Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет «Радиотехнический»

Кафедра ИУ5 «Системы обработки информации и управления»

Лабораторная работа №4

по дисциплине «Технологии машинного обучения»

Выполнил:
студент группы РТ5-61Б
М.А. Ходосов

2022 г.

Задание лабораторной работы:

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 следующие модели:
    - одну из линейных моделей (линейную или полиномиальную регрессию при решении задачи регрессии, логистическую регрессию при решении задачи классификации);
    - SVM;
    - дерево решений.
5. Оцените качество моделей с помощью двух подходящих для задачи метрик. Сравните качество полученных моделей.
6. Постройте график, показывающий важность признаков в дереве решений.
7. Визуализируйте дерево решений или выведите правила дерева решений в текстовом виде.

Лабораторная работа 4

Линейные модели, SVM и деревья решений.

Цель лабораторной работы: изучение линейных моделей, SVM и деревьев решений.

Выберите набор данных (датасет) для решения задачи классификации или регрессии. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков. С использованием метода train_test_split разделите выборку на обучающую и тестовую.

Обучите следующие модели:

- одну из линейных моделей (линейную или полиномиальную регрессию при решении задачи регрессии, логистическую регрессию при решении задачи классификации);
- SVM;
- дерево решений.

Оцените качество моделей с помощью двух подходящих для задачи метрик. Сравните качество полученных моделей.

Постройте график, показывающий важность признаков в дереве решений.

Визуализируйте дерево решений или выведите правила дерева решений в текстовом виде.

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import f1_score, precision_score
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import GridSearchCV
import matplotlib.pyplot as plt

target_col='class'

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

In [2]:
```
data = pd.read_csv('../datasets/mushrooms.csv')
data.head()
```

Out[2]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	population	habitat
0	p	x	s	n	t	p	f	c	n	k	...	s	w	w	p	w	o	p	k	s	u
1	e	x	s	y	t	a	f	c	b	k	...	s	w	w	p	w	o	p	n	n	g
2	e	b	s	w	t	l	f	c	b	n	...	s	w	w	p	w	o	p	n	n	m
3	p	x	y	w	t	p	f	c	n	n	...	s	w	w	p	w	o	p	k	s	u
4	e	x	s	g	f	n	f	w	b	k	...	s	w	w	p	w	o	e	n	a	g

5 rows × 23 columns

In [3]:
```
data.shape
```

Out[3]:
```
(8124, 23)
```

Удаляем пустые значения и кодируем категориальные признаки

In [4]:
```
data = data.dropna(axis=1, how='any')
data.head()
```

Out[4]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	population	habitat
0	p	x	s	n	t	p	f	c	n	k	...	s	w	w	p	w	o	p	k	s	u
1	e	x	s	y	t	a	f	c	b	k	...	s	w	w	p	w	o	p	n	n	g
2	e	b	s	w	t	l	f	c	b	n	...	s	w	w	p	w	o	p	n	n	m
3	p	x	y	w	t	p	f	c	n	n	...	s	w	w	p	w	o	p	k	s	u
4	e	x	s	g	f	n	f	w	b	k	...	s	w	w	p	w	o	e	n	a	g

5 rows × 23 columns

In [5]:
```
for col in data.columns:
    null_count = data[data[col].isnull()].shape[0]
    if null_count == 0:
        column_type = data[col].dtype
        print('{} - {} - {}'.format(col, column_type, null_count))
```

```
class - object - 0
cap-shape - object - 0
cap-surface - object - 0
cap-color - object - 0
bruises - object - 0
odor - object - 0
gill-attachment - object - 0
gill-spacing - object - 0
gill-size - object - 0
gill-color - object - 0
stalk-shape - object - 0
stalk-root - object - 0
stalk-surface-above-ring - object - 0
stalk-surface-below-ring - object - 0
stalk-color-above-ring - object - 0
stalk-color-below-ring - object - 0
veil-type - object - 0
veil-color - object - 0
ring-number - object - 0
ring-type - object - 0
spore-print-color - object - 0
population - object - 0
habitat - object - 0
```

Категориальные признаки

In [6]:
```
le = LabelEncoder()
for col in data.columns:
    column_type = data[col].dtype
    if column_type == 'object':
        data[col] = le.fit_transform(data[col]);
        print(col)
```

```
class
cap-shape
cap-surface
cap-color
bruises
odor
gill-attachment
gill-spacing
gill-size
gill-color
stalk-shape
stalk-root
stalk-surface-above-ring
stalk-surface-below-ring
stalk-color-above-ring
stalk-color-below-ring
veil-type
veil-color
ring-number
ring-type
spore-print-color
population
habitat
```

Разделение выборки на обучающую и тестовую

In [7]:
```
X = data.drop(target_col, axis=1)
Y = data[target_col]
```

In [8]:
```
X
```

Out[8]:

