Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика с системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчёт по рубежному контролю №2

Выполнил:
студент группы РТ5-61Б
Андреев Виктор
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2023 г

Задание

Вариант №2, группа РТ5-61Б

Постройте модель классификации. Для построения моделей используйте методы "Дерево решений" и "Градиентный бустинг". Оцените качество моделей на основе подходящих метрик качества (не менее двух метрик).

Набор данных: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html#sklearn.datasets.load_wine

Ход работы

Загрузка датасета

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import *
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn import svm, tree
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from operator import itemgetter

def make_dataframe(ds_function):
    ds = ds_function()
    df = pd.DataFrame(data= np.c_[ds['data'], ds['target']],
                      columns= list(ds['feature_names']) + ['target'])
    return df

wine = load_wine()

df = make_dataframe(load_wine)

# Первые 5 строк датасета
df.head()
```

```
   alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  total_phenols  \
0    14.23        1.71  2.43               15.6      127.0           2.80
1    13.20        1.78  2.14               11.2      100.0           2.65
2    13.16        2.36  2.67               18.6      101.0           2.80
3    14.37        1.95  2.50               16.8      113.0           3.85
4    13.24        2.59  2.87               21.0      118.0           2.80

   flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity   hue
```

```
\
0        3.06                  0.28             2.29             5.64  1.04
1        2.76                  0.26             1.28             4.38  1.05
2        3.24                  0.30             2.81             5.68  1.03
3        3.49                  0.24             2.18             7.80  0.86
4        2.69                  0.39             1.82             4.32  1.04

   od280/od315_of_diluted_wines  proline  target
0                          3.92   1065.0     0.0
1                          3.40   1050.0     0.0
2                          3.17   1185.0     0.0
3                          3.45   1480.0     0.0
4                          2.93    735.0     0.0
```

```
df.dtypes
```

```
alcohol                         float64
malic_acid                      float64
ash                             float64
alcalinity_of_ash               float64
magnesium                       float64
total_phenols                   float64
flavanoids                      float64
nonflavanoid_phenols            float64
proanthocyanins                 float64
color_intensity                 float64
hue                             float64
od280/od315_of_diluted_wines    float64
proline                         float64
target                          float64
dtype: object
```

Все значения имеют тип float64, поэтому нет необходимости в кодировании категориальных признаков

```
# Проверим наличие пустых значений
# Цикл по колонкам датасета
for col in df.columns:
    # Количество пустых значений - все значения заполнены
    temp_null_count = df[df[col].isnull()].shape[0]
    print('{} - {}'.format(col, temp_null_count))
```

```
alcohol - 0
malic_acid - 0
ash - 0
alcalinity_of_ash - 0
magnesium - 0
total_phenols - 0
flavanoids - 0
nonflavanoid_phenols - 0
proanthocyanins - 0
```

```
color_intensity - 0
hue - 0
od280/od315_of_diluted_wines - 0
proline - 0
target - 0
```

Пустых значений нет, поэтому нет необходимости заполнять пропуски

Разделение на тестовую и обучающую выборки

```
y = df['target']
x = df.drop('target', axis = 1)

scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(x)

x_train, x_test, y_train, y_test = train_test_split(scaled_data, y, test_size
= 0.2, random_state = 0)

print(f"Обучающая выборка:\n{x_train, y_train}")
print(f"Тестовая выборка:\n{x_test, y_test}")
```

```
Обучающая выборка:
(array([[0.7       , 0.49802372, 0.63101604, ..., 0.3902439 , 0.2014652 ,
        0.28673324],
       [0.43684211, 0.15612648, 0.48128342, ..., 0.3902439 , 0.28937729,
        0.15477889],
       [0.15526316, 0.24703557, 0.49197861, ..., 0.55284553, 0.61904762,
        0.04778887],
       ...,
       [0.36578947, 0.17193676, 0.44385027, ..., 0.47154472, 0.61904762,
        0.04778887],
       [0.75526316, 0.18577075, 0.40641711, ..., 0.3495935 , 0.75457875,
        0.5042796 ],
       [0.82368421, 0.34980237, 0.59893048, ..., 0.11382114, 0.16117216,
        0.2724679 ]]), 161    2.0
92     1.0
94     1.0
174    2.0
24     0.0
      ...
103    1.0
67     1.0
117    1.0
47     0.0
172    2.0
Name: target, Length: 142, dtype: float64)
Тестовая выборка:
(array([[0.71315789, 0.18379447, 0.47593583, 0.29896907, 0.52173913,
        0.55862069, 0.54008439, 0.1509434 , 0.38170347, 0.38993174,
        0.35772358, 0.70695971, 0.55777461],
```

