



**Министерство образования и науки 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

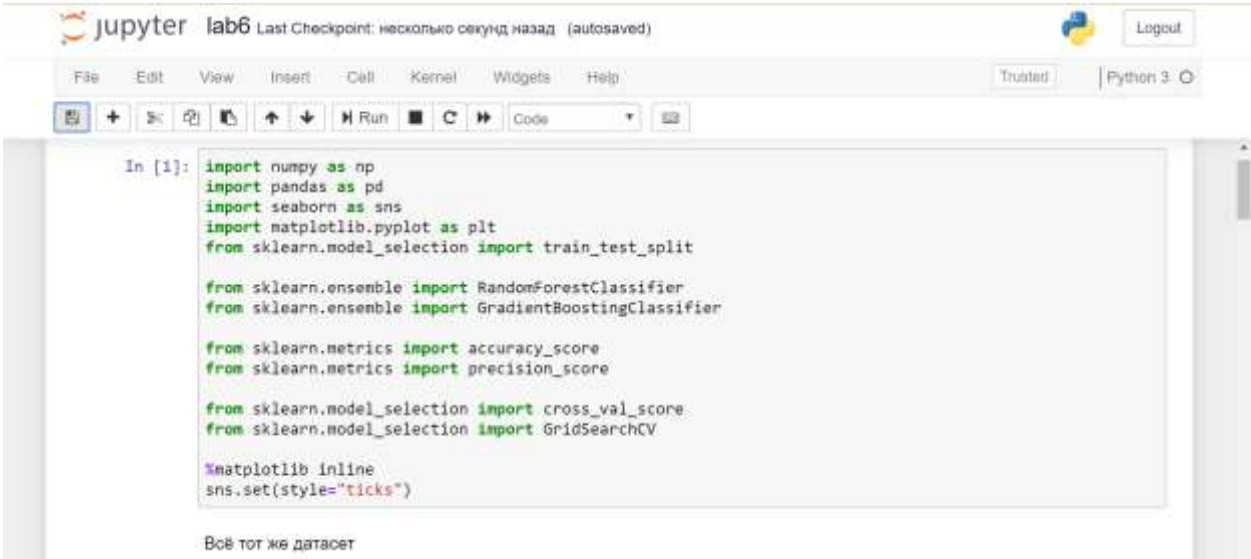
**Отчет по лабораторной работе № 6
«Ансамбли моделей машинного обучения»
по курсу “Технологии машинного обучения”**

Исполнитель:
Студент группы ИУ5-63
Желанкина А.С.
_____ 27.04.2019

Задание лабораторной работы

1. Выберите набор данных (датасет) для решения задачи классификации или регрессии.
2. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
3. С использованием метода `train_test_split` разделите выборку на обучающую и тестовую.
4. Обучите две ансамблевые модели. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.
5. Произведите для каждой модели подбор значений одного гиперпараметра. В зависимости от используемой библиотеки можно применять функцию `GridSearchCV`, использовать перебор параметров в цикле, или использовать другие методы.
6. Повторите пункт 4 для найденных оптимальных значений гиперпараметров. Сравните качество полученных моделей с качеством моделей, полученных в пункте 4.

Экранные формы с текстом программы и примерами её выполнения



The screenshot shows the JupyterLab interface with a code cell containing the following Python code:

```
In [1]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier

from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV

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

Below the code cell, the text "Всё тот же датасет" (The same dataset) is visible.

```
In [2]: data = pd.read_csv('heart.csv', sep=',')
data.head()
```

```
Out[2]:
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	260	0	1	167	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Деление данных

```
In [3]: X_train, X_test, y_train, y_test = train_test_split(
data, data['target'], test_size=0.2, random_state=1)
```

Обучение двух ансамблевых моделей, оценка их качества

```
In [70]: random_forest = RandomForestClassifier(n_estimators=10, max_depth=1, random_state=0).fit(X_train, y_train)
```

```
In [66]: gradient_boosting = GradientBoostingClassifier(n_estimators=10, max_depth=10, learning_rate=0.01).fit(X_train, y_train)
```

```
In [41]: target_random_forest = random_forest.predict(X_test)
```

```
In [67]: target_gradient_boosting = gradient_boosting.predict(X_test)
```

```
In [44]: accuracy_score(y_test, target_random_forest), \
precision_score(y_test, target_random_forest)
```

```
Out[44]: (0.9672131147540983, 0.9393939393939394)
```

```
In [71]: accuracy_score(y_test, target_gradient_boosting), \
precision_score(y_test, target_gradient_boosting)
```

```
Out[71]: (0.5081967213114754, 0.5081967213114754)
```

