

Лабораторная работа 6  
по дисциплине  
«Методы машинного обучения»  
на тему  
«Ансамбли моделей машинного обучения.»

Выполнил:  
студент группы ИУ5-21М  
Могильников И. А.

---

# 1. Описание задания

Цель лабораторной работы: изучение ансамблей моделей машинного обучения.

## 1.1. 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.

# 2. Ход выполнения лабораторной работы

## 2.1. Выбор датасета

В качестве исходных данных выбираем датасет Heart Disease UCI (<https://www.kaggle.com/ronitf/heart-disease-uci>). 303 записи, 14 признаков, целевой признак относится к наличию болезни сердца у пациента: 0 - нет болезни сердца, 1 - есть.

```
In [0]: from google.colab import drive, files
        drive.mount('/content/drive')
```

Go to this URL in a browser: [https://accounts.google.com/o/oauth2/auth?client\\_id=9473189898](https://accounts.google.com/o/oauth2/auth?client_id=9473189898)

Enter your authorization code:

.....

Mounted at /content/drive

```
In [0]: from google.colab import files
        import os
        import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline
        os.listdir()
        data = pd.read_csv('drive/My Drive/mmo_datasets/heart.csv',
                           sep=";", encoding="iso-8859-1")
```

```
In [0]: total_count = data.shape[0]
        num_cols = []
        for col in data.columns:
            # Количество пустых значений
            temp_null_count = data[data[col].isnull()].shape[0]
            dt = str(data[col].dtype)
            if temp_null_count>0:
                num_cols.append(col)
                temp_perc = round((temp_null_count / total_count) * 100.0, 2)
                print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'
                      .format(col, dt, temp_null_count, temp_perc))
```

```
data_cleared = data
```

```
In [0]: uniquevalues = np.unique(data_cleared['target'].values)
        uniquevalues
```

```
Out[0]: array([0, 1])
```

## 2.2. train\_test\_split

```
In [0]: target = data_cleared['target']
        data_cleared = data_cleared.drop('target', axis=1)
```

```
In [0]: data_cleared.head(10)
```

```
Out[0]:
```

	age	sex	cp	trestbps	chol	fbs	...	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	...	150	0	2.3	0	0	1
1	37	1	2	130	250	0	...	187	0	3.5	0	0	2
2	41	0	1	130	204	0	...	172	0	1.4	2	0	2
3	56	1	1	120	236	0	...	178	0	0.8	2	0	2
4	57	0	0	120	354	0	...	163	1	0.6	2	0	2
5	57	1	0	140	192	0	...	148	0	0.4	1	0	1
6	56	0	1	140	294	0	...	153	0	1.3	1	0	2
7	44	1	1	120	263	0	...	173	0	0.0	2	0	3
8	52	1	2	172	199	1	...	162	0	0.5	2	0	3
9	57	1	2	150	168	0	...	174	0	1.6	2	0	2

```
[10 rows x 13 columns]
```

```
In [0]: from sklearn.model_selection import train_test_split
        X_train, X_test, Y_train, Y_test = train_test_split(
            data_cleared,
            target,
            test_size=0.2,
            random_state=1
        )
```

```
In [0]: X_train.shape, Y_train.shape
```

```
Out[0]: ((242, 13), (242,))
```

```
In [0]: X_test.shape, Y_test.shape
```

```
Out[0]: ((61, 13), (61,))
```

## 2.3. Обучение

```
In [0]: 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
```

### 2.3.1. Случайный лес

```
In [0]: # n_estimators = 10 (default)
rfc = RandomForestClassifier().fit(X_train, Y_train)
predicted_rfc = rfc.predict(X_test)
```

