|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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

**Отчёты по лабораторной работе №6**

По дисциплине:

«Технологии машинного обучения»

Выполнил:

Студент группыИУ5-62 **\_\_\_ \_ \_ \_\_\_\_ \_ Ким Р.И. \_\_**

(Подпись, дата) (Фамилия И.О.)

Проверил:

**\_\_\_\_\_\_\_\_\_ \_ Гапанюк Ю. Е.\_**

(Подпись, дата) (Фамилия И.О.)

Москва, 2020

## 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]:

data = pd.read\_csv('pulsar\_stars.csv')

In [3]:

data.head()

Out[3]:

**Mean of the integrated**

**profile**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **profile** | **profile** | **profile** |  | | | | |
| **0** | 140.562500 | 55.683782 | -0.234571 | -0.699648 | 3.199833 | 19.110426 | 7.975532 | 74.242225 | 0 |
| **1** | 102.507812 | 58.882430 | 0.465318 | -0.515088 | 1.677258 | 14.860146 | 10.576487 | 127.393580 | 0 |
| **2** | 103.015625 | 39.341649 | 0.323328 | 1.051164 | 3.121237 | 21.744669 | 7.735822 | 63.171909 | 0 |
| **3** | 136.750000 | 57.178449 | -0.068415 | -0.636238 | 3.642977 | 20.959280 | 6.896499 | 53.593661 | 0 |
| **4** | 88.726562 | 40.672225 | 0.600866 | 1.123492 | 1.178930 | 11.468720 | 14.269573 | 252.567306 | 0 |

**Standard deviation of the**

**integrated**

**Excess kurtosis of the**

**integrated**

**Skewness of**

**the integrated**

**Mean of the DM- SNR curve**

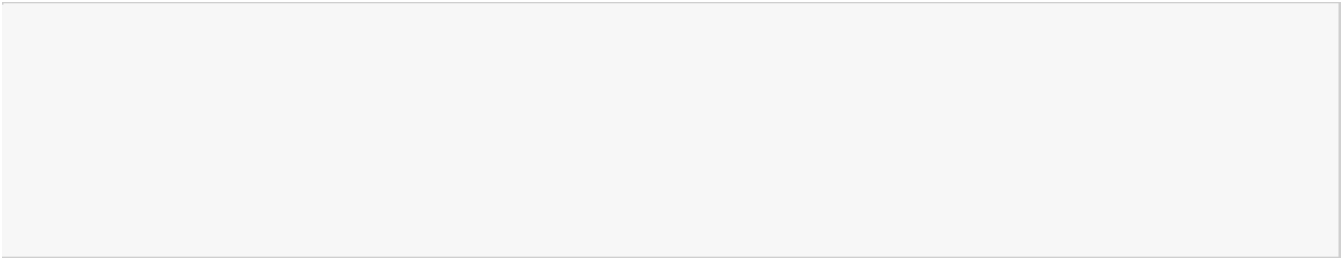
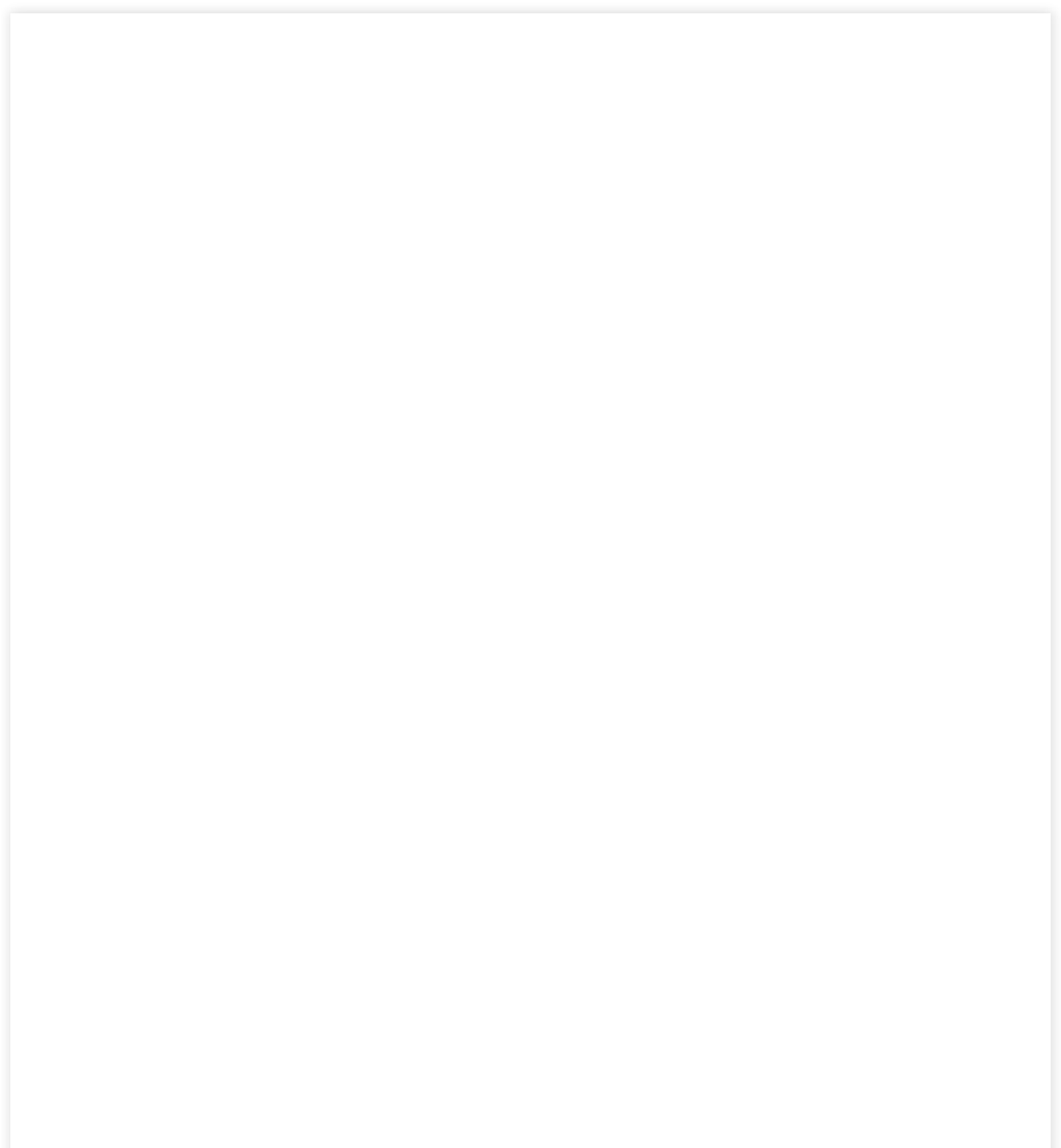
**Standard deviation of the DM-SNR curve**

