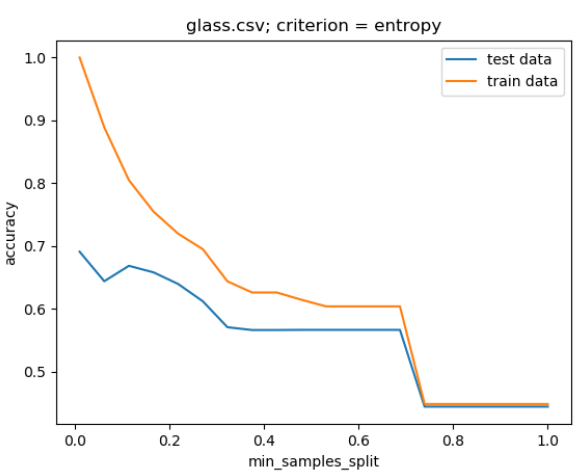

glass.csv; criterion = gini
test data
train data
accuracy
max_features

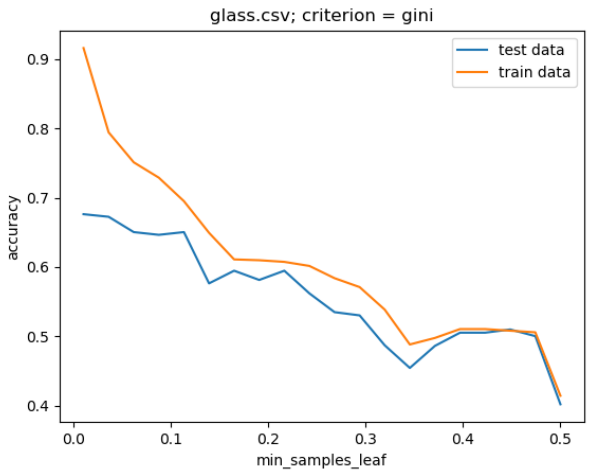

glass.csv; criterion = entropy
test data
train data
accuracy
max_features

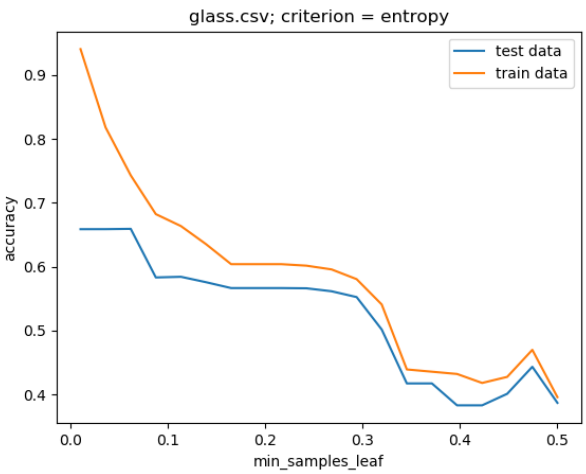

glass.csv; criterion = gini
test data
train data
accuracy
max_depth

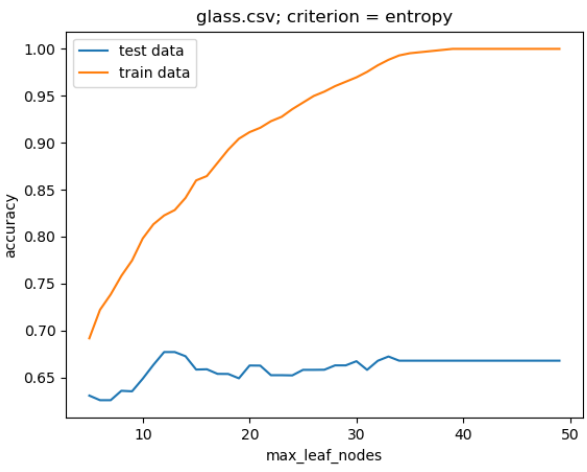

glass.csv; criterion = gini
test data
train data
accuracy
max_features

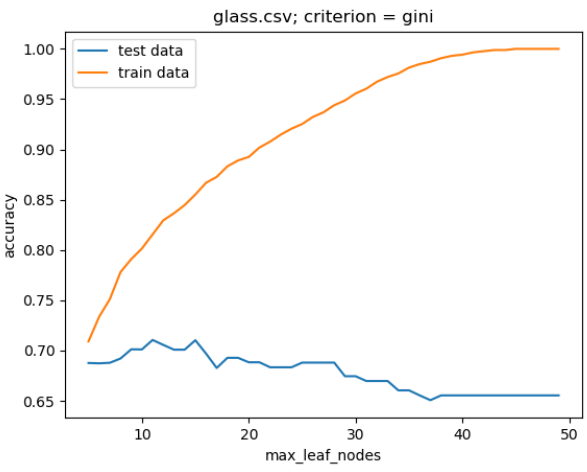

