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

Категариальные 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

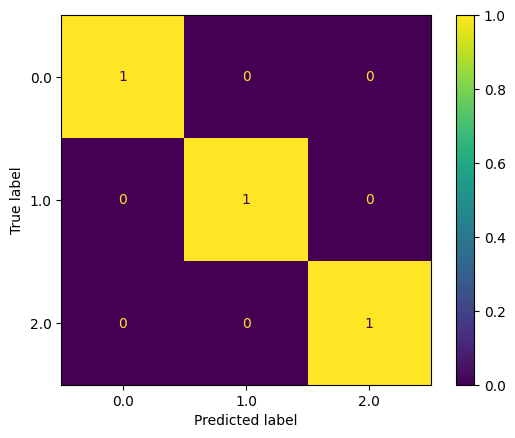
dt1 = DecisionTreeClassifier(random\_state=0)  
dt1\_prediction = dt1.fit(x\_train, y\_train).predict(x\_test)

### 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

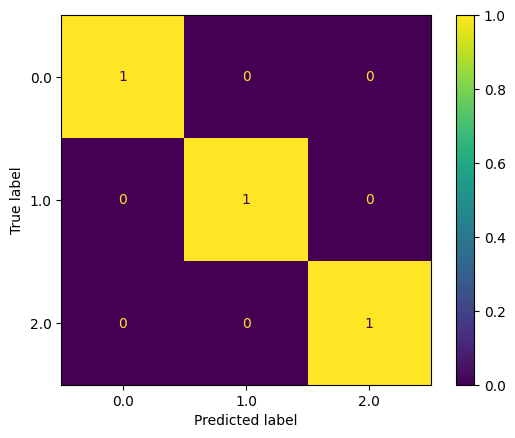
<sklearn.metrics.\_plot.confusion\_matrix.ConfusionMatrixDisplay at 0x23aa1aa8610>



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