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

Отчет

Лабораторная работа № 4

По курсу «Технологии машинного обучения»

ИСПОЛНИТЕЛЬ:
Группа ИУ5-65Б
Погосян С.Л.

" " ______ 2021 г.

ПРЕПОДАВАТЕЛЬ:
Гапанюк Ю.Е.

____

"__"___________2021 г.

Москва 2021

In [1]:

```
import numpy as np
import pandas as pd
from sklearn.datasets import *
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt

from operator import itemgetter
import matplotlib.ticker as ticker
import math

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import precision_score, recall_score, f1_score, classifi
from sklearn.metrics import confusion_matrix
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squ
from sklearn.metrics import roc_curve, roc_auc_score


from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier

from sklearn.model_selection import cross_val_score, cross_validate
from sklearn.model_selection import KFold, RepeatedKFold, LeaveOneOut, LeaveP



from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer


from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.model_selection import learning_curve, validation_curve

from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import SGDRegressor
from sklearn.linear_model import SGDClassifier
from typing import Dict, Tuple
from scipy import stats
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, Linea
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, expor

%matplotlib inline
sns.set(style="ticks")
```

Выборка датасета и ее разделение на тестовую и обучающую

In [2]:

```
wine = load_wine()
```

In [3]:

```
for x in wine:
    print(x)
```

```
data
target
frame
target_names
```

DESCR
feature_names

In [4]:
```
# Сформируем DataFrame
wine_df = pd.DataFrame(data= np.c_[wine['data']],
                     columns= wine['feature_names'])
```

In [5]:
```
wine_df
```

Out[5]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavan
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	
...	...	...	...	...	...	...	...	
173	13.71	5.65	2.45	20.5	95.0	1.68	0.61	
174	13.40	3.91	2.48	23.0	102.0	1.80	0.75	
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	
176	13.17	2.59	2.37	20.0	120.0	1.65	0.68	
177	14.13	4.10	2.74	24.5	96.0	2.05	0.76	

178 rows × 13 columns

In [6]:
```
X_train, X_test, Y_train, Y_test = train_test_split(
    wine.data, wine.target, test_size=0.35, random_state=1)
```

Обучение моделей

Обучение линейной модели

In [7]:
```
reg1 = LogisticRegression(max_iter=10000).fit(X_train, Y_train)
```

In [8]:
```
target1 = reg1.predict(X_test)
```

In [9]:
```
accuracy_score(Y_test, target1), precision_score(Y_test, target1, average='ma
```

Out[9]:
```
(0.9206349206349206, 0.9381499726327313)
```

In [10]:
```
Y_test
```

Out[10]:
```
array([2, 1, 0, 1, 0, 2, 1, 0, 2, 1, 0, 0, 1, 0, 1, 1, 2, 0, 1, 0, 0, 1,
       2, 1, 0, 2, 0, 0, 0, 2, 1, 2, 2, 0, 1, 1, 1, 1, 1, 0, 0, 1, 2, 0,
       0, 0, 1, 0, 0, 0, 1, 2, 2, 0, 1, 1, 0, 1, 2, 1, 1, 0, 2])
```

In [11]:
```
target1
```

Out[11]:
```
array([2, 1, 0, 0, 0, 2, 1, 0, 2, 1, 0, 0, 1, 0, 1, 1, 2, 0, 1, 0, 0, 1,
       1, 0, 0, 2, 0, 0, 0, 2, 1, 2, 2, 0, 1, 1, 1, 1, 1, 0, 0, 1, 2, 0,
       0, 0, 0, 0, 0, 0, 1, 2, 2, 0, 1, 0, 0, 1, 2, 1, 1, 0, 2])
```

In [12]:
```
def accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray) -> Dict[int, float]:
    """
    Вычисление метрики accuracy для каждого класса
    y_true - истинные значения классов
    y_pred - предсказанные значения классов
    Возвращает словарь: ключ - метка класса,
    значение - Accuracy для данного класса
    """
    # Для удобства фильтрации сформируем Pandas DataFrame
    d = {'t': y_true, 'p': y_pred}
    df = pd.DataFrame(data=d)
    # Метки классов
    classes = np.unique(y_true)
    # Результирующий словарь
    res = dict()
    # Перебор меток классов
    for c in classes:
        # отфильтруем данные, которые соответствуют
        # текущей метке класса в истинных значениях
        temp_data_flt = df[df['t']==c]
        # расчет accuracy для заданной метки класса
        temp_acc = accuracy_score(
            temp_data_flt['t'].values,
            temp_data_flt['p'].values)
        # сохранение результата в словарь
        res[c] = temp_acc
    return res

def print_accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray):
    """
    Вывод метрики accuracy для каждого класса
    """
    accs = accuracy_score_for_classes(y_true, y_pred)
    if len(accs)>0:
        print('Метка \t Accuracy')
    for i in accs:
        print('{} \t {}'.format(i, accs[i]))
```