glass.csv; criterion = gini
test data
train data
accuracy
min_samples_split

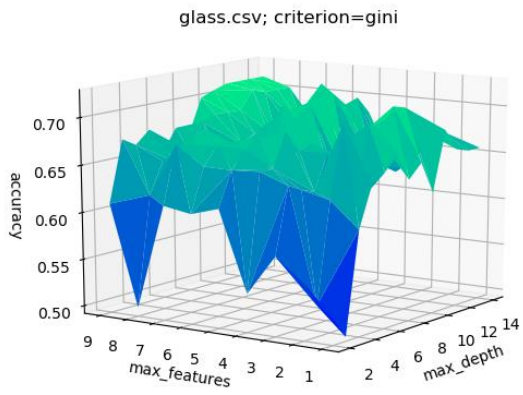

glass.csv; criterion = entropy
test data
train data
accuracy
min_samples_split

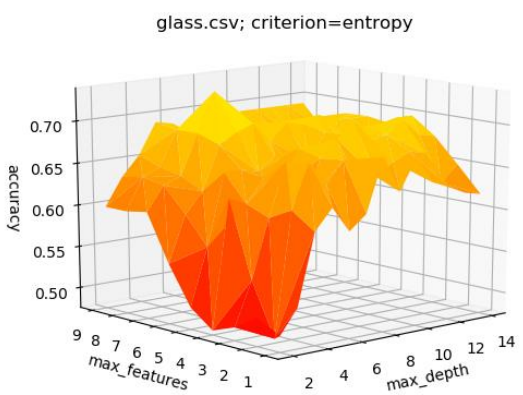

glass.csv; criterion = gini
test data
train data
accuracy
min_samples_leaf

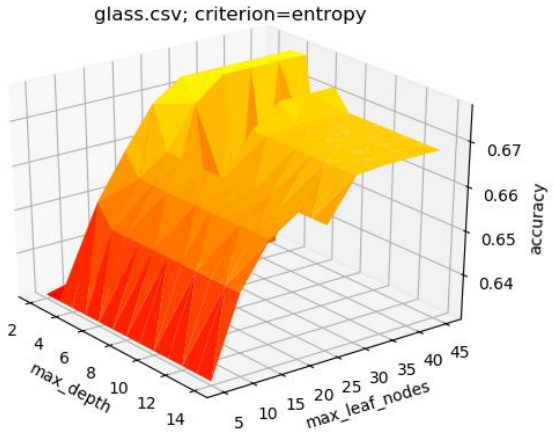

glass.csv; criterion = entropy
test data
train data
accuracy
min_samples_leaf

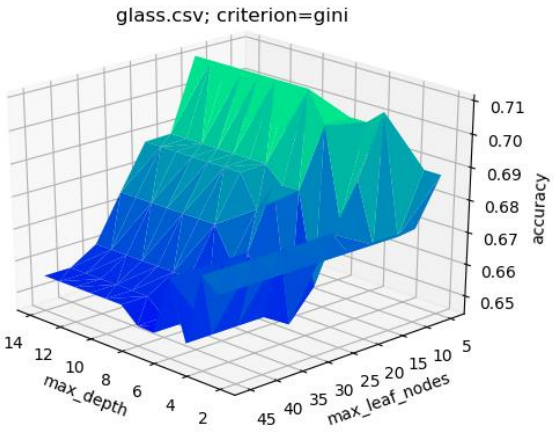

glass.csv; criterion = entropy
