**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: Проверил:

студент группы РТ5-61Б преподаватель каф. ИУ5

Коваленко В.И. Гапанюк Ю.Е.

Подпись и дата: Подпись и дата:

Москва, 2023 г.

Вариант 5

Для заданного набора данных (по вар. 5) постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения моделей используйте методы 1 и 2 (Дерево решений, Градиентный бустинг). Оцените качество моделей на основе подходящих метрик качества (не менее двух метрик). Какие метрики качества Вы использовали и почему? Какие выводы Вы можете сделать о качестве построенных моделей? Для построения моделей необходимо выполнить требуемую предобработку данных: заполнение пропусков, кодирование категориальных признаков, и т.д.

Описание датасета

Набор данных содержит несколько параметров, которые считаются важными при подаче заявки на магистерские программы. Параметры:

1. GRE Scores ( out of 340 )

2. TOEFL Scores ( out of 120 )

3. University Rating ( out of 5 )

4. Statement of Purpose and Letter of Recommendation Strength ( out of 5 )

5. Undergraduate GPA ( out of 10 )

6. Research Experience ( either 0 or 1 )

7. Chance of Admit ( ranging from 0 to 1 )

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.ensemble import GradientBoostingClassifier from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import train\_test\_split from sklearn import preprocessing

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import MinMaxScaler

from typing import Dict, Tuple

from sklearn.metrics import multilabel\_confusion\_matrix, ConfusionMatrixDisplay

from sklearn.metrics import precision\_score, recall\_score, f1\_score, classification\_report

filename = '/Users/kovalenkov/Documents/Бауманка/3 курс/6 семестр/ТМО/Machine\_learning\_technologies/Admission\_Predict.csv' ds = pd.read\_csv(filename)Scale = MinMaxScaler()

features = ['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA',

'Research', 'Chance of Admit ']

ds[features] = Scale.fit\_transform(ds[features])

pd.set\_option('display.max\_colwidth', None)

pd.set\_option('display.float\_format', '{:.2f}'.format)

ds = pd.DataFrame(ds[features])

ds.head()

GRE Score

Research \

0 0.94

1.00

1 0.68

1.00

2 0.52

1.00

3 0.64

1.00

4 0.48 0.00

TOEFL Score

0.93

0.54

0.43

0.64

0.39

University Rating SOP LOR CGPA

0.75 0.88 0.88 0.91

0.75 0.75 0.88 0.66

0.50 0.50 0.62 0.38

0.50 0.62 0.38 0.60

0.25 0.25 0.50 0.45

Chance of Admit

0 0.92

1 0.67

2 0.60

3 0.73

4 0.49

# размер датасета

ds.shape

(400, 8)

# проверим наличие пустых значений

for col in ds.columns:

temp\_null\_count = ds[ds[col].isnull()].shape[0] print('{} - {}'.format(col, temp\_null\_count))

GRE Score - 0

TOEFL Score - 0

University Rating - 0

SOP - 0

LOR - 0

CGPA - 0

Research - 0

Chance of Admit - 0

# Основные статистические характеристки набора данных

ds.describe()

GRE Score

count 400.00 400.00 mean 0.54 0.55 std 0.23 0.22 min 0.00 0.00 25% 0.36 0.39 50% 0.54 0.54 75% 0.70 0.71 max 1.00 1.00

University Rating SOP LOR CGPA 400.00 400.00 400.00 400.00 0.52 0.60 0.61 0.58 0.29 0.25 0.22 0.19 0.00 0.00 0.00 0.00 0.25 0.38 0.50 0.44 0.50 0.62 0.62 0.58 0.75 0.75 0.75 0.73 1.00 1.00 1.00 1.00

\

TOEFL Score

Research Chance of Admit count 400.00 400.00 mean 0.55 0.61 std 0.50 0.23 min 0.00 0.00 25% 0.00 0.48 50% 1.00 0.62 75% 1.00 0.78 max 1.00 1.00