```
[0.46315789, 0.38142292, 0.59893048, 0.58762887, 0.45652174,
 0.17241379, 0.21518987, 0.20754717, 0.2681388 , 0.81228669,
 0.        , 0.07326007, 0.14407989],
[0.35263158, 0.0770751 , 0.42780749, 0.43298969, 0.18478261,
 0.86896552, 0.58227848, 0.11320755, 0.46056782, 0.27047782,
 0.60162602, 0.58608059, 0.10128388],
[0.66578947, 0.19565217, 0.58823529, 0.51030928, 0.5       ,
 0.68275862, 0.51476793, 0.13207547, 0.64353312, 0.42406143,
 0.40650407, 0.64468864, 0.60057061],
[0.53157895, 1.        , 0.41176471, 0.56185567, 0.17391304,
 0.56551724, 0.48734177, 0.32075472, 0.50473186, 0.11262799,
 0.20325203, 0.67032967, 0.07275321],
[0.13947368, 0.25889328, 1.        , 0.92268041, 0.5326087 ,
 0.75862069, 1.        , 0.64150943, 0.46056782, 0.40273038,
 0.36585366, 0.88644689, 0.13338088],
[0.79736842, 0.27865613, 0.6684492 , 0.36082474, 0.55434783,
 0.55862069, 0.45780591, 0.33962264, 0.26498423, 0.32167235,
 0.47154472, 0.84615385, 0.7253923 ],
[0.35      , 0.61067194, 0.54545455, 0.53608247, 0.19565217,
 0.45517241, 0.12236287, 0.69811321, 0.19873817, 0.54351536,
 0.06504065, 0.11355311, 0.17261056],
[0.32105263, 0.19565217, 0.40641711, 0.43298969, 0.10869565,
 0.23103448, 0.35654008, 0.45283019, 0.38485804, 0.18088737,
 0.42276423, 0.6959707 , 0.16547789],
[0.27631579, 0.21541502, 0.51336898, 0.40721649, 0.11956522,
 0.2137931 , 0.24472574, 0.73584906, 0.38801262, 0.09556314,
 0.48780488, 0.36630037, 0.14407989],
[0.61315789, 0.35968379, 0.52941176, 0.48453608, 0.20652174,
 0.14482759, 0.03375527, 0.45283019, 0.07255521, 0.36860068,
 0.17886179, 0.43956044, 0.35805991],
[0.75      , 0.84980237, 0.46524064, 0.48453608, 0.10869565,
 0.        , 0.        , 0.50943396, 0.0851735 , 0.30887372,
 0.08130081, 0.02197802, 0.09771755],
[0.83421053, 0.20158103, 0.5828877 , 0.2371134 , 0.45652174,
 0.78965517, 0.64345992, 0.39622642, 0.49211356, 0.46672355,
 0.46341463, 0.57875458, 0.83594864],
[0.35263158, 0.06521739, 0.39572193, 0.40721649, 0.19565217,
 0.87586207, 0.71940928, 0.20754717, 0.48580442, 0.27474403,
 0.45528455, 0.54945055, 0.2724679 ],
[0.67105263, 0.36363636, 0.71122995, 0.71649485, 0.38043478,
 0.19655172, 0.10548523, 0.49056604, 0.35646688, 0.62969283,
 0.21138211, 0.19413919, 0.33666191],
[0.25526316, 0.03557312, 0.34224599, 0.43298969, 0.17391304,
 0.49655172, 0.40506329, 0.32075472, 0.32176656, 0.10409556,
 0.73170732, 0.67765568, 0.        ],
[0.71842105, 0.15612648, 0.71657754, 0.45876289, 0.67391304,
 0.67931034, 0.50632911, 0.69811321, 0.29652997, 0.35153584,
 0.62601626, 0.63369963, 0.68259629],
[0.83157895, 0.16798419, 0.59893048, 0.30412371, 0.41304348,
 0.8       , 0.75738397, 0.35849057, 0.45741325, 0.6331058 ,
```