test data
train data
accuracy
max_leaf_nodes

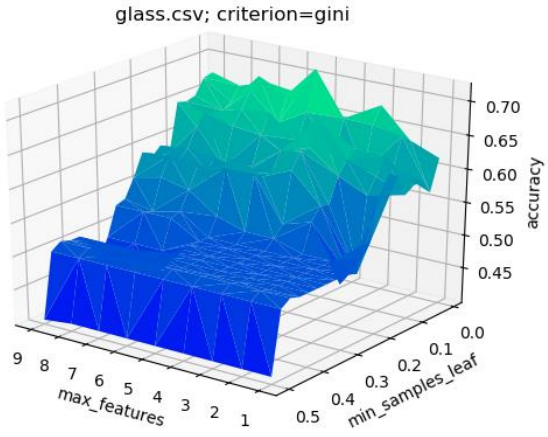

glass.csv; criterion = gini
test data
train data
accuracy
max_leaf_nodes

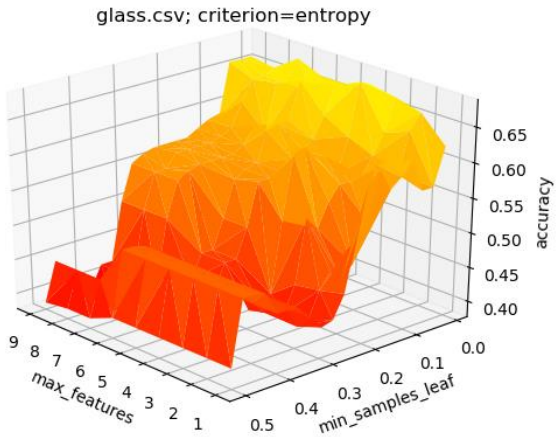

glass.csv; criterion=gini
accuracy
0.70
0.65
0.60
0.55
0.50
9 8 7 6 5 4 3 2 1
max_features
2 4 6 8 10 12 14
max_depth

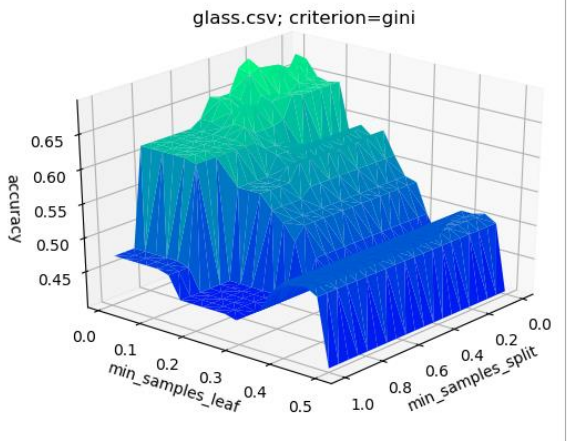

glass.csv; criterion=entropy
accuracy
0.70
0.65
0.60
0.55
0.50
9 8 7 6 5 4 3 2 1
max_features
2 4 6 8 10 12 14
max_depth