**Excess kurtosis of the DM-SNR curve**

**Skewness of the DM-SNR**

**curve**

**target\_class**



In [4]:

data.isnull().sum()

Out[4]:

Standard deviation of the integrated profile 0

Excess kurtosis of the integrated profile 0

Skewness of the integrated profile 0

Mean of the DM-SNR curve 0

Standard deviation of the DM-SNR curve 0

Excess kurtosis of the DM-SNR curve 0

Skewness of the DM-SNR curve 0

target\_class 0

dtype: int64

Таким образом, пропуски данных отсутствуют.

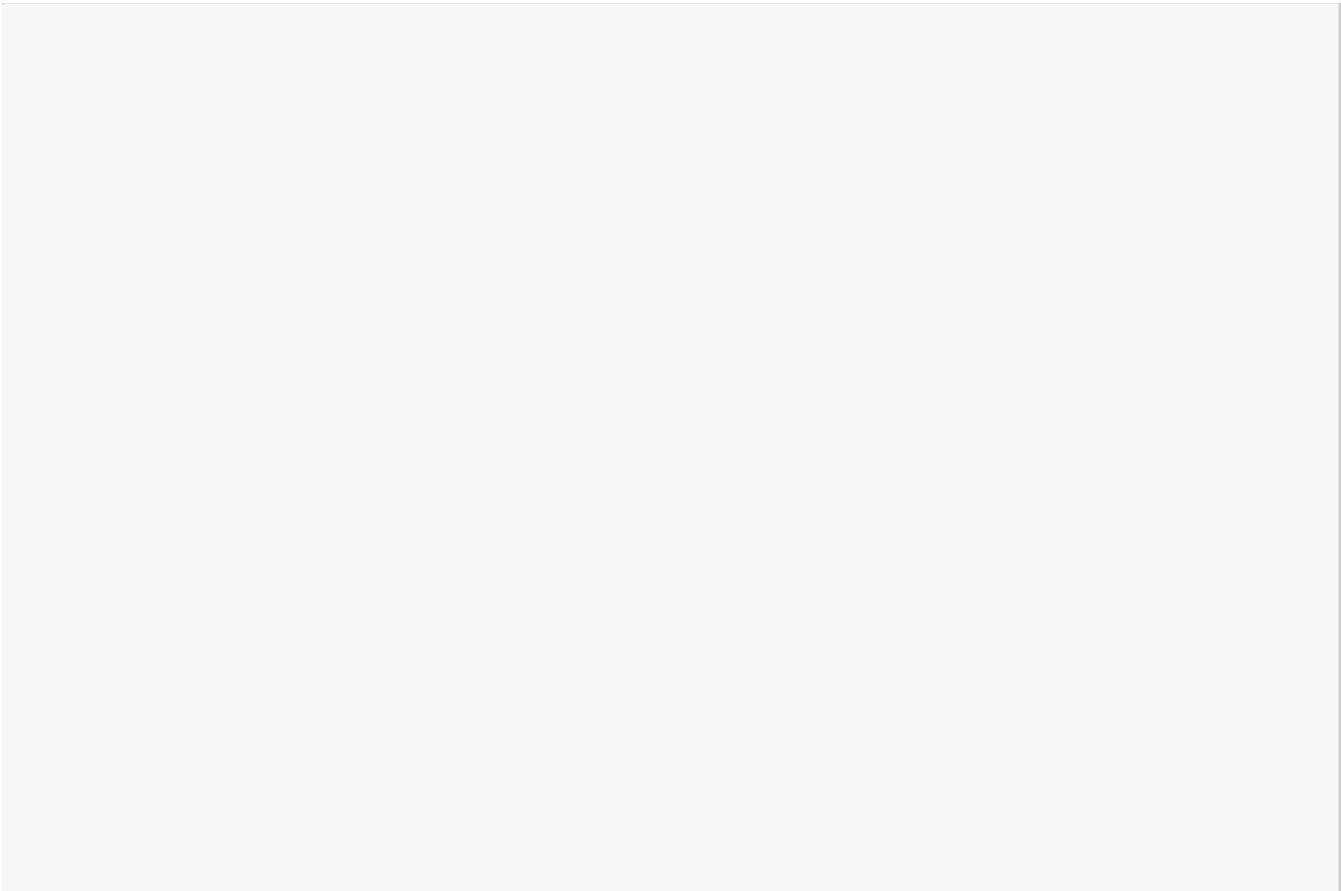
#### В качестве метрик для решения задачи классификации будем использовать:

Precision - доля верно предсказанных классификатором положительных объектов, из всех объектов, которые классификатор верно или неверно определил как положительные.

Recall - доля верно предсказанных классификатором положительных объектов, из всех действительно положительных объектов.

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

In [5]:



#### class MetricLogger:

**def** init (self): self.df = pd.DataFrame(

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

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

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

**ue**)

**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 = **Tr**

*# 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 [6]:



data.columns

Out[6]:

Index([' Mean of the integrated profile',

* Standard deviation of the integrated profile', ' Excess kurtosis of the integrated profile',
* Skewness of the integrated profile', ' Mean of the DM-SNR curve', ' Standard deviation of the DM-SNR curve',
* Excess kurtosis of the DM-SNR curve', ' Skewness of the DM-SNR curve', 'target\_class'],

dtype='object')

In [7]:



data.dtypes

Out[7]:

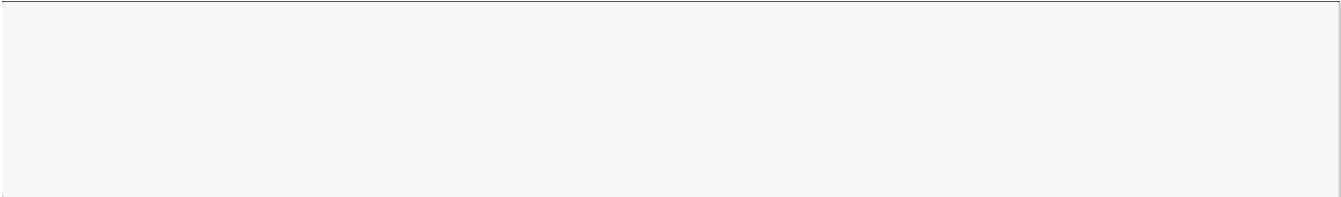
Mean of the integrated profile float64 Standard deviation of the integrated profile float64 Excess kurtosis of the integrated profile float64 Skewness of the integrated profile float64

Mean of the DM-SNR curve float64 Standard deviation of the DM-SNR curve float64 Excess kurtosis of the DM-SNR curve float64

Skewness of the DM-SNR curve float64 target\_class int64

dtype: object

In [8]:

