### þÿМинистерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский государственный технический университет имени Н.Э. Баумана**

### (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ (ИУ5)

О Т Ч Е Т

**по лабораторной работе**

по дисциплине: Технологии машинного обучения

на тему: Ансамбли моделей машинного обучения.

\_

\_

\_

\_

\_

\_

\_

\_

Студент РТ5-61Б **А.C. Акушко**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель

**Ю.Е. Гапанюк**

(Подпись, дата) (И.О.Фамилия)

*2020 г.*

# Лабораторная работа №6

**Ансамбли моделей машинного обучения**

## Цель лабораторной работы

Изучение ансамблей моделей машинного обучения.

## Задание

Выберите набор данных (датасет) для решения задачи классификации или регресии.

В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.

С использованием метода train\_test\_split разделите выборку на обучающую и тестовую. Обучите две ансамблевые модели. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.

# Ход выполнения лабораторной работы

In [1]:

**import pandas as pd import seaborn as sns import numpy as np**

**import matplotlib.pyplot as plt**

**from sklearn.preprocessing import** MinMaxScaler

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

**from sklearn.metrics import** f1\_score, precision\_score, recall\_score, accuracy\_score

**from sklearn.ensemble import** RandomForestClassifier

**from sklearn.ensemble import** GradientBoostingClassifier

%**matplotlib** inline sns.set(style="ticks")

In [2]:

col\_list = ['Pelvic\_incidence', 'Pelvic\_tilt', 'Lumbar\_lordosis\_angle', 'Sacral\_slope', 'Pelvic\_radius', 'Degree\_spondylolisthesis', 'Pelvic\_slope',

'Direct\_tilt', 'Thoracic\_slope', 'Cervical\_tilt', 'Sacrum\_angle', 'Scoliosis\_slope', 'Class\_att', 'To\_drop']

data = pd.read\_csv('data/Dataset\_spine.csv', names=col\_list, header=1, sep=",") data.drop('To\_drop', axis=1, inplace=**True**)

In [3]:

data.head()

Out[3]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Pelvic\_incidence** | **Pelvic\_tilt** | **Lumbar\_lordosis\_angle** | **Sacral\_slope** | **Pelvic\_radius** | **Degree\_spondy** |
| **0** 39.056951 | 10.060991 | 25.015378 | 28.995960 | 114.405425 |  |
| **1** 68.832021 | 22.218482 | 50.092194 | 46.613539 | 105.985135 |  |
| **2** 69.297008 | 24.652878 | 44.311238 | 44.644130 | 101.868495 |  |
| **3** 49.712859 | 9.652075 | 28.317406 | 40.060784 | 108.168725 |  |
| **4** 40.250200 | 13.921907 | 25.124950 | 26.328293 | 130.327871 |  |

In [4]:

data.isnull().sum()

Out[4]: Pelvic\_incidence 0

Pelvic\_tilt 0

Lumbar\_lordosis\_angle 0

Sacral\_slope 0

Pelvic\_radius 0

Degree\_spondylolisthesis 0

Pelvic\_slope 0

Direct\_tilt 0

Thoracic\_slope 0

Cervical\_tilt 0

Sacrum\_angle 0

Scoliosis\_slope 0

Class\_att 0

dtype: int64

Пропуски данных отсутствуют.

In [5]:

*#Кодирование категориальных признаков*

data['Class\_att'] = data['Class\_att'].map({'Abnormal': 1, 'Normal': 0})

**В качестве метрики для решения задачи классификации будем использовать: Precision - доля верно предсказанных классификатором положительных объектов, из всех объектов, которые классификатор верно или неверно определил как положительные.**

Разработаем класс, который позволит сохранять метрики качества построенных моделей и реализует визуализацию метрик качества.

In [6]:

**class MetricLogger**:

**def** init (self):

self.df = pd.DataFrame(

{'metric': pd.Series([], dtype='str'),

'alg': pd.Series([], dtype='str'),

'value': pd.Series([], dtype='float')})

**def** add(self, metric, alg, value):

*"""*

*Добавление значения*

*"""*

*# Удаление значения если оно уже было ранее добавлено* self.df.drop(self.df[(self.df['metric']==metric)&(self.df['alg']==alg)].index, inplace = **True**) *# Добавление нового значения*

temp = [{'metric':metric, 'alg':alg, 'value':value}] self.df = self.df.append(temp, ignore\_index=**True**)

**def** get\_data\_for\_metric(self, metric, ascending=**True**):

*"""*

*Формирование данных с фильтром по метрике*

*"""*

temp\_data = self.df[self.df['metric']==metric]

temp\_data\_2 = temp\_data.sort\_values(by='value', ascending=ascending)

