

Лабораторная работа №5
по дисциплине
«Методы машинного обучения»
на тему
«Линейные модели, SVM и деревья решений»

Выполнил:
студент группы ИУ5-21М
Якубов А. Р.

1. Цель лабораторной работы

Изучение линейных моделей, SVM и деревьев решений. [1]

2. 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 1) одну из линейных моделей, 2) SVM и 3) дерево решений. Оцените качество моделей с помощью трех подходящих для задачи метрик. Сравните качество полученных моделей.
5. Произведите для каждой модели подбор одного гиперпараметра с использованием `GridSearchCV` и кросс-валидации.
6. Повторите пункт 4 для найденных оптимальных значений гиперпараметров. Сравните качество полученных моделей с качеством моделей, полученных в пункте 4.

3. Выполнение

3.1. Загрузка датасета

Выбранный набор: Classifying wine varieties.

```
In [6]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import learning_curve, validation_curve
from sklearn.model_selection import KFold, RepeatedKFold, LeaveOneOut,
from sklearn.model_selection import cross_val_score, cross_validate
from sklearn.metrics import roc_curve, confusion_matrix, roc_auc_score,

from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LinearRegression

import warnings

warnings.filterwarnings('ignore')
plt.style.use('ggplot')

In [7]: data = pd.read_csv('Wine.csv', sep=";")
data.head()
```

```

Out[7]:
  Class  Alcohol  Malic acid  Ash  Alcalinity of ash  Magnesium \
0      1    14.23         1.71  2.43              15.6        127
1      1    13.20         1.78  2.14              11.2        100
2      1    13.16         2.36  2.67              18.6        101
3      1    14.37         1.95  2.50              16.8        113
4      1    13.24         2.59  2.87              21.0        118

      Total phenols  Flavanoids  Nonflavanoid phenols  Proanthocyanins \
0              2.80         3.06              0.28        2.29
1              2.65         2.76              0.26        1.28
2              2.80         3.24              0.30        2.81
3              3.85         3.49              0.24        2.18
4              2.80         2.69              0.39        1.82

      Color intensity  Hue  OD280/OD315 of diluted wines  Proline
0              5.64  1.04              3.92        1065
1              4.38  1.05              3.40        1050
2              5.68  1.03              3.17        1185
3              7.80  0.86              3.45        1480
4              4.32  1.04              2.93        735

```

Колонки и их типы данных

```
In [8]: data.dtypes
```

```

Out[8]:
Class                int64
Alcohol              float64
Malic acid            float64
Ash                  float64
Alcalinity of ash    float64
Magnesium            int64
Total phenols        float64
Flavanoids            float64
Nonflavanoid phenols float64
Proanthocyanins       float64
Color intensity       float64
Hue                   float64
OD280/OD315 of diluted wines float64
Proline               int64
dtype: object

```

Проверка на пустые значение

```
In [9]: for col in data.columns:
        print('{} - {}'.format(col, data[data[col].isnull()].shape[0]))
```

```

Class - 0
Alcohol - 0
Malic acid - 0
Ash - 0
Alcalinity of ash - 0

```

```
Magnesium - 0
Total phenols - 0
Flavanoids - 0
Nonflavanoid phenols - 0
Proanthocyanins - 0
Color intensity - 0
Hue - 0
OD280/OD315 of diluted wines - 0
Proline - 0
```

```
In [10]: data.shape
```

```
Out[10]: (178, 14)
```

```
In [11]: CLASS = 'Class'
RANDOM_STATE = 17
TEST_SIZE = 0.3
```

```
X = data.drop(CLASS, axis=1).values
Y = data[CLASS].values
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3,
                                                    random_state=RANDOM_STATE)
print('X_train: {}'.format(X_train.shape))
print('X_test: {}'.format(X_test.shape))
```

```
X_train: (124, 13)
```

```
X_test: (54, 13)
```

3.2. Обучение

Машина опорных векторов

```
In [12]: clf = SVC(gamma='auto')
clf.fit(X_train, Y_train)
clf.score(X_test, Y_test)
```

```
Out[12]: 0.3888888888888889
```

```
In [13]: Y_pred = clf.predict(X_test)
print(classification_report(Y_test, Y_pred))
```

	precision	recall	f1-score	support
1	0.00	0.00	0.00	18
2	0.39	1.00	0.56	21
3	0.00	0.00	0.00	15
micro avg	0.39	0.39	0.39	54
macro avg	0.13	0.33	0.19	54
weighted avg	0.15	0.39	0.22	54

