

Московский государственный технический университет им. Н.Э. Баумана  
Факультет «Информатика и системы управления»  
Кафедра «Системы обработки информации и управления»



**Отчёт**

**“Методы машинного обучения”**

**Лабораторная работа № 6**

**“Ансамбли моделей машинного обучения”**

ИСПОЛНИТЕЛЬ:

Студент группы ИУ5-21М

Коростелёв В. М. \_\_\_\_\_

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю. Е. \_\_\_\_\_

**Москва – 2019**

---

# Ансамбли моделей машинного обучения

Цель лабораторной работы: изучение ансамблей моделей машинного обучения. Задание:

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.

```
In [1]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
data = pd.read_csv('heart.csv', sep=",")
data.head(5)
```

Out[1]:

	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	250	0	1	187	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

```
In [2]: data.shape
```

Out[2]: (303, 14)

```
In [3]: # Проверка на пустые значения
data.isnull().sum()
```

```
Out[3]: age          0
sex          0
cp           0
trestbps     0
chol         0
fbs          0
restecg      0
thalach      0
exang        0
oldpeak      0
slope        0
ca           0
thal         0
target       0
dtype: int64
```

```
In [4]: from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, auc
import pylab as pl
from sklearn.preprocessing import MinMaxScaler
import warnings
warnings.filterwarnings('ignore')
```

```
In [5]: # Пустых значений нет
# Перейдем к разделению выборки на обучающую и тестовую.
X = data.drop('target',axis = 1).values
y = data['target'].values
```

## Ансамблевые модели

```
In [6]: from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.model_selection import train_test_split
# Функция train_test_split разделила исходную выборку таким образом,
# чтобы в обучающей и тестовой частях сохранились пропорции классов.
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.35, random_state=1)
```

## "Случайный лес"

```
In [7]: # n_estimators = 10 (default)
rfc = RandomForestClassifier().fit(X_train, y_train)
predicted_rfc = rfc.predict(X_test)
```

```
In [8]: accuracy_score(y_test, predicted_rfc)
```

```
Out[8]: 0.719626168224299
```

```
In [9]: balanced_accuracy_score(y_test, predicted_rfc)
```

```
Out[9]: 0.7233333333333334
```

```
In [10]: (precision_score(y_test, predicted_rfc, average='weighted'),  
         recall_score(y_test, predicted_rfc, average='weighted'))
```

```
Out[10]: (0.7273343111011726, 0.719626168224299)
```

```
In [11]: f1_score(y_test, predicted_rfc, average='weighted')
```

```
Out[11]: 0.7194302396206822
```

## Алгоритм AdaBoost

```
In [12]: # n_estimators = 50 (default)  
         abc = AdaBoostClassifier().fit(X_train, y_train)  
         predicted_abc = abc.predict(X_test)
```

```
In [13]: accuracy_score(y_test, predicted_abc)
```

```
Out[13]: 0.7289719626168224
```

```
In [14]: balanced_accuracy_score(y_test, predicted_abc)
```

```
Out[14]: 0.7284210526315789
```

```
In [15]: (precision_score(y_test, predicted_abc, average='weighted'),  
         recall_score(y_test, predicted_abc, average='weighted'))
```

```
Out[15]: (0.7293842770753162, 0.7289719626168224)
```

```
In [16]: f1_score(y_test, predicted_abc, average='weighted')
```

```
Out[16]: 0.7291144464706996
```

Из двух представленных ансамблевых моделей с параметрами по умолчанию с задачей классификации на выбранном датасете лучше справляется модель "AdaBoost"

## Подбор гиперпараметров

### "Случайный лес"

```
In [17]: rfc_n_range = np.array(range(5,100,5))  
         rfc_tuned_parameters = [{'n_estimators': rfc_n_range}]  
         rfc_tuned_parameters
```

```
Out[17]: [{'n_estimators': array([ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75,  
                                80, 85,  
                                90, 95])}]
```

```
In [18]: import warnings
from sklearn.model_selection import GridSearchCV
warnings.filterwarnings('ignore')

gs_rfc = GridSearchCV(RandomForestClassifier(), rfc_tuned_parameters, cv=5,
                      scoring='accuracy')
gs_rfc.fit(X_train, y_train)
```

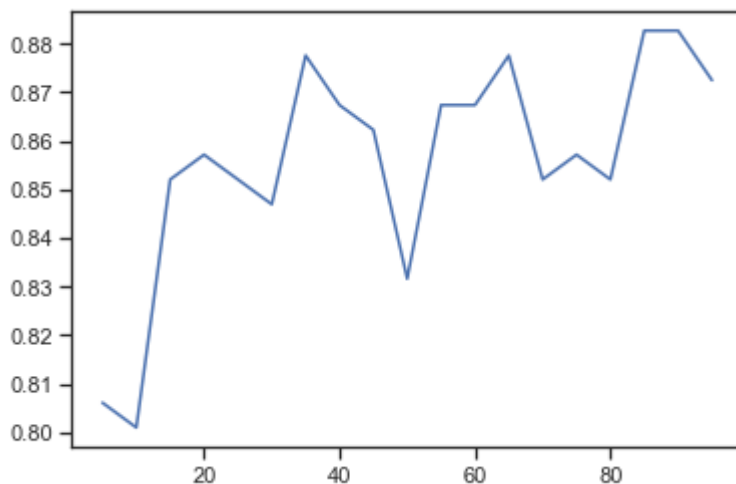
```
Out[18]: GridSearchCV(cv=5, error_score='raise-deprecating',
                      estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion
='gini',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, 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,
                      warm_start=False),
                      fit_params=None, iid='warn', n_jobs=None,
                      param_grid=[{'n_estimators': array([ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 5
5, 60, 65, 70, 75, 80, 85,
                      90, 95])}],
                      pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                      scoring='accuracy', verbose=0)
```

```
In [19]: gs_rfc.best_params_
```

```
Out[19]: {'n_estimators': 85}
```

```
In [20]: plt.plot(rfc_n_range, gs_rfc.cv_results_['mean_test_score'])
```

```
Out[20]: [<matplotlib.lines.Line2D at 0x25c344a6898>]
```