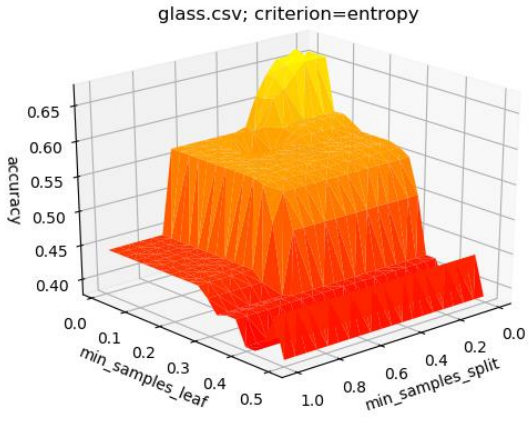

glass.csv; criterion=entropy
accuracy
0.67
0.66
0.65
0.64
2 4 6 8 10 12 14
max_depth
5 10 15 20 25 30 35 40 45
max_leaf_nodes

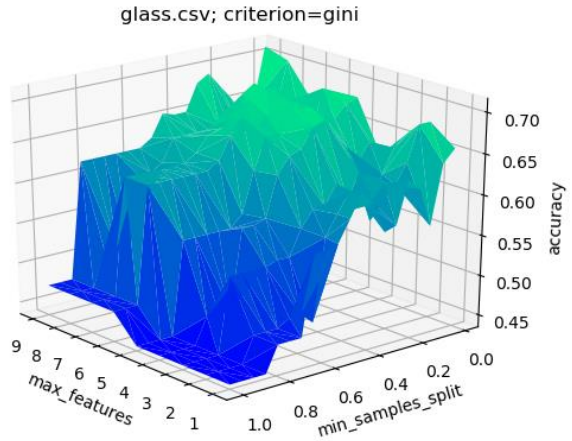

glass.csv; criterion=gini
accuracy
0.71
0.70
0.69
0.68
0.67
0.66
0.65
14 12 10 8 6 4 2
max_depth
45 40 35 30 25 20 15 10 5
max_leaf_nodes

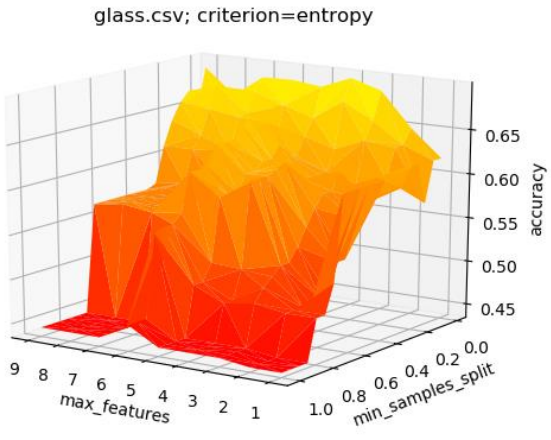

glass.csv; criterion=gini
accuracy
0.70
0.65
0.60
0.55
0.50
0.45
9 8 7 6 5 4 3 2 1
max_features
0.5 0.4 0.3 0.2 0.1 0.0
min_samples_leaf

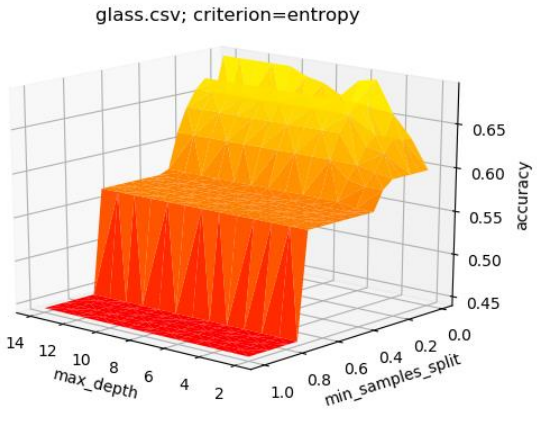

glass.csv; criterion=entropy
accuracy
0.65
0.60
0.55
0.50
0.45
0.40
9 8 7 6 5 4 3 2 1
max_features
0.5 0.4 0.3 0.2 0.1 0.0
min_samples_leaf