/usr/local/lib/python3.6/dist-packages/sklearn/ensemble/forest.py:246: FutureWarning: The default value of n\_estimators will change from 10 in version 0.20 to 100 in 0.22.", FutureWarning)

```
In [0]: accuracy_score(Y_test, predicted_rfc)
```

```
Out[0]: 0.7377049180327869
```

```
In [0]: balanced_accuracy_score(Y_test, predicted_rfc)
```

```
Out[0]: 0.7370967741935484
```

```
In [0]: (precision_score(Y_test, predicted_rfc, average='weighted'),
        recall_score(Y_test, predicted_rfc, average='weighted'))
```

```
Out[0]: (0.7384500745156483, 0.7377049180327869)
```

```
In [0]: f1_score(Y_test, predicted_rfc, average='weighted')
```

```
Out[0]: 0.7372809496890899
```

### 2.3.2. Алгоритм AdaBoost

```
In [0]: # n_estimators = 50 (default)
abc = AdaBoostClassifier().fit(X_train, Y_train)
predicted_abc = abc.predict(X_test)
```

```
In [0]: accuracy_score(Y_test, predicted_abc)
```

```
Out[0]: 0.6721311475409836
```

```
In [0]: balanced_accuracy_score(Y_test, predicted_abc)
```

```
Out[0]: 0.6720430107526881
```

```
In [0]: (precision_score(Y_test, predicted_abc, average='weighted'),
        recall_score(Y_test, predicted_abc, average='weighted'))
```

```
Out[0]: (0.6721311475409836, 0.6721311475409836)
```

```
In [0]: f1_score(Y_test, predicted_abc, average='weighted')
```

```
Out[0]: 0.6721311475409836
```

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

## 2.4. Подбор гиперпараметров

### 2.4.1. Случайный лес

```
In [0]: rfc_n_range = np.array(range(5,100,5))
```

```
rfc_tuned_parameters = [{'n_estimators': rfc_n_range}]
rfc_tuned_parameters
```

```
Out[0]: [{'n_estimators': array([ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85,
 90, 95])}]
```

```
In [0]: import warnings
```

```
warnings.filterwarnings('ignore')
```

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

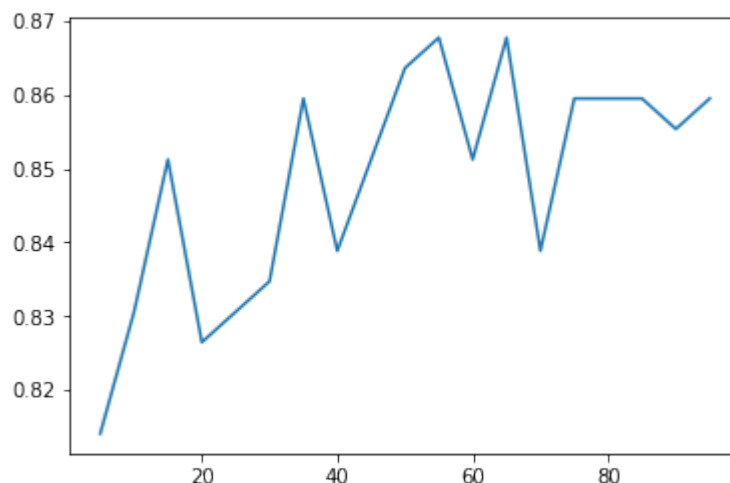
```
Out[0]: GridSearchCV(cv=5, error_score='raise-deprecating',
                    estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='g
                    max_depth=None, max_features='auto', max_leaf_nodes=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    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, 55, 60, 65,
                    90, 95])}],
                    pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                    scoring='accuracy', verbose=0)
```

```
In [0]: gs_rfc.best_params_
```

```
Out[0]: {'n_estimators': 55}
```

```
In [0]: plt.plot(rfc_n_range, gs_rfc.cv_results_['mean_test_score'])
```

```
Out[0]: [<matplotlib.lines.Line2D at 0x7f90250ff860>]
```



### 2.4.2. Алгоритм AdaBoost

```
In [0]: abc_n_range = np.array(range(5,100,5))
        abc_tuned_parameters = [{'n_estimators': abc_n_range}]
        abc_tuned_parameters
```

```
Out[0]: [{'n_estimators': array([ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85,
                                   90, 95])}]
```

```
In [0]: gs_abc = GridSearchCV(AdaBoostClassifier(), abc_tuned_parameters, cv=5,
                               scoring='accuracy')
        gs_abc.fit(X_train, Y_train)
```