Дерево решений

```
In [14]: tree = DecisionTreeClassifier(random_state=0)
         tree.fit(X_train, Y_train)
         tree.score(X_test, Y_test)
```

```
Out[14]: 0.98148148148148151
```

```
In [15]: Y_pred = tree.predict(X_test)
         print(classification_report(Y_test, Y_pred))
```

	precision	recall	f1-score	support
1	0.95	1.00	0.97	18
2	1.00	0.95	0.98	21
3	1.00	1.00	1.00	15
micro avg	0.98	0.98	0.98	54
macro avg	0.98	0.98	0.98	54
weighted avg	0.98	0.98	0.98	54

Линейная регрессия

```
In [16]: lin = LinearRegression()
         lin.fit(X_train, Y_train)
         lin.score(X_test, Y_test)
```

```
Out[16]: 0.88205015361986894
```

3.3. Подбор гиперпараметра с использованием GridSearchCV и кросс-валидации

Машина опорных векторов

```
In [17]: CROSS_VALIDATOR_GENERATOR = 5
         PARAMETER_TAG = 'C'
         PARAMETER_MAX_VALUE = 3

         param_grid = {PARAMETER_TAG : np.arange(0.01, PARAMETER_MAX_VALUE, 0.01)}
         clf = SVC(gamma='auto')

         clf_cv = GridSearchCV(clf, param_grid, cv = CROSS_VALIDATOR_GENERATOR)
         clf_cv.fit(X_train, Y_train)
         clf_cv.best_score_
```

```
Out[17]: 0.47580645161290325
```

```
In [18]: clf_cv.best_params_
```

```
Out[18]: {'C': 1.21}
```

```
In [19]: clf = SVC(gamma='auto', C = clf_cv.best_params_[PARAMETER_TAG])
        clf.fit(X_train, Y_train)
        clf.score(X_test, Y_test)
```

```
Out[19]: 0.40740740740740738
```

```
In [20]: Y_pred = clf.predict(X_test)
        print(classification_report(Y_test, Y_pred))
```

	precision	recall	f1-score	support
1	1.00	0.06	0.11	18
2	0.40	1.00	0.57	21
3	0.00	0.00	0.00	15
micro avg	0.41	0.41	0.41	54
macro avg	0.47	0.35	0.22	54
weighted avg	0.49	0.41	0.26	54

Дерево решений

```
In [21]: PARAMETER_TAG = 'min_impurity_decrease'
```

```
param_grid = {PARAMETER_TAG : np.arange(0.01, PARAMETER_MAX_VALUE, 0.01)}
tree = DecisionTreeClassifier(random_state=0)
```

```
tree_cv = GridSearchCV(tree, param_grid, cv = CROSS_VALIDATOR_GENERATOR)
tree_cv.fit(X_train, Y_train)
tree_cv.best_score_
```

```
Out[21]: 0.91935483870967738
```

```
In [22]: tree_cv.best_params_
```

```
Out[22]: {'min_impurity_decrease': 0.050000000000000003}
```

```
In [23]: tree = DecisionTreeClassifier(random_state=0, min_impurity_decrease = 0.05)
        tree.fit(X_train, Y_train)
        tree.score(X_test, Y_test)
```

```
Out[23]: 0.92592592592592593
```

```
In [24]: Y_pred = tree.predict(X_test)
        print(classification_report(Y_test, Y_pred))
```

	precision	recall	f1-score	support
1	0.82	1.00	0.90	18
2	1.00	0.81	0.89	21

	3	1.00	1.00	1.00	15
micro avg		0.93	0.93	0.93	54
macro avg		0.94	0.94	0.93	54
weighted avg		0.94	0.93	0.93	54

Линейная регрессия

```
In [25]: PARAMETER_TAG = 'n_jobs'
```

```
param_grid = {PARAMETER_TAG : np.arange(1, 100)}
lin = LinearRegression()
```

```
lin_cv = GridSearchCV(lin, param_grid, cv = CROSS_VALIDATOR_GENERATOR)
lin_cv.fit(X_train, Y_train)
lin_cv.best_score_
```

```
Out[25]: 0.86736626705757613
```

```
In [26]: lin_cv.best_params_
```

```
Out[26]: {'n_jobs': 1}
```

```
In [27]: lin = LinearRegression(n_jobs = lin_cv.best_params_[PARAMETER_TAG])
lin.fit(X_train, Y_train)
lin.score(X_test, Y_test)
```

```
Out[27]: 0.88205015361986894
```

Результаты

Наилучший результат показало дерево решений.

Наихудший - машина опорных векторов

Список литературы

- [1] Гапанюк Ю. Е. Лабораторная работа «Линейные модели, SVM и деревья решений» [Электронный ресурс] // GitHub. — 2019. — Режим доступа: https://github.com/ugapanyuk/ml_course/wiki/LAB_TREES (дата обращения: 19.04.2019).