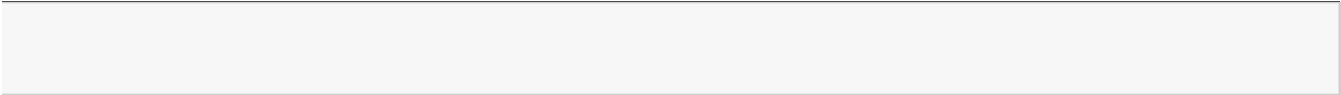


*# Признаки для задачи классификации*

class\_cols = [' Mean of the integrated profile',

* Standard deviation of the integrated profile', ' Excess kurtosis of the integrated profile',
* Skewness of the integrated profile', ' Mean of the DM-SNR curve',
* Standard deviation of the DM-SNR curve', ' Excess kurtosis of the DM-SNR curve']

In [9]:



X = data[class\_cols]

Y = data['target\_class'] X.shape

Out[9]:

(17898, 7)

In [10]:



*# 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 [11]:



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

Out[11]:

((13423, 7), (4475, 7), (13423,), (4475,))

### Обучение моделей

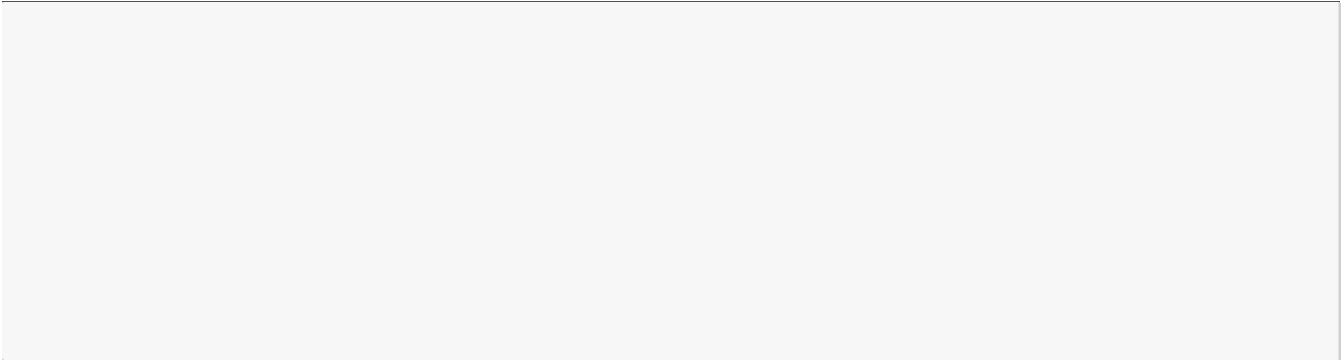
In [12]:



*# Сохранение метрик*

clasMetricLogger = MetricLogger()

In [13]:



**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) recall = recall\_score(Y\_test.values, Y\_pred)

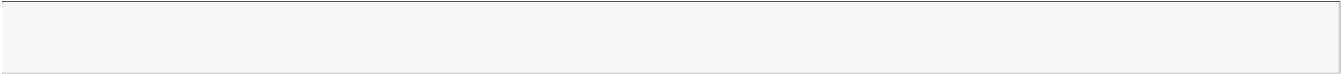
MetricLogger.add('precision', model\_name, precision) MetricLogger.add('recall', model\_name, recall)

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

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

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

In [14]:



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: 0.9272237196765498

recall: 0.8514851485148515

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

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 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)

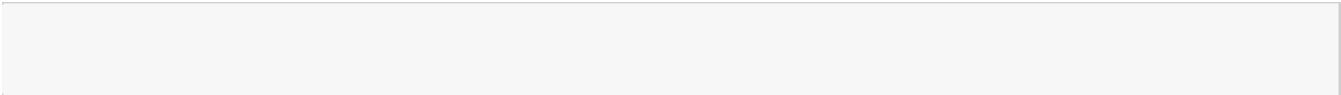
precision\_score: 0.9071618037135278

recall: 0.8465346534653465

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

### Оценка качества моделей

In [15]:

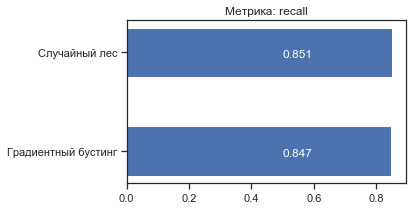
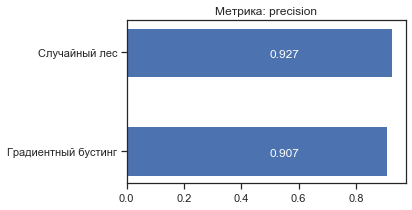
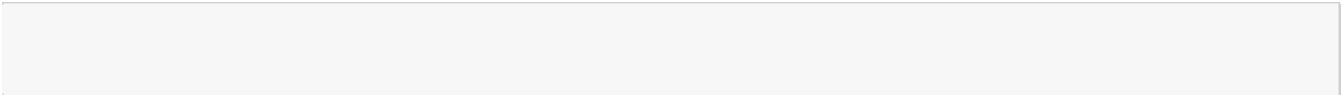
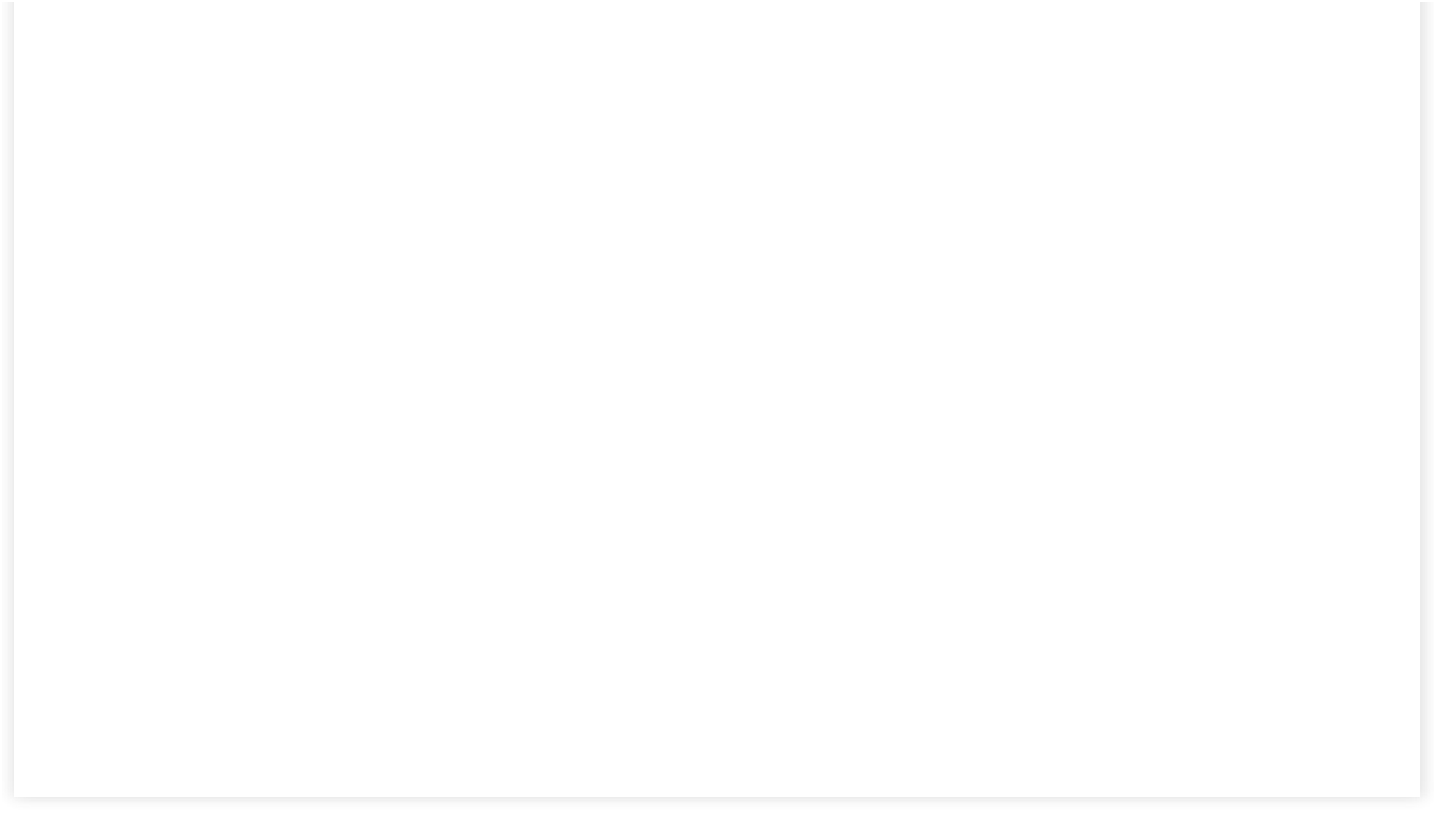


*# Метрики качества модели*

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

Out[15]:

array(['precision', 'recall'], dtype=object)



In [16]:

*# Построим графики метрик качества модели*

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

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

Таким образом, на основании обеих метрик лучшей моделью оказался случайный лес.