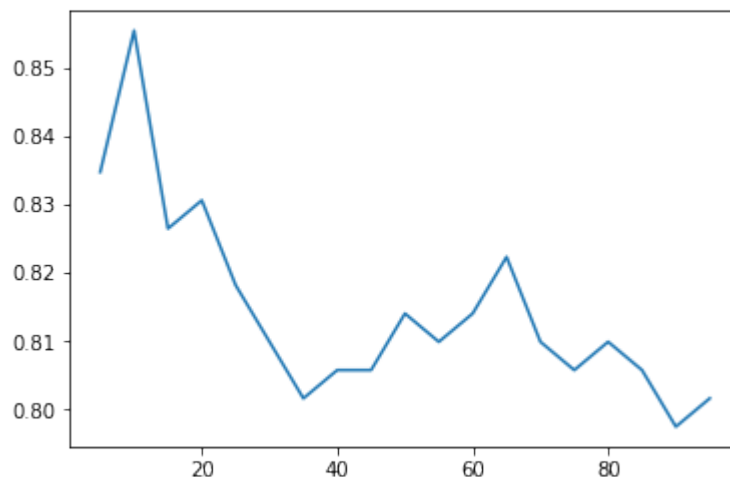
```
Out[0]: GridSearchCV(cv=5, error_score='raise-deprecating',
                     estimator=AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None,
                                                  learning_rate=1.0, n_estimators=50, random_state=None),
                     fit_params=None, iid='warn', n_jobs=None,
                     param_grid=[{'n_estimators': array([ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65,
                                                           70, 75, 80, 85, 90, 95])}],
                     pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                     scoring='accuracy', verbose=0)
```

```
In [0]: gs_abc.best_params_
```

```
Out[0]: {'n_estimators': 10}
```

```
In [0]: plt.plot(abc_n_range, gs_abc.cv_results_['mean_test_score'])
```

```
Out[0]: [<matplotlib.lines.Line2D at 0x7f9025103e80>]
```



## 2.5. Сравнение моделей после подбора гиперпараметров

### 2.5.1. Случайный лес

```
In [0]: rfc_optimized = RandomForestClassifier(n_estimators=gs_rfc.best_params_['n_estimators'])
        predicted_rfc_opt = rfc_optimized.predict(X_test)
```

```
In [0]: accuracy_score(Y_test, predicted_rfc_opt)
```

```
Out[0]: 0.7704918032786885
```

```
In [0]: balanced_accuracy_score(Y_test, predicted_rfc_opt)
```

```
Out[0]: 0.7688172043010753
```

```
In [0]: (precision_score(Y_test, predicted_rfc_opt, average='weighted'),  
        recall_score(Y_test, predicted_rfc_opt, average='weighted'))
```

```
Out[0]: (0.7806823216659282, 0.7704918032786885)
```

```
In [0]: f1_score(Y_test, predicted_rfc_opt, average='weighted')
```

```
Out[0]: 0.7678667095253402
```

### 2.5.2. Алгоритм AdaBoost

```
In [0]: abc_optimized = RandomForestClassifier(n_estimators=gs_abc.best_params_['n_estimators'],  
        predicted_abc_opt = abc_optimized.predict(X_test)
```

```
In [0]: accuracy_score(Y_test, predicted_abc_opt)
```

```
Out[0]: 0.7213114754098361
```

```
In [0]: balanced_accuracy_score(Y_test, predicted_abc_opt)
```

```
Out[0]: 0.7209677419354839
```

```
In [0]: (precision_score(Y_test, predicted_abc_opt, average='weighted'),  
        recall_score(Y_test, predicted_abc_opt, average='weighted'))
```

```
Out[0]: (0.7213998021481063, 0.7213114754098361)
```

```
In [0]: f1_score(Y_test, predicted_abc_opt, average='weighted')
```

```
Out[0]: 0.7211615219395159
```

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