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.