Московский государственный технический университет им. Н.Э. Баумана

Кафедра «Системы обработки информации и управления» – ИУ5

Факультет «Радиотехнический» – РТ5

**Отчёт по рубежному контролю №2 по курсу**

**Технологии машинного обучения**

4

(количество листов)

|  |  |  |
| --- | --- | --- |
| Исполнитель |  |  |
| студент группы РТ5-61б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Нижаметдинов М. Ш. |
|  |  | “\_\_\_” \_\_\_\_\_\_\_\_\_\_ 2023 г. |
|  |  |  |
| Проверил |  |  |
| Преподаватель кафедры ИУ5 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Гапанюк Ю. Е. |
|  |  | “\_\_\_” \_\_\_\_\_\_\_\_\_\_ 2023 г. |

Москва, 2023 г.

**Задание**

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>

**Исходный текст проекта**

### 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

wine = load\_wine()

df = make\_dataframe(load\_wine)

# Первые 5 строк датасета

df.head()

df.dtypes

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))

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}")

### Дерево решений

dt = DecisionTreeClassifier(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))

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()

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()

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, но не идеальную точность: ~0.92 для дерева решений и ~0.94 для градиентного бустинга.