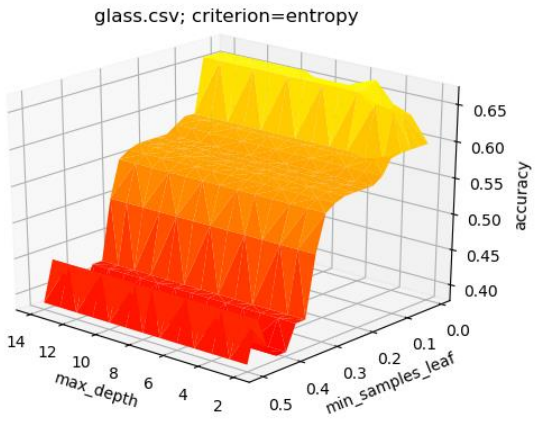

glass.csv; criterion=gini
accuracy
0.65
0.60
0.55
0.50
0.45
0.0 0.1 0.2 0.3 0.4 0.5
min_samples_leaf
1.0 0.8 0.6 0.4 0.2 0.0
min_samples_split


glass.csv; criterion=entropy
accuracy
0.65
0.60
0.55
0.50
0.45
0.40
0.0 0.1 0.2 0.3 0.4 0.5
min_samples_leaf
1.0 0.8 0.6 0.4 0.2 0.0
min_samples_split


glass.csv; criterion=gini
accuracy
0.70
0.65
0.60
0.55
0.50
0.45
9 8 7 6 5 4 3 2 1
max_features
1.0 0.8 0.6 0.4 0.2 0.0
min_samples_split


glass.csv; criterion=entropy
accuracy
0.65
0.60
0.55
0.50
0.45
9 8 7 6 5 4 3 2 1
max_features
1.0 0.8 0.6 0.4 0.2 0.0
min_samples_split


glass.csv; criterion=entropy
accuracy
0.65
0.60
0.55
0.50
0.45
14 12 10 8 6 4 2
max_depth
1.0 0.8 0.6 0.4 0.2 0.0
min_samples_split


glass.csv; criterion=entropy
accuracy
0.65
0.60
0.55
0.50
0.45
0.40
14 12 10 8 6 4 2
max_depth
0.5 0.4 0.3 0.2 0.1 0.0
min_samples_leaf

Вывод

В ходе данной лабораторной работы было получен опыт работы с деревьев решении для задач классификации. Было исследовано как разные параметры для построения дерева влияют на точность классификации. Данные были представление на графики и был получен опыт работы с построением трехразмерные графики с использованием функции библиотеки matplotlib. Был получен опыт рабоы с категориальные признаки и функцию OneHotEncoder, поиск оптимальных параметров построения дервья решений, а так же и визуального отображения дерева при использованием библиотеки graphviz.

Текст программы

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn import tree
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_validate
from sklearn.model_selection import GridSearchCV
from sklearn import metrics
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm

import os
os.environ["PATH"] += os.pathsep + 'C:/Program Files/Graphviz-2.38/bin/'
import graphviz

def train_test_check(x, y, criterion, parameter, parameter_values):
    accuracies = []
    for pval in parameter_values:
        d = {parameter: pval}
        clf = tree.DecisionTreeClassifier(criterion=criterion, **d, random_state=1)
        kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)
        scores = cross_validate(clf, x, y, scoring='accuracy', return_train_score=True,
cv=kfold)
        accuracies.append((scores['test_score'].mean(), scores['train_score'].mean()))
    plt.figure()
    plt.plot(parameter_values, [acc[0] for acc in accuracies], label='test data')
    plt.plot(parameter_values, [acc[1] for acc in accuracies], label='train data')
    plt.xlabel(parameter)
    plt.ylabel('accuracy')
    plt.title(f'glass.csv; criterion = {criterion}')
    plt.legend()

def check2d(x, y, parameter, parameter_values):
    plt.figure()
```

```
    for criterion in ['gini', 'entropy']:
        accuracies = []
        for pval in parameter_values:
            d = {parameter: pval}
            kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)
            clf = tree.DecisionTreeClassifier(criterion=criterion, **d, random_state=1)
            accuracies.append(cross_val_score(clf, x, y, cv=kfold,
scoring='accuracy').mean())
        plt.plot(parameter_values, accuracies, label=criterion)
    plt.xlabel(parameter)
    plt.ylabel('accuracy')
    plt.title('Glass')
    plt.legend(title='criterion:')