In [13]:
```
print_accuracy_score_for_classes(Y_test, target1)
```

```
Метка    Accuracy
0        1.0
1        0.8333333333333334
2        0.9285714285714286
```

Обучение SVM

In [14]:
```
model_svc = LinearSVC(C=1.0, max_iter=10000)
model_svc.fit(X_train, Y_train)
target2 = model_svc.predict(X_test)
```

```
/home/zeus/anaconda3/envs/tml_env/lib/python3.9/site-packages/sklearn/svm/_ba
se.py:985: ConvergenceWarning: Liblinear failed to converge, increase the num
ber of iterations.
  warnings.warn("Liblinear failed to converge, increase "
```

In [15]:
```
accuracy_score(Y_test, target2), precision_score(Y_test, target2, average='ma
```

Out[15]:
```
(0.7619047619047619, 0.8717948717948718)
```

In [16]:
```
Y_test
```

Out[16]:
```
array([2, 1, 0, 1, 0, 2, 1, 0, 2, 1, 0, 0, 1, 0, 1, 1, 2, 0, 1, 0, 0, 1,
       2, 1, 0, 2, 0, 0, 0, 2, 1, 2, 2, 0, 1, 1, 1, 1, 1, 0, 0, 1, 2, 0,
       0, 0, 1, 0, 0, 0, 1, 2, 2, 0, 1, 1, 0, 1, 2, 1, 1, 0, 2])
```

In [17]:
```
target2
```

Out[17]:
```
array([2, 1, 1, 1, 0, 2, 1, 0, 2, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1,
       1, 1, 1, 2, 1, 1, 0, 2, 1, 2, 2, 0, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1,
       1, 1, 1, 0, 0, 0, 1, 2, 2, 0, 1, 1, 1, 1, 2, 1, 1, 1, 2])
```

In [18]:
```
print_accuracy_score_for_classes(Y_test, target2)
```

```
Метка     Accuracy
0         0.48
1         1.0
2         0.8571428571428571
```

