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 № 5

“Линейный модели, SVM и деревья решений”

ИСПОЛНИТЕЛЬ:

Студент группы ИУ5-21М

Коростелёв В. М. _____

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю. Е. _____

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 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.

Датасет: wine (https://www.kaggle.com/brynja/wineuci/downloads/wineuci.zip/1)

In [1]:

```
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, LeavePOut, Shuff
leSplit, StratifiedKFold
from sklearn.model_selection import cross_val_score, cross_validate
from sklearn.metrics import roc_curve,confusion_matrix, roc_auc_score, accuracy_score,
balanced_accuracy_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')
```

Получение данных

In [2]:

```
data = pd.read_csv('Wine.csv', sep=";")
data.head()
```

Out[2]:

	Class	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	P
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	

Колонки и их типы данных

In [3]:

```
data.dtypes
```

Out[3]:

```
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 [4]:

```
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 [5]:

```
data.shape
```

Out[5]:

```
(178, 14)
```

In [6]:

```
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=TEST_SIZE, random_s
tate=RANDOM_STATE, stratify=Y)
print('X_train: {}'.format(X_train.shape))
print('X_test: {}'.format(X_test.shape))
```

```
X_train: (124, 13)
X_test: (54, 13)
```

Обучение

Машина опрных векторов

In [7]:

```
clf = SVC(gamma='auto')
clf.fit(X_train, Y_train)
clf.score(X_test, Y_test)
```

Out[7]:

```
0.3888888888888889
```

In [8]:

```
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 [9]:

```
tree = DecisionTreeClassifier(random_state=0)
tree.fit(X_train, Y_train)
tree.score(X_test, Y_test)
```

Out[9]:

```
0.9814814814814815
```

In [10]:

```
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 [11]:

```
lin = LinearRegression()
lin.fit(X_train, Y_train)
lin.score(X_test, Y_test)
```

Out[11]:

```
0.8820501536198689
```

Подбор гиперпараметра с использованием GridSearchCV и кросс-валидации

Машина опрных векторов

In [12]:

```
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[12]:

```
0.47580645161290325
```

In [13]:

```
clf_cv.best_params_
```

Out[13]:

```
{'C': 1.21}
```

In [14]:

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

Out[14]:

```
0.4074074074074074
```

In [15]:

```
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 [16]:

```
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[16]:

```
0.9193548387096774
```

In [17]:

```
tree_cv.best_params_
```

Out[17]:

```
{'min_impurity_decrease': 0.05}
```

In [18]:

```
tree = DecisionTreeClassifier(random_state=0, min_impurity_decrease = tree_cv.best_para
ms_[PARAMETER_TAG])
tree.fit(X_train, Y_train)
tree.score(X_test, Y_test)
```

Out[18]:

```
0.9259259259259259
```

In [19]:

```
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 [20]:

```
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[20]:

```
0.8673662670575761
```

In [21]:

```
lin_cv.best_params_
```

Out[21]:

```
{'n_jobs': 1}
```

In [22]:

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

Out[22]:

```
0.8820501536198689
```

Результаты

Наилиучший результат показо дерево решений.

Наихудший - машина опрных векторов