def check3d(x, y, param1, param2, p1_values, p2_values, criterion='entropy',
cmap=cm.autumn):
    zvals, xvals, yvals = [], [], []
    for p1v in p1_values:
        for p2v in p2_values:
            d = {param1: p1v, param2: p2v}
            kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)
            clf = tree.DecisionTreeClassifier(criterion=criterion, **d, random_state=1)
            zvals.append(cross_val_score(clf, x, y, cv=kfold).mean())
            xvals.append(p1v)
            yvals.append(p2v)
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.plot_trisurf(xvals, yvals, zvals, cmap=cmap, vmin=min(zvals), vmax=max(zvals))
    ax.set_zlabel('accuracy')
    plt.xlabel(param1)
    plt.ylabel(param2)
    plt.title(f'glass.csv; criterion={criterion}')

def glass():
    df = pd.read_csv('glass.csv')
    x = df.iloc[:, 2:-1].values
    y = df['Type'].values

    train_test_check(x, y, 'gini', 'max_leaf_nodes', range(5, 50))
    train_test_check(x, y, 'entropy', 'max_leaf_nodes', range(5, 50))

    check3d(x,y, 'max_features', 'max_leaf_nodes', range(1, 10, 1), range(5, 50, 5))
    check3d(x,y, 'max_depth', 'max_leaf_nodes', range(2, 15), range(5, 50, 5))
    check3d(x,y, 'max_features', 'max_leaf_nodes', range(1, 10, 1), range(5, 50,
5),criterion='gini', cmap=cm.winter)
    check3d(x,y, 'max_depth', 'max_leaf_nodes', range(2, 15), range(5, 50, 5),
criterion='gini', cmap=cm.winter)

    train_test_check(x, y, 'gini', 'max_depth', range(2, 15))
    train_test_check(x, y, 'gini', 'max_features', range(2, 10))
    train_test_check(x, y, 'gini', 'min_samples_split', np.linspace(0.01, 1, 20))
    train_test_check(x, y, 'gini', 'min_samples_leaf', np.linspace(0.01, 0.5, 20))
    train_test_check(x, y, 'entropy', 'max_depth', range(2, 15))
    train_test_check(x, y, 'entropy', 'max_features', range(2, 10))
    train_test_check(x, y, 'entropy', 'min_samples_split', np.linspace(0.01, 1, 20))
    train_test_check(x, y, 'entropy', 'min_samples_leaf', np.linspace(0.01, 0.5, 20))
```

```
    check2d(x, y, 'max_depth', range(2, 15, 1))
    check2d(x, y, 'max_features', range(1, 10, 1))
    check3d(x, y, 'max_depth', 'min_samples_split', range(2, 15, 1), np.linspace(0.01, 1,
20))
    check3d(x, y, 'max_depth', 'min_samples_leaf', range(2, 15, 1), np.linspace(0.01,
0.5, 20))
    check3d(x, y, 'min_samples_split', 'min_samples_leaf', np.linspace(0.01, 1, 20),
np.linspace(0.01, 0.5, 20))
    check3d(x, y, 'min_samples_split', 'min_samples_leaf', np.linspace(0.01, 1, 20),
np.linspace(0.01, 0.5, 20),
            criterion='gini', cmap=cm.winter)
    check3d(x, y, 'max_depth', 'max_features', range(2, 15, 1), range(1, 10, 1),
            criterion='gini', cmap=cm.winter)