Подбор гиперпараметров

```
In [73]: parameters_random_forest = {'n_estimators': [1, 3, 5, 7, 10],
'max_depth': [1, 3, 5, 7, 10],
'random_state': [0, 2, 4, 6, 8, 10]}
best_random_forest = GridSearchCV(RandomForestClassifier(), parameters_random_forest, cv=3, scoring='a
best_random_forest.fit(X_train, y_train)
```

C:\Anaconda\lib\site-packages\sklearn\model_selection_search.py:841: DeprecationWarning: The default of the 'iid' parameter will change from True to False in version 0.22 and will be removed in 0.24. This will change numeric results when test-set sizes are unequal.
DeprecationWarning)

```
Out[73]: GridSearchCV(cv=3, error_score='raise-deprecating',
estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
max_depth=None, max_features='auto', 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='warn', n_jobs=None,
oob_score=False, random_state=None, verbose=0,
warn_start=False),
fit_params=None, iid='warn', n_jobs=None,
param_grid={'n_estimators': [1, 3, 5, 7, 10], 'max_depth': [1, 3, 5, 7, 10], 'random_state':
[0, 2, 4, 6, 8, 10]},
pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
scoring='accuracy', verbose=0)
```

```
In [78]: parameters_gradient_boosting = {'n_estimators': [1, 3, 5, 7, 10],
'max_depth': [1, 3, 5, 7, 10],
'learning_rate': [0.001, 0.0025, 0.005, 0.0075, 0.01, 0.025]}
best_gradient_boosting = GridSearchCV(GradientBoostingClassifier(), parameters_gradient_boosting, cv=3,
best_gradient_boosting.fit(X_train, y_train)
```

C:\Anaconda\lib\site-packages\sklearn\model_selection_search.py:841: DeprecationWarning: The default of the 'iid' parameter will change from True to False in version 0.22 and will be removed in 0.24. This will change numeric results when test-set sizes are unequal.
DeprecationWarning)

```
Out[78]: GridSearchCV(cv=3, error_score='raise-deprecating',
estimator=GradientBoostingClassifier(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, subsample=1.0, tol=0.0001, validation_fraction=0.
1,
        verbose=0, warn_start=False),
        fit_params=None, iid='warn', n_jobs=None,
        param_grid={'n_estimators': [1, 3, 5, 7, 10], 'max_depth': [1, 3, 5, 7, 10], 'learning_rate':
[0.001, 0.0025, 0.005, 0.0075, 0.01, 0.025]},
        pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
        scoring='accuracy', verbose=0)

```

In [75]: `best_random_forest.best_params_`

Out[75]: {'max_depth': 1, 'n_estimators': 1, 'random_state': 0}

In [79]: `best_gradient_boosting.best_params_`

Out[79]: {'learning_rate': 0.025, 'max_depth': 1, 'n_estimators': 5}

Обучение двух ансамблевых моделей с подобранными гиперпараметрами, оценка их качества

In [77]: `new_random_forest = RandomForestClassifier(n_estimators=1, max_depth=1, random_state=0).fit(X_train, y_train)`

In [82]: `new_gradient_boosting = GradientBoostingClassifier(n_estimators=5, max_depth=1, learning_rate=0.025).fit(X_train, y_train)`

In [83]: `new_target_random_forest = new_random_forest.predict(X_test)`

In [84]: `new_target_gradient_boosting = new_gradient_boosting.predict(X_test)`

In [85]: `accuracy_score(y_test, new_target_random_forest), \
precision_score(y_test, new_target_random_forest)`

Out[85]: (1.0, 1.0)

In [86]: `accuracy_score(y_test, new_target_gradient_boosting), \
precision_score(y_test, new_target_gradient_boosting)`

Out[86]: (1.0, 1.0)