```
 0.6097561 , 0.56776557, 1.        ],
[0.42368421, 0.12252964, 0.35294118, 0.31958763, 0.32608696,
 0.35862069, 0.2257384 , 0.75471698, 0.06624606, 0.38139932,
 0.40650407, 0.11721612, 0.12268188],
[0.73684211, 0.1798419 , 0.6631016 , 0.34020619, 0.26086957,
 0.50689655, 0.55907173, 0.16981132, 0.59305994, 0.36860068,
 0.61788618, 0.76923077, 0.70399429],
[0.54736842, 0.05335968, 0.18181818, 0.22680412, 0.08695652,
 0.68965517, 0.59915612, 0.24528302, 0.58990536, 0.34300341,
 0.5203252 , 0.6996337 , 0.15977175],
[0.53157895, 0.1798419 , 0.63636364, 0.3814433 , 0.30434783,
 0.50689655, 0.44092827, 0.30188679, 0.32492114, 0.25341297,
 0.5203252 , 0.45421245, 0.58987161],
[0.58157895, 0.36561265, 0.80748663, 0.53608247, 0.52173913,
 0.62758621, 0.49578059, 0.49056604, 0.44479495, 0.25938567,
 0.45528455, 0.60805861, 0.32596291],
[0.38947368, 0.19565217, 0.3315508 , 0.51030928, 0.16304348,
 0.42068966, 0.33333333, 0.35849057, 0.33753943, 0.14163823,
 0.45528455, 0.84249084, 0.2810271 ],
[0.34210526, 0.07114625, 0.49197861, 0.27835052, 0.33695652,
 0.36896552, 0.15822785, 0.94339623, 0.        , 0.16979522,
 0.62601626, 0.14652015, 0.28673324],
[0.39210526, 0.33399209, 0.43315508, 0.53608247, 0.19565217,
 0.54137931, 0.407173  , 0.24528302, 0.2555205 , 0.06143345,
 0.34146341, 0.55311355, 0.03352354],
[0.36842105, 0.15612648, 0.4973262 , 0.56185567, 0.17391304,
 0.60689655, 0.592827  , 0.49056604, 0.42902208, 0.22696246,
 0.17073171, 0.57509158, 0.05278174],
[0.29736842, 0.17193676, 0.50802139, 0.62886598, 0.2173913 ,
 0.27586207, 0.28481013, 0.56603774, 0.36277603, 0.09982935,
 0.69105691, 0.36263736, 0.15477889],
[0.19210526, 0.38339921, 0.8342246 , 0.48453608, 0.35869565,
 0.26551724, 0.35654008, 0.88679245, 0.20189274, 0.21501706,
 0.6097561 , 0.45054945, 0.23466476],
[0.72368421, 0.39920949, 0.5026738 , 0.58762887, 0.2173913 ,
 0.12758621, 0.07172996, 0.52830189, 0.1955836 , 0.70819113,
 0.17886179, 0.15018315, 0.2403709 ],
[0.62105263, 0.20355731, 0.67379679, 0.28350515, 0.25      ,
 0.64482759, 0.54852321, 0.39622642, 0.32807571, 0.3003413 ,
 0.35772358, 0.71428571, 0.65406562],
[0.83947368, 0.18972332, 0.5026738 , 0.29381443, 0.52173913,
 0.76551724, 0.56118143, 0.24528302, 0.51104101, 0.43515358,
 0.37398374, 0.74725275, 0.4935806 ],
[0.26578947, 0.70355731, 0.54545455, 0.58762887, 0.10869565,
 0.3862069 , 0.29746835, 0.54716981, 0.29652997, 0.11262799,
 0.25203252, 0.47619048, 0.21540656],
[0.83684211, 0.65217391, 0.57754011, 0.42783505, 0.44565217,
 0.64482759, 0.48734177, 0.32075472, 0.26498423, 0.33788396,
 0.31707317, 0.75457875, 0.57203994],
[1.        , 0.17786561, 0.43315508, 0.17525773, 0.29347826,
```