**return** temp\_data\_2['alg'].values, temp\_data\_2['value'].values

**def** plot(self, str\_header, metric, ascending=**True**, figsize=(5, 5)):

*"""*

*Вывод графика*

*"""*

array\_labels, array\_metric = self.get\_data\_for\_metric(metric, ascending) fig, ax1 = plt.subplots(figsize=figsize)

pos = np.arange(len(array\_metric)) rects = ax1.barh(pos, array\_metric, align='center',

height=0.5, tick\_label=array\_labels)

ax1.set\_title(str\_header)

**for** a,b **in** zip(pos, array\_metric):

plt.text(0.5, a-0.05, str(round(b,3)), color='white') plt.show()

Для задачи классификации будем использовать случайный лес и градиентный бустинг.

## Формирование обучающей и тестовой выборок

In [7]:

data.columns

Out[7]: Index(['Pelvic\_incidence', 'Pelvic\_tilt', 'Lumbar\_lordosis\_angle', 'Sacral\_slope', 'Pelvic\_radius', 'Degree\_spondylolisthesis', 'Pelvic\_slope', 'Direct\_tilt', 'Thoracic\_slope', 'Cervical\_tilt', 'Sacrum\_angle', 'Scoliosis\_slope', 'Class\_att'], dtype='object')

In [8]:

data.dtypes

Out[8]: Pelvic\_incidence float64 Pelvic\_tilt float64 Lumbar\_lordosis\_angle float64 Sacral\_slope float64

Pelvic\_radius float64 Degree\_spondylolisthesis float64 Pelvic\_slope float64

Direct\_tilt float64 Thoracic\_slope float64

Cervical\_tilt float64 Sacrum\_angle float64

Scoliosis\_slope float64

Class\_att int64 dtype: object

In [9]:

*# Признаки для задачи классификации*

class\_cols = ['Pelvic\_incidence', 'Pelvic\_tilt', 'Lumbar\_lordosis\_angle', 'Degree\_spondylolisthesis',

]

In [10]:

X = data[class\_cols] Y = data['Class\_att'] X.shape

Out[10]: (309, 4)

In [11]:

*# С использованием метода train\_test\_split разделим выборку на обучающую и тестовую*

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.25, random\_state=1)

In [12]:

X\_train.shape, X\_test.shape, Y\_train.shape, Y\_test.shape

Out[12]: ((231, 4), (78, 4), (231,), (78,))

## Обучение моделей

In [13]:

*# Сохранение метрик*

clasMetricLogger = MetricLogger()

In [14]:

**def** train\_model(model\_name, model, MetricLogger): model.fit(X\_train, Y\_train)

Y\_pred = model.predict(X\_test)

precision = precision\_score(Y\_test.values, Y\_pred) MetricLogger.add('precision', model\_name, precision)

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*') print(model\_name)

print(model) print("precision\_score:", precision)

In [15]:

train\_model('Случайный лес', RandomForestClassifier(), clasMetricLogger) train\_model('Градиентный бустинг', GradientBoostingClassifier(), clasMetricLogger)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Случайный лес

RandomForestClassifier(bootstrap=True, ccp\_alpha=0.0, class\_weight=None, criterion='gini', max\_depth=None, max\_features='auto', max\_leaf\_nodes=None, max\_samples=None, min\_impurity\_decrease=0.0, min\_impurity\_split=None, min\_samples\_leaf=1, min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0, n\_estimators=100, n\_jobs=None, oob\_score=False, random\_state=None, verbose=0, warm\_start=False)

precision\_score: 0.8363636363636363

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Градиентный бустинг

GradientBoostingClassifier(ccp\_alpha=0.0, criterion='friedman\_mse', init=None, learning\_rate=0.1, loss='deviance', max\_depth=3, max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None, min\_samples\_leaf=1, min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0, n\_estimators=100, n\_iter\_no\_change=None, presort='deprecated', random\_state=None, subsample=1.0, tol=0.0001, validation\_fraction=0.1, verbose=0,

warm\_start=False) precision\_score: 0.8333333333333334

## Оценка качества моделей

In [16]:

*# Метрики качества модели*

clas\_metrics = clasMetricLogger.df['metric'].unique() clas\_metrics

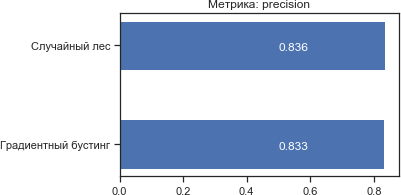
Out[16]: array(['precision'], dtype=object)

In [17]:

*# Построим графики метрик качества модели*

**for** metric **in** clas\_metrics:

clasMetricLogger.plot('Метрика: ' + metric, metric, figsize=(5, 3))



На основании метрики precision лучшим оказался случайный лес.