Вариант №1, 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_iris.html#sklearn.datasets.load_iris

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 DecisionTreeClassifier
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

iris = load_iris()

df = make_dataframe(load_iris)

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

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
\
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target
0     0.0
```

```
1     0.0
2     0.0
3     0.0
4     0.0
```

```
df.dtypes
```

```
sepal length (cm)    float64
sepal width (cm)     float64
petal length (cm)    float64
petal width (cm)     float64
target               float64
dtype: object
```

Категориальные признаки отсутствуют

```
# 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))
```

```
sepal length (cm) - 0
sepal width (cm) - 0
petal length (cm) - 0
petal width (cm) - 0
target - 0
```

Пустых значений нет

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)
```

Градиентный бустинг

```
gb1 = GradientBoostingClassifier(random_state=0)
gb1_prediction = gb1.fit(x_train, y_train).predict(x_test)
```

Дерево решений

```
dt1 = DecisionTreeClassifier(random_state=0)
dt1_prediction = dt1.fit(x_train, y_train).predict(x_test)
```

Оценка качества решений

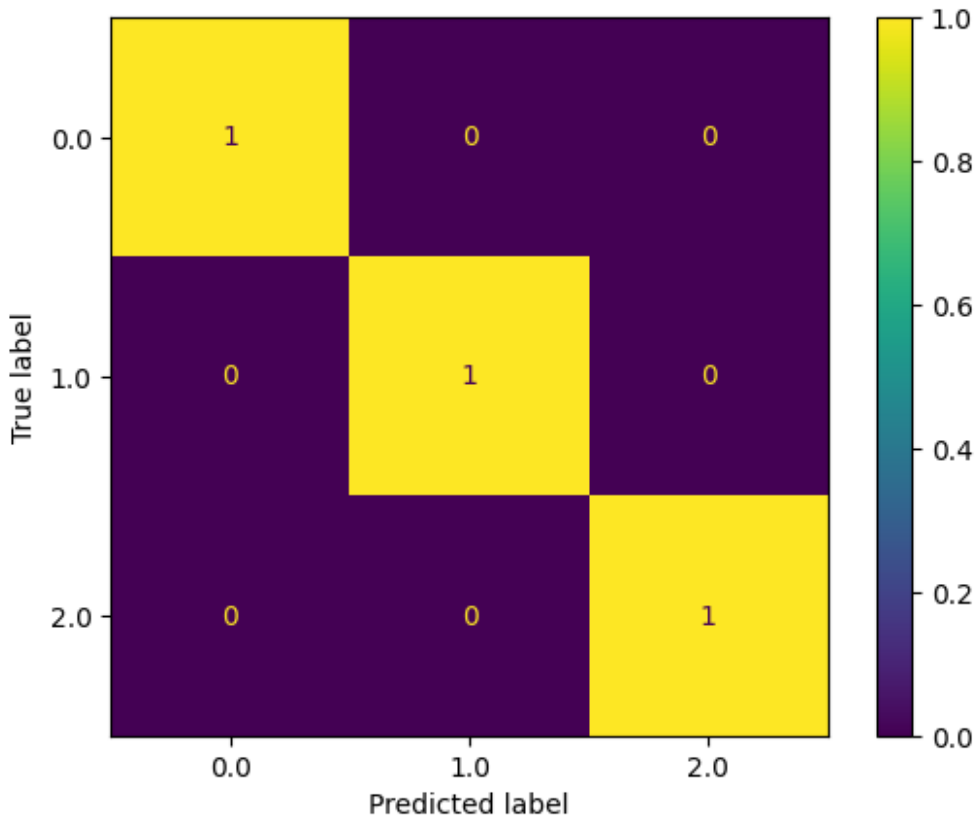
```
print("Decision tree: ", accuracy_score(y_test, dt1_prediction))

cm = confusion_matrix(y_test, dt1_prediction, labels=np.unique(df.target),
```

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

```
Decision tree:  1.0
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x23aa1aa8610>
```

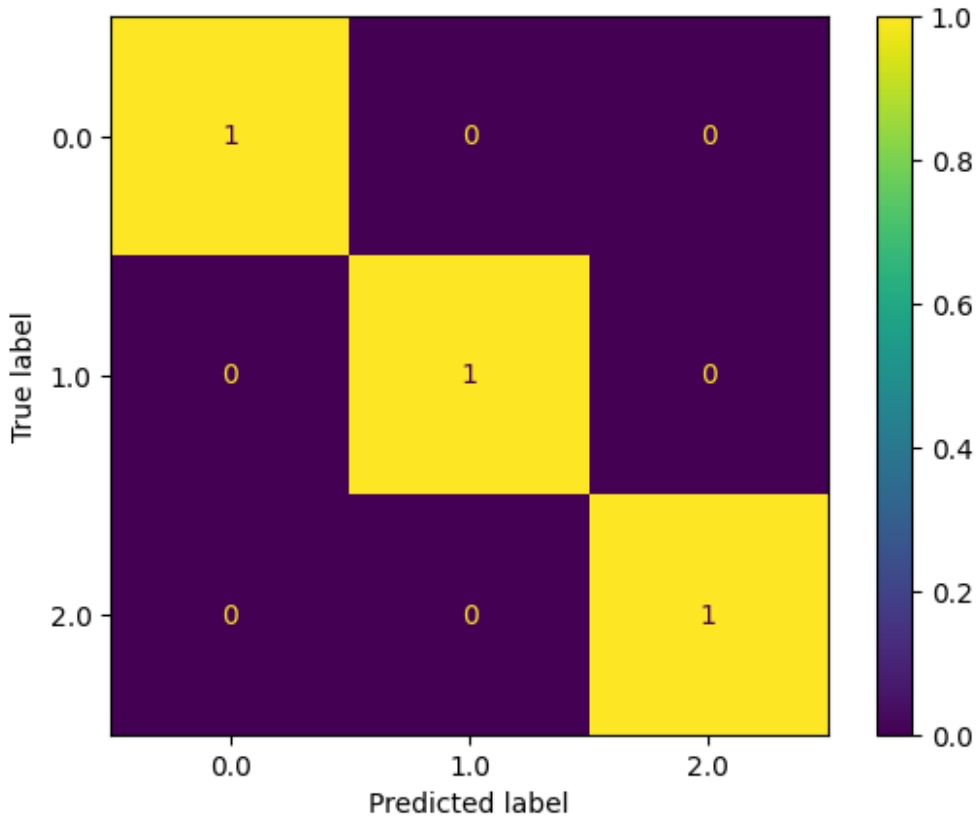



```
print("Gradient boosting: ", accuracy_score(y_test, gb1_prediction))

cm = confusion_matrix(y_test, gb1_prediction, labels=np.unique(df.target),
normalize='true')
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=np.unique(df.target))
disp.plot()
```

```
Gradient boosting:  1.0
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x23ad64a86d0>
```




```
print("Decision tree: ", accuracy_score(y_test, dt1_prediction))
print("Gradient boosting: ", accuracy_score(y_test, gb1_prediction))
```

```
Decision tree:  1.0
Gradient boosting:  1.0
```

Для оценки качества решений были использованы метрики: accuracy и confusion matrix.

Таким образом, обе модели имеют идеальную точность: 1