```
        0.62758621, 0.55696203, 0.30188679, 0.49526814, 0.33447099,
        0.48780488, 0.57875458, 0.54707561],
       [0.53157895, 0.20355731, 0.39572193, 0.32989691, 0.40217391,
        0.69655172, 0.56118143, 0.28301887, 0.51104101, 0.32081911,
        0.32520325, 0.76190476, 0.43295292]]), 54     0.0
151    2.0
63     1.0
55     0.0
123    1.0
121    1.0
7      0.0
160    2.0
106    1.0
90     1.0
141    2.0
146    2.0
5      0.0
98     1.0
168    2.0
80     1.0
33     0.0
18     0.0
61     1.0
51     0.0
66     1.0
37     0.0
4      0.0
104    1.0
60     1.0
111    1.0
126    1.0
86     1.0
112    1.0
164    2.0
26     0.0
56     0.0
129    1.0
45     0.0
8      0.0
44     0.0
Name: target, dtype: float64)
```

Дерево решений

```
dt = DecisionTreeRegressor(random_state=0)
dt_prediction = dt.fit(x_train, y_train).predict(x_test)
```

Градиентный бустинг

```
gb = GradientBoostingClassifier(random_state=0)
gb_prediction = gb.fit(x_train, y_train).predict(x_test)
```

Оценка качества решений

```
print("Decision tree: ", accuracy_score(y_test, dt_prediction))
print("Gradient boosting: ", accuracy_score(y_test, gb_prediction))
```

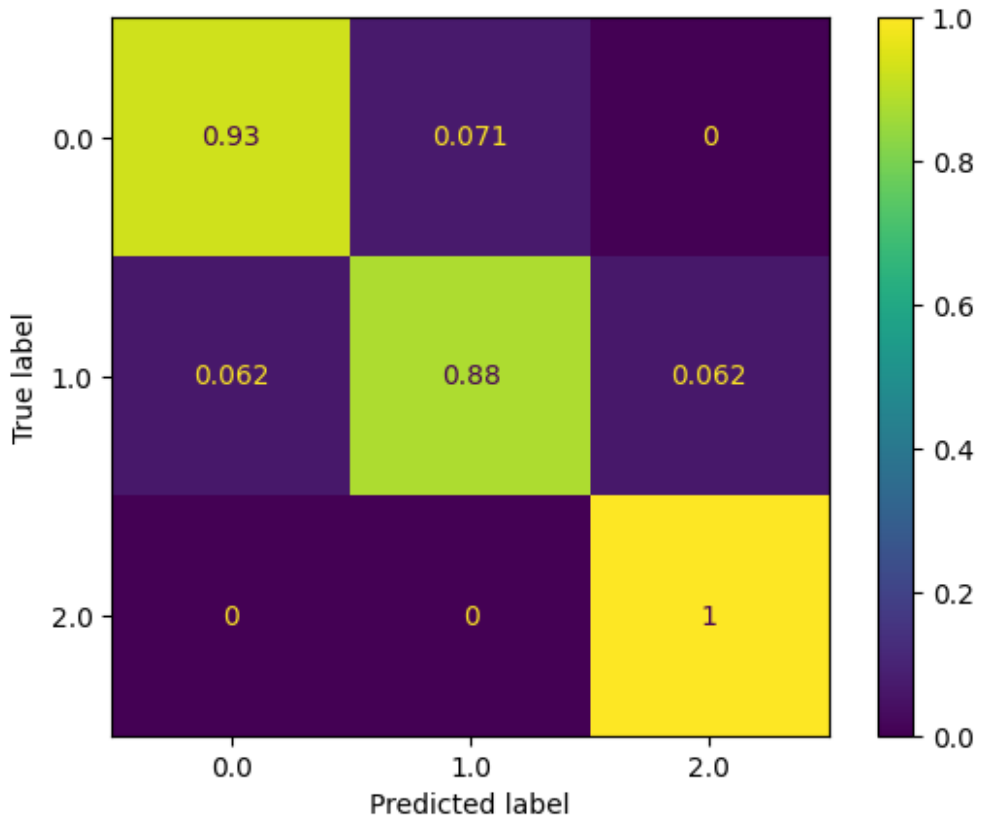
```
Decision tree:  0.9166666666666666
Gradient boosting:  0.9444444444444444
```