# уникальные значения для целевого признака

ds['Chance of Admit '].unique()

array([0.92063492, 0.66666667, 0.6031746 , 0.73015873, 0.49206349, 0.88888889, 0.65079365, 0.53968254, 0.25396825, 0.17460317, 0.28571429, 0.79365079, 0.6984127 , 0.44444444, 0.42857143, 0.31746032, 0.50793651, 0.46031746, 0.47619048, 0.57142857, 0.95238095, 0.96825397, 1. , 0.15873016, 0.19047619, 0.63492063, 0.9047619 , 0.85714286, 0.38095238, 0.22222222, 0.23809524, 0.3015873 , 0.84126984, 0.82539683, 0.87301587, 0.76190476, 0.34920635, 0.03174603, 0.12698413, 0.20634921, 0.33333333, 0.36507937, 0.98412698, 0.93650794, 0.06349206, 0. , 0.71428571, 0.58730159, 0.55555556, 0.3968254 , 0.80952381, 0.68253968, 0.74603175, 0.77777778, 0.52380952, 0.61904762, 0.41269841, 0.14285714, 0.26984127, 0.07936508])

Разделим на тестовую и обучающую выборку

y = ds['Chance of Admit ']

x = ds.drop('Chance of Admit ', axis = 1)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.5)

print(f"Обучающая выборка:\n{X\_train, y\_train}")

print(f"Тестовая выборка:\n{X\_test, y\_test}")

Обучающая выборка:

( GRE

Research

199

0.00

296

0.00

7

0.00

46

1.00

251

0.00

..

...

349

0.00

140

1.00

20

1.00

206

0.00

287

1.00

Score TOEFL

0.46

0.40

0.36

0.78

0.52

...

0.46

0.78

0.44

0.50

0.68

Score University Rating SOP LOR CGPA

[200 rows x 7 columns], 199

296 0.67

7 0.54

46 0.83

251 0.57 ...

0.60

0.54

0.54

0.32

0.79

0.25

...

0.32

0.64

0.54

0.25

0.79

0.50 0.75 0.88 0.61

0.50 0.62 0.62 0.60

0.25 0.50 0.75 0.35

1.00 0.75 1.00 0.80

0.25 0.38 0.50 0.71

... ... ... ...

0.50 0.38 0.50 0.40

0.25 0.75 0.50 0.75

0.50 0.50 0.25 0.35

0.25 0.62 0.50 0.35

1.00 1.00 0.88 0.73

349 0.44

140 0.79

20 0.48

206 0.46

287 0.87

Name: Chance of Admit , Length: 200, dtype: float64) Тестовая выборка:

( GRE

Research

0

1.00

108

1.00

184

0.00

313

0.00

276

1.00

..

...

85

0.00

266

0.00

293

1.00

186

1.00

27

1.00

Score TOEFL

0.94

0.82

0.52

0.22

0.78

...

0.58

0.44

0.44

0.54

0.16

Score University Rating SOP LOR CGPA

[200 rows x 7 columns], 0

108 0.94

184 0.60

313 0.52

276 0.87 ...

0.92

0.93

0.86

0.50

0.29

0.75

...

0.39

0.46

0.21

0.54

0.21

0.75 0.88 0.88 0.91

1.00 1.00 1.00 0.83

0.25 0.38 0.75 0.49

0.50 0.62 0.50 0.40

1.00 1.00 0.88 0.85

... ... ... ...

0.75 0.88 0.62 0.60

0.25 0.25 0.38 0.53

0.00 0.62 0.50 0.44

0.50 0.62 0.50 0.60

0.25 0.12 0.38 0.22

85 0.67

266 0.60

293 0.48

186 0.79

27 0.16

Name: Chance of Admit , Length: 200, dtype: float64)

Дерево решений

desTree = DecisionTreeClassifier(random\_state=0)

lab = preprocessing.LabelEncoder()

y\_transformed = lab.fit\_transform(y\_train)

desTree\_prediction = desTree.fit(X\_train,

y\_transformed).predict(X\_test)

Градиентный бустинг

gradBoost = GradientBoostingClassifier(random\_state=0)

gradBoost\_prediction = gradBoost.fit(X\_train,

y\_transformed).predict(X\_test)

Оценка качества моделей с помощью accuracy\_score для классов

def accuracy\_score\_for\_classes(

y\_true: np.ndarray,

y\_pred: np.ndarray) -> Dict[int, float]: d = {'t': y\_true, 'p': y\_pred}

df = pd.DataFrame(data=d)