## Алгоритм AdaBoost

```
In [21]: abc_n_range = np.array(range(5,100,5))
abc_tuned_parameters = [{'n_estimators': abc_n_range}]
abc_tuned_parameters
```

```
Out[21]: [{'n_estimators': array([ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75,
80, 85,
                      90, 95])}]
```

```
In [22]: gs_abc = GridSearchCV(AdaBoostClassifier(), abc_tuned_parameters, cv=5,  
                                scoring='accuracy')  
gs_abc.fit(X_train, y_train)
```

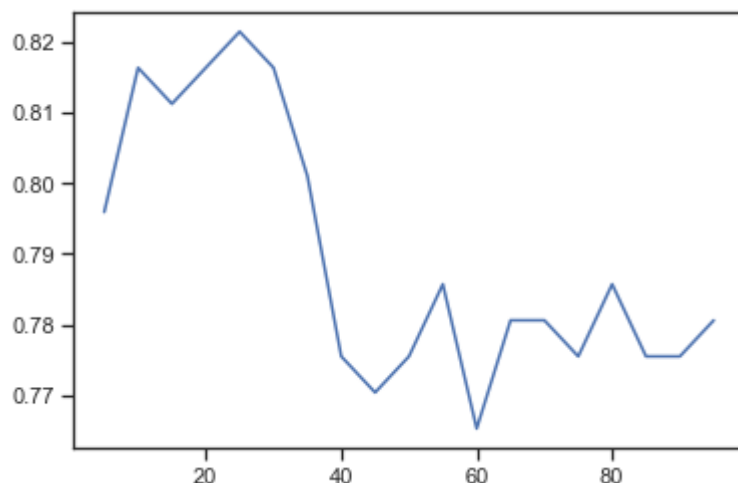
```
Out[22]: GridSearchCV(cv=5, error_score='raise-deprecating',  
                      estimator=AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None,  
                                                    learning_rate=1.0, n_estimators=50, random_state=None),  
                      fit_params=None, iid='warn', n_jobs=None,  
                      param_grid=[{'n_estimators': array([ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 5  
5, 60, 65, 70, 75, 80, 85,  
90, 95])}],  
                      pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',  
                      scoring='accuracy', verbose=0)
```

```
In [23]: gs_abc.best_params_
```

```
Out[23]: {'n_estimators': 25}
```

```
In [24]: plt.plot(abc_n_range, gs_abc.cv_results_['mean_test_score'])
```

```
Out[24]: [matplotlib.lines.Line2D at 0x25c3453ecf8]
```



## Сравнение моделей после подбора гиперпараметров

### "Случайный лес"

```
In [25]: rfc_optimized = RandomForestClassifier(n_estimators=gs_rfc.best_params_['n_estimator  
s']).fit(X_train, y_train)  
predicted_rfc_opt = rfc_optimized.predict(X_test)
```

```
In [26]: accuracy_score(y_test, predicted_rfc_opt)
```

```
Out[26]: 0.7476635514018691
```

```
In [27]: balanced_accuracy_score(y_test, predicted_rfc_opt)
```

```
Out[27]: 0.7471929824561403
```

```
In [28]: (precision_score(y_test, predicted_rfc_opt, average='weighted'),  
          recall_score(y_test, predicted_rfc_opt, average='weighted'))
```

```
Out[28]: (0.748059504175502, 0.7476635514018691)
```

```
In [29]: f1_score(y_test, predicted_rfc_opt, average='weighted')
```

```
Out[29]: 0.7477962087830651
```

## Алгоритм AdaBoost

```
In [30]: abc_optimized = RandomForestClassifier(n_estimators=gs_abc.best_params_['n_estimators']).fit(X_train, y_train)
predicted_abc_opt = abc_optimized.predict(X_test)
```

```
In [31]: accuracy_score(y_test, predicted_abc_opt)
```

```
Out[31]: 0.7383177570093458
```

```
In [32]: balanced_accuracy_score(y_test, predicted_abc_opt)
```

```
Out[32]: 0.7347368421052631
```

```
In [33]: (precision_score(y_test, predicted_abc_opt, average='weighted'),
recall_score(y_test, predicted_abc_opt, average='weighted'))
```

```
Out[33]: (0.7383710473551335, 0.7383177570093458)
```

```
In [34]: f1_score(y_test, predicted_abc_opt, average='weighted')
```

```
Out[34]: 0.7373013358677862
```

Подбор гиперпараметра `n_estimators` для моделей "Случайный лес" и "Алгоритм AdaBoost" позволил увеличить точность классификации.