```
print("Decision tree: ", accuracy_score(y_test, dt_prediction))

cm = confusion_matrix(y_test, dt_prediction, labels=np.unique(df.target),
normalize='true')
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=np.unique(df.target))
disp.plot()
```

```
Decision tree:  0.9166666666666666

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7fc8d3749af0>
```



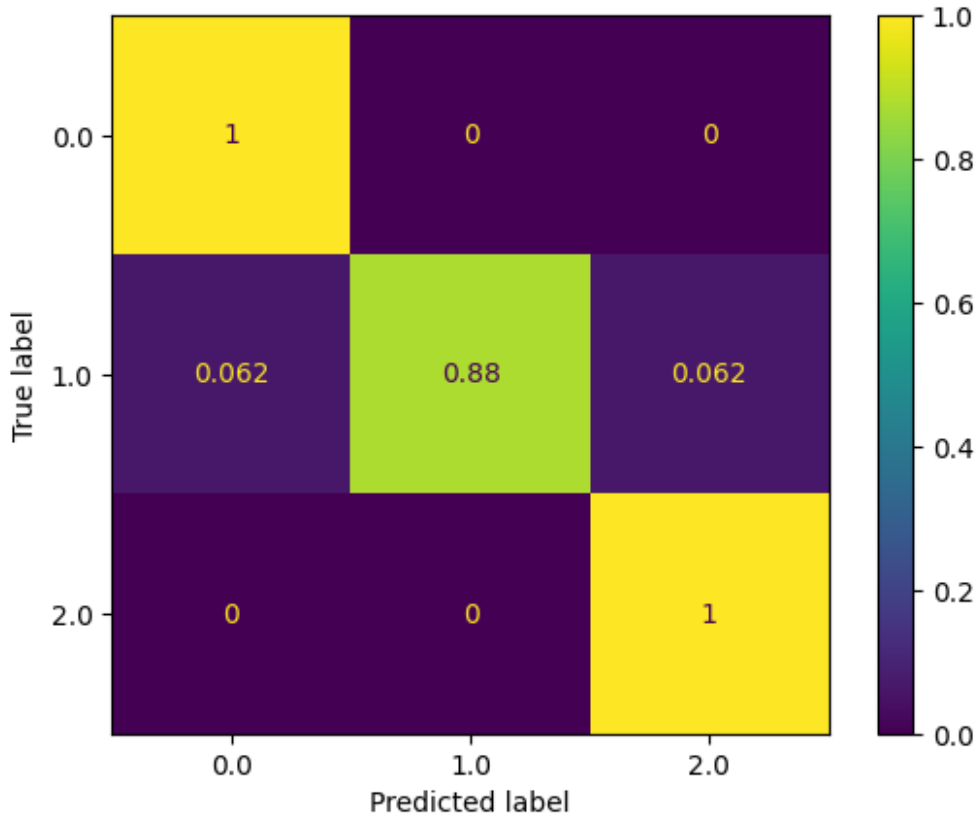

```
print("Gradient boosting: ", accuracy_score(y_test, gb_prediction))

cm = confusion_matrix(y_test, gb_prediction, labels=np.unique(df.target),
normalize='true')
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
```

```
display_labels=np.unique(df.target))
disp.plot()
```

```
Gradient boosting:  0.9444444444444444
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7fc8d8040cd0>
```



Для оценки качества решений были использованы метрики, подходящие для задач классификации: accuracy и confusion matrix.

По итогам исследования можно сделать вывод, что обе модели имеют достаточно высокую, однако не идеальную точность: 0.92 для дерева решений и 0.94 для градиентного бустинга.