Обучение деревья решений

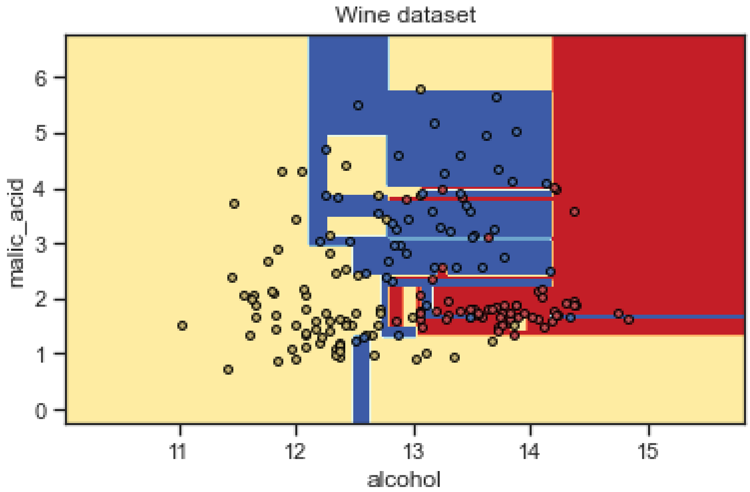
Классификация

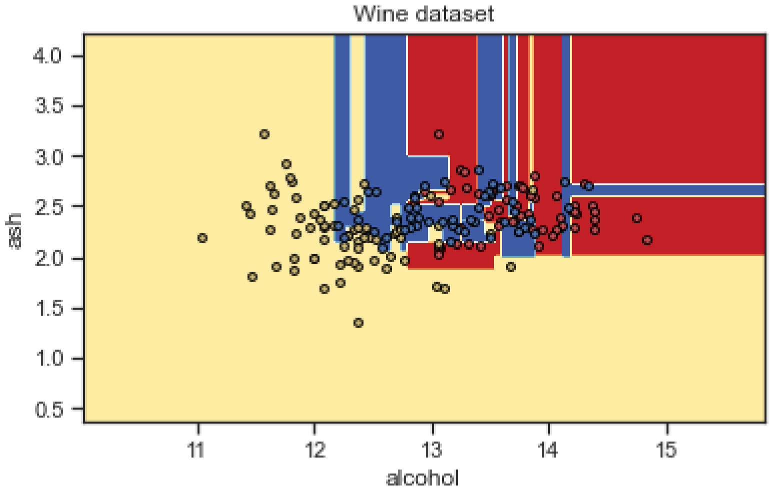
In [19]:
```
def plot_tree_classification(title_param, ds):
    """
    Построение деревьев и вывод графиков для заданного датасета
    """

    n_classes = len(np.unique(ds.target))
    plot_colors = "ryb"
    plot_step = 0.02

    for pairidx, pair in enumerate([[0, 1], [0, 2], [0, 3],
                                    [1, 2], [1, 3], [2, 3]]):
        # We only take the two corresponding features
        X = ds.data[:, pair]
        y = ds.target

        # Train
        clf = DecisionTreeClassifier(random_state=1).fit(X, y)

        plt.title(title_param)

        x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
        y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
        xx, yy = np.meshgrid(np.arange(x_min, x_max, plot_step),
                             np.arange(y_min, y_max, plot_step))
        plt.tight_layout(h_pad=0.5, w_pad=0.5, pad=2.5)

        Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
```

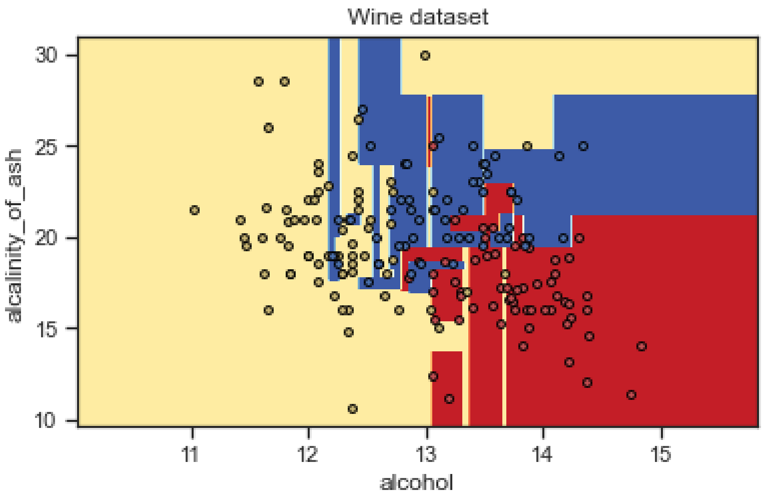
```
        Z = Z.reshape(xx.shape)
        cs = plt.contourf(xx, yy, Z, cmap=plt.cm.RdYlBu)

        plt.xlabel(ds.feature_names[pair[0]])
        plt.ylabel(ds.feature_names[pair[1]])

        # Plot the training points
        for i, color in zip(range(n_classes), plot_colors):
            idx = np.where(y == i)
            plt.scatter(X[idx, 0], X[idx, 1], c=color, label=ds.target_names[
                        cmap=plt.cm.RdYlBu, edgecolor='black', s=15)

        plt.show()
```

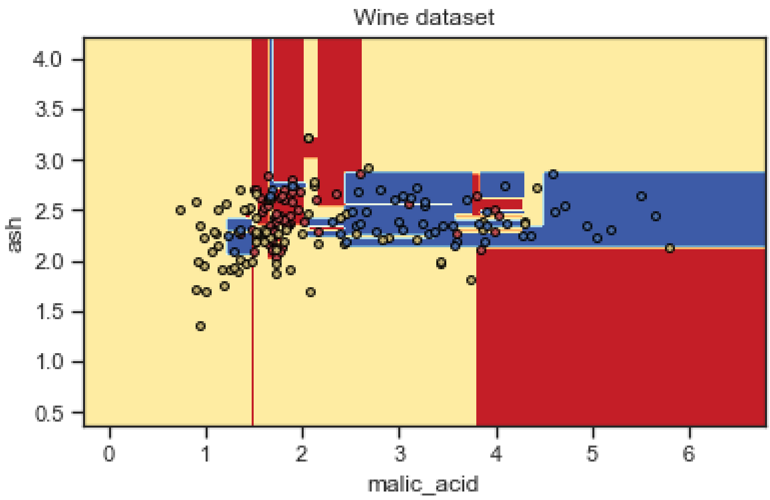
In [20]:

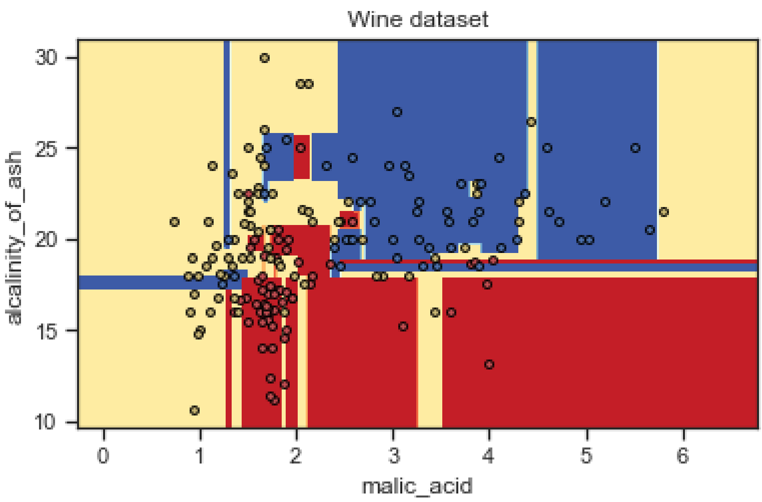
```
plot_tree_classification('Wine dataset', wine)
```

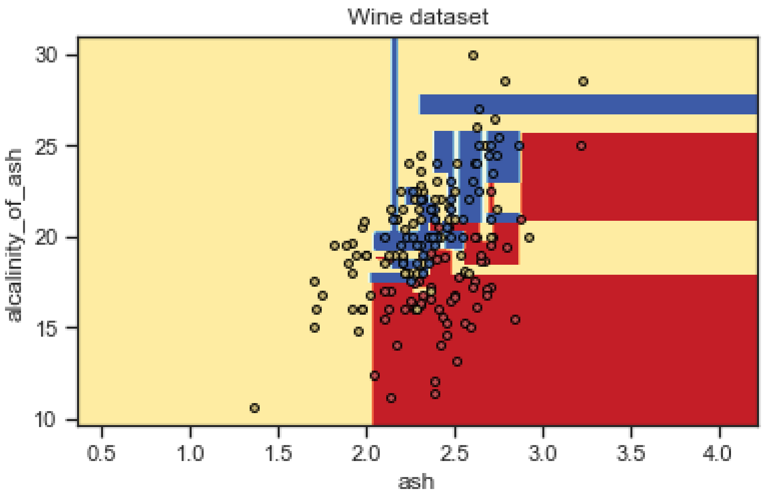





Wine dataset
alcalinity_of_ash
alcohol
10
15
20
25
30
11
12
13
14
15


Wine dataset
ash
malic_acid
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0
1
2
3
4
5
6


Wine dataset
alcalinity_of_ash
malic_acid
10
15
20
25
30
0
1
2
3
4
5
6



In [21]:
```
clf = DecisionTreeClassifier(random_state=1).fit(X_train, Y_train)
target3 = clf.predict(X_test)
accuracy_score(Y_test, target3), precision_score(Y_test, target3, average='ma
```

Out[21]:
```
(0.9206349206349206, 0.9221256038647342)
```

In [22]:
```
Y_test
```

Out[22]:
```
array([2, 1, 0, 1, 0, 2, 1, 0, 2, 1, 0, 0, 1, 0, 1, 1, 2, 0, 1, 0, 0, 1,
       2, 1, 0, 2, 0, 0, 0, 2, 1, 2, 2, 0, 1, 1, 1, 1, 1, 0, 0, 1, 2, 0,
       0, 0, 1, 0, 0, 0, 1, 2, 2, 0, 1, 1, 0, 1, 2, 1, 1, 0, 2])
```

In [23]:
```
target3
```

Out[23]:
```
array([2, 1, 0, 1, 0, 2, 1, 0, 2, 1, 0, 1, 1, 0, 1, 1, 2, 0, 1, 0, 0, 1,
       2, 0, 0, 2, 0, 0, 0, 2, 1, 2, 2, 0, 1, 1, 1, 1, 1, 0, 0, 2, 2, 1,
       0, 0, 1, 0, 0, 0, 1, 2, 2, 0, 1, 1, 0, 1, 2, 1, 0, 0, 2])
```

In [24]:
```
print_accuracy_score_for_classes(Y_test, target3)
```

```
Метка     Accuracy
0         0.92
1         0.875
2         1.0
```

Итоги

In [25]:
```
print('Accuracy для "Логистической регресии"', accuracy_score(Y_test, target1
print('Accuracy для "SVM"', accuracy_score(Y_test, target2))
print('Accuracy для "Дерева решений"', accuracy_score(Y_test, target3))
```

```
Accuracy для "Логистической регресии" 0.9206349206349206
Accuracy для "SVM" 0.7619047619047619
Accuracy для "Дерева решений" 0.9206349206349206
```