# Метки классов

classes = np.unique(y\_true)

# Результирующий словарь

res = dict()

# Перебор меток классов

for c in classes:

# отфильтруем данные, которые соответствуют # текущей метке класса в истинных значениях temp\_data\_flt = df[df['t']==c]

# расчет accuracy для заданной метки класса temp\_acc = accuracy\_score(

temp\_data\_flt['t'].values,

temp\_data\_flt['p'].values)

# сохранение результата в словарь res[c] = temp\_acc

return res

def print\_accuracy\_score\_for\_classes(

y\_true: np.ndarray,

y\_pred: np.ndarray):

accs = accuracy\_score\_for\_classes(y\_true, y\_pred) if len(accs)>0:

print('Метка \t Accuracy') for i in accs:

print('{} \t {}'.format(i, accs[i]))

yTest\_transformed = lab.fit\_transform(y\_test)

print("Decision tree:")

print\_accuracy\_score\_for\_classes(yTest\_transformed,

desTree\_prediction)

print("Gradient boosting:")

print\_accuracy\_score\_for\_classes(yTest\_transformed,

gradBoost\_prediction)

Decision tree: Метка Accuracy 0 0.0

1 0.0

2 0.0

3 0.0

4 0.0

5 0.0

6 0.0

7 0.0

8 0.0

9 0.0

10 0.0

11 0.0

12 0.0

13 0.2

14 0.0

15 0.0

16 0.0

17 0.0

18 0.0

19 0.0

20 0.5

21 0.0

22 0.0

23 0.2222222222222222 24 0.0

25 0.25

26 0.0

27 0.0

28 0.0

29 0.14285714285714285 30 0.0

31 0.0

32 0.0

33 0.16666666666666666 34 0.0

35 0.0

36 0.0

37 0.0

38 0.0

39 0.0

40 0.0

41 0.0

42 0.0

43 0.0

44 0.0

45 0.0

46 0.0

47 0.0

48 0.0

49 0.0

50 0.0

51 0.0

52 0.0

53 0.0

54 0.0

55 0.0

56 0.0

Gradient boosting:

Метка Accuracy 0 0.0

1 0.0

2 0.0

3 0.0

4 0.0

5 0.0

6 0.0

7 0.0

8 0.0

9 0.0

10 0.0

11 0.0

12 0.0

13 0.2

14 0.0

15 0.0

16 0.0

17 0.0

18 0.5

19 0.0

20 0.0

21 0.0

22 0.0

23 0.0

24 0.0

25 0.0

26 0.16666666666666666 27 0.0

28 0.0

29 0.0

30 0.0

31 0.0

32 0.0

33 0.16666666666666666 34 0.0

35 0.1111111111111111 36 0.0

37 0.0

38 0.0

39 0.0 40 0.0 41 0.0 42 0.0 43 0.0 44 0.0 45 0.0 46 0.0 47 0.0 48 0.0

49 0.0 50 0.0 51 0.0 52 0.0 53 0.0 54 0.0 55 0.0 56 0.0

def convert\_target\_to\_binary(array:np.ndarray, target:int) -> np.ndarray:

# Если целевой признак совпадает с указанным, то 1 иначе 0

res = [1 if x==target else 0 for x in array] return res

bin\_target1\_1 = convert\_target\_to\_binary(desTree\_prediction, 1) bin\_y\_train = convert\_target\_to\_binary(y\_train, 1) print("Decision tree:")

precision\_score(bin\_y\_train, bin\_target1\_1, average='weighted')

Decision tree:

0.9701507537688443

print("Gradient boosting:")

bin\_target1\_2 = convert\_target\_to\_binary(gradBoost\_prediction, 2) precision\_score(bin\_y\_train, bin\_target1\_2, average='weighted')

Gradient boosting:

0.9698461538461539

Для оценки качества решений были использованы метрики, подходящие для задач классификации: accuracy для классов и precision\_score.

По итогам исследования можно сделать вывод, что обе модели имеют достаточно высокую точность: 0.9701 для дерева решений и 0.969 для градиентного бустинга.