



Министерство 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 ИУ5-62Б
(Группа)

(Подпись, дата)

Карягин А.Д.
(И.О.Фамилия)

Руководитель

Ю.Е.
Гапанюк

(Подпись, дата)
(И.О.Фамилия)

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 регрессии.
- 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
        """
```



```
plt.show()
```

Для задачи классификации будем использовать случайный лес и градиентный бустинг.

Формирование обучающей и тестовой выборки

```
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_spondylolisthe',
              'sis',
              ]
```

```
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,
```

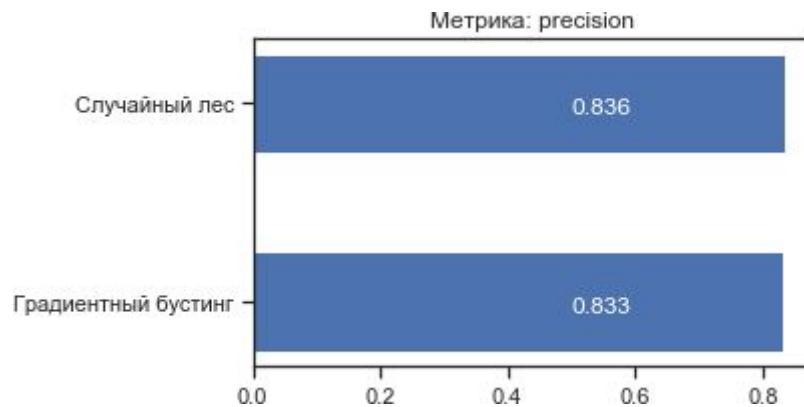
```
min_impurity_decrease=0.0, min_impurity_split=None,  
min_samples_leaf=1, min_samples_split=2,  
min_weight_fraction_leaf=0.0, n_estimators=100,  
n_iter_no_change=None, presort='deprecated',  
random_state=None, subsample=1.0, tol=0.0001,  
validation_fraction=0.1, verbose=0,  
warm_start=False)  
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.