Московский Государственный Технический Университет
им. Н. Э. Баумана

Лабораторная работа №6
по курсу: «Технологии машинного обучения»

# Ансамбли моделей машинного обучения.

Выполнила:
Студентка группы ИУ5-63
Нурлыева Д.Д.

Москва
2019

**Задание:**

1. Выберите набор данных (датасет) для решения задачи классификации или регресии.
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.

**Текстовое описание набора данных**

В качестве набора данных мы будем использовать набор данных Heart Disease UCI - https://www.kaggle.com/ronitf/heart-disease-uci В датасете отражено наличие сердечного заболевания у пациента в зависимости от разных признаков.
Датасет содержит следующие колонки:
age - возраст в годах
sex - (1 = мужчина; 0 = женщина)
cp - тип боли в груди
trestbps - артериальное давление в состоянии покоя (в мм рт. ст. при поступлении в стационар)
chol - холестерин в мг/дл
fbs - уровень сахара в крови натощак > 120 мг / дл) (1 = да; 0 = нет)
restecg- электрокардиографические результаты покоя
thalach - максимальная ЧСС
exang - стенокардия, вызванная физическими упражнениями (1 = да; 0 = Нет)
oldpeak - Депрессия, вызванная физическими упражнениями относительно покоя
slope - наклон пика упражнения сегмента
ca - количество крупных сосудов (0-3)
thal - 3 = нормальный; 6 = фиксированный дефект; 7 = реверзибельный дефект
target - заболевание 1-есть или 0-нет

**Текст программы:**

```
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
```

In [3]:

```
data=pd.read_csv("/Users/user/Desktop/data2.csv")
data.head()
```

Out[3]:

| | 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 [4]:

```
target = data['target']
data = data.drop('target', axis = 1)
```

In [5]:

```
#деление на тестовую и обучающую выборку
X_train, X_test, y_train, y_test = train_test_split(
    data, target, test_size=0.2, random_state=1)
```

In [6]:

```
#алгоритм случайного леса
random_forest = RandomForestClassifier(n_estimators=10, max_depth=1,
random_state=0).fit(X_train, y_train)
```

In [7]:

```
#градиентный бустинг
gradient_boosting = GradientBoostingClassifier(n_estimators=10,
max_depth=10, learning_rate=0.01).fit(X_train, y_train)
```

In [8]:

```
target_random_forest = random_forest.predict(X_test)
```

In [9]:

```
target_gradient_boosting = gradient_boosting.predict(X_test)
```

In [10]:

```
accuracy_score(y_test, target_random_forest), \
precision_score(y_test, target_random_forest)
```

Out[10]:

```
(0.8032786885245902, 0.7567567567567568)
```

In [11]:

```
accuracy_score(y_test, target_gradient_boosting), \
precision_score(y_test, target_gradient_boosting)
```

Out[11]:

```
(0.5081967213114754, 0.5081967213114754)
```

In [12]:

```
#Подбираем гиперпараметры
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='accuracy')
best_random_forest.fit(X_train, y_train)
```

```
/anaconda3/lib/python3.6/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[12]:

```
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,
            warm_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 [13]:

```
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,
            warm_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)
```

Out[13]:

```
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,
```

```
            warm_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 [14]:

```
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, scoring='accuracy')
best_gradient_boosting.fit(X_train, y_train)
```

```
/anaconda3/lib/python3.6/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[14]:

```
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_sampl...      subsample=1.0,
tol=0.0001, validation_fraction=0.1,
              verbose=0, warm_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 [15]:

```
best_random_forest.best_params_
```

Out[15]:

```
{'max_depth': 3, 'n_estimators': 5, 'random_state': 10}
```

In [16]:

```
best_gradient_boosting.best_params_
```

Out[16]:

```
{'learning_rate': 0.025, 'max_depth': 5, 'n_estimators': 10}
```

In [17]:

```
new_random_forest = RandomForestClassifier(n_estimators=5, max_depth=3,
random_state=10).fit(X_train, y_train)
```

In [18]:

```
new_gradient_boosting = GradientBoostingClassifier(n_estimators=10,
max_depth=3, learning_rate=0.025).fit(X_train, y_train)
```

In [19]:

```
new_target_random_forest = new_random_forest.predict(X_test)
```

In [20]:

```
new_target_gradient_boosting = new_gradient_boosting.predict(X_test)
```

In [21]:

```
accuracy_score(y_test, new_target_random_forest), \
precision_score(y_test, new_target_random_forest)
```

Out[21]:

```
(0.7377049180327869, 0.7027027027027027)
```

In [22]:

```
accuracy_score(y_test, new_target_gradient_boosting), \
precision_score(y_test, new_target_gradient_boosting)
```

Out[22]:

```
(0.7377049180327869, 0.6744186046511628)
```