    #optimal
    kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)
    clf = tree.DecisionTreeClassifier(criterion='entropy', max_features=7, max_depth=6,
random_state=1)
    print('accuracy: ', cross_val_score(clf, x, y, cv=kfold).mean())
    clf = clf.fit(x, y)
    dot_data = tree.export_graphviz(clf, out_file=None,
      feature_names=df.columns.values[2:-1], class_names=['1', '2', '3', '5', '6', '7'],
      filled=True, rounded=True, special_characters=True)
    graph = graphviz.Source(dot_data)
    graph.render("glass_opt")

    #smallest
    clf = tree.DecisionTreeClassifier(criterion='gini', max_depth=8, max_leaf_nodes=15,
random_state=1)
    print('accuracy: ', cross_val_score(clf, x, y, cv=kfold).mean())
    clf = clf.fit(x, y)
    dot_data = tree.export_graphviz(clf, out_file=None,
      feature_names=df.columns.values[2:-1], class_names=['1', '2', '3', '5', '6', '7'],
      filled=True, rounded=True, special_characters=True)
    graph = graphviz.Source(dot_data)
    graph.render("glass_md8_ml15")

    #default
    clf = tree.DecisionTreeClassifier(random_state=1)
    print('accuracy: ', cross_val_score(clf, x, y, cv=kfold).mean())
    clf = clf.fit(x, y)
    dot_data = tree.export_graphviz(clf, out_file=None,
      feature_names=df.columns.values[2:-1], class_names=['1', '2', '3', '5', '6', '7'],
      filled=True, rounded=True, special_characters=True)
    graph = graphviz.Source(dot_data)
    graph.render("glass")

def lenses():
    df = pd.read_csv('Lenses.txt', delim_whitespace=True, header=None)

    x = df.iloc[:, 1:-1].values
    y = df.iloc[:, -1].values

    clf = tree.DecisionTreeClassifier()
    clf.fit(x, y)
    print(clf.predict([[2, 1, 2, 1]]))

    dot_data = tree.export_graphviz(clf, out_file=None,
```

```
        feature_names=['возраст', 'состояние зрения', 'астигматизм', 'состояние слезы'],
        class_names=['жесткие линзы', 'мягкие линзы', 'не следует носить линзы'],
        filled=True, rounded=True, special_characters=True)
    graph = graphviz.Source(dot_data)
    graph.render("Lenses")

    from sklearn.preprocessing import OneHotEncoder
    x = df.iloc[:, 1:-1].values
    y = df.iloc[:, -1].values

    enc = OneHotEncoder(handle_unknown='ignore')
    enc.fit(x)
    clf = tree.DecisionTreeClassifier()
    clf.fit(enc.transform(x).toarray(), y)

    print(clf.predict(enc.transform([[2,1,2,1]]).toarray()))

    feature_names = ['молодой', 'предстарческая дальнозоркость', 'старческая
дальнозоркость',
                     'близорукий', 'дальнозоркий',
                     'астигматизм-да', 'астигматизм-нет',
                     'слезы-сокращенная', 'слезы-нормаьная']
    dot_data = tree.export_graphviz(clf, out_file=None,
        feature_names=feature_names,
        class_names=['жесткие линзы', 'мягкие линзы', 'не следует носить линзы'],
        filled=True, rounded=True, special_characters=True)
    graph = graphviz.Source(dot_data)
    graph.render("Lenses_onehot")

def spam7():
    df = pd.read_csv('spam7.csv')
    x = df.iloc[:, 0:-1].values
    y = df['yesno'].values

    param_grid = {
        'criterion': ['gini', 'entropy'],
        'max_depth': range(1, 15),
        'max_features': range(1, 7)
    }
    clf = tree.DecisionTreeClassifier(random_state=1)
    kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)
    clf_cv = GridSearchCV(estimator=clf, param_grid=param_grid, cv=kfold)
    clf_cv.fit(x, y)
    print(clf_cv.best_params_)
    print(clf_cv.best_score_)

    kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)
    clf = tree.DecisionTreeClassifier(random_state=1)
    print('accuracy: ', cross_val_score(clf, x, y, cv=kfold).mean())

glass()
lenses()
spam7()
plt.show()
```