	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	population	habitat
0	5	2	4	1	6	1	0	1	4	0	...	2	7	7	0	2	1	4	2	3	5
1	5	2	9	1	0	1	0	0	4	0	...	2	7	7	0	2	1	4	3	2	1
2	0	2	8	1	3	1	0	0	5	0	...	2	7	7	0	2	1	4	3	2	3
3	5	3	8	1	6	1	0	1	5	0	...	2	7	7	0	2	1	4	2	3	5
4	5	2	3	0	5	1	1	0	4	1	...	2	7	7	0	2	1	0	3	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8119	3	2	4	0	5	0	0	0	11	0	...	2	5	5	0	1	1	4	0	1	2
8120	5	2	4	0	5	0	0	0	11	0	...	2	5	5	0	0	1	4	0	4	2
8121	2	2	4	0	5	0	0	0	5	0	...	2	5	5	0	1	1	4	0	1	2
8122	3	3	4	0	8	1	0	1	0	1	...	1	7	7	0	2	1	0	7	4	2
8123	5	2	4	0	5	0	0	0	11	0	...	2	5	5	0	1	1	4	4	1	2

8124 rows × 22 columns

In [9]:
```
Y
```

Out[9]:
```
0       1
1       0
2       0
3       1
4       0
       ..
8119    0
8120    0
8121    0
8122    1
8123    0
Name: class, Length: 8124, dtype: int32
```

In [10]:
```
pd.DataFrame(X, columns=X.columns).describe()
```

Out[10]:

	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	population	habitat
count	8124.000000	8124.000000	8124.000000	8124.000000	8124.000000	8124.000000	8124.000000	8124.000000	8124.000000	8124.000000	...	8124.000000	8124.000000	8124.000000	8124.0	8124.000000	8124.000000	8124.000000	8124.000000	8124.000000	8124.000000
mean	3.348104	1.827671	4.504677	0.415559	4.144756	0.974151	0.161497	0.309207	4.810684	0.567208	...	1.603644	5.816347	5.794682	0.0	1.965534	1.069424	2.291974	3.596750	3.644018	1.508616
std	1.604329	1.229873	2.545821	0.492848	2.103729	0.158695	0.368011	0.462195	3.540359	0.495493	...	0.675974	1.901747	1.907291	0.0	0.242669	0.271064	1.801672	2.382663	1.252082	1.719975
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	2.000000	0.000000	3.000000	0.000000	2.000000	1.000000	0.000000	0.000000	2.000000	0.000000	...	1.000000	6.000000	6.000000	0.0	2.000000	1.000000	0.000000	2.000000	3.000000	0.000000
50%	3.000000	2.000000	4.000000	0.000000	5.000000	1.000000	0.000000	0.000000	5.000000	1.000000	...	2.000000	7.000000	7.000000	0.0	2.000000	1.000000	2.000000	3.000000	4.000000	1.000000
75%	5.000000	3.000000	8.000000	1.000000	5.000000	1.000000	0.000000	1.000000	7.000000	1.000000	...	2.000000	7.000000	7.000000	0.0	2.000000	1.000000	4.000000	7.000000	4.000000	2.000000
max	5.000000	3.000000	9.000000	1.000000	8.000000	1.000000	1.000000	1.000000	11.000000	1.000000	...	3.000000	8.000000	8.000000	0.0	3.000000	2.000000	4.000000	8.000000	5.000000	6.000000

8 rows × 22 columns

Разделим выборку на обучающую и тестовую:

In [11]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=1)
print('{}, {}'.format(X_train.shape, X_test.shape))
print('{}, {}'.format(Y_train.shape, Y_test.shape))
```

```
(6093, 22), (2031, 22)
(6093,), (2031,)
```

Обучение моделей

Линейная модель

In [12]:
```
SGD = SGDClassifier(max_iter=10000)
SGD.fit(X_train, Y_train)
```

Out[12]:
```
SGDClassifier(max_iter=10000)
```

In [13]:
```
from sklearn.metrics import median_absolute_error, r2_score, precision_score
f1_score(Y_test, SGD.predict(X_test), average='micro')
precision_score(Y_test, SGD.predict(X_test), average='micro')
```

Out[13]:
```
0.9187592319054653
```

SVM

In [14]:
```
SVC = SVC(kernel='rbf')
SVC.fit(X_train, Y_train)
```

Out[14]:
```
SVC()
```

In [15]:
```
f1_score(Y_test, SVC.predict(X_test), average='micro')
precision_score(Y_test, SVC.predict(X_test), average='micro')
```

Out[15]:
```
0.9862136878385032
```

Дерево решений

In [16]:
```
DT = DecisionTreeClassifier(random_state=1)
DT.fit(X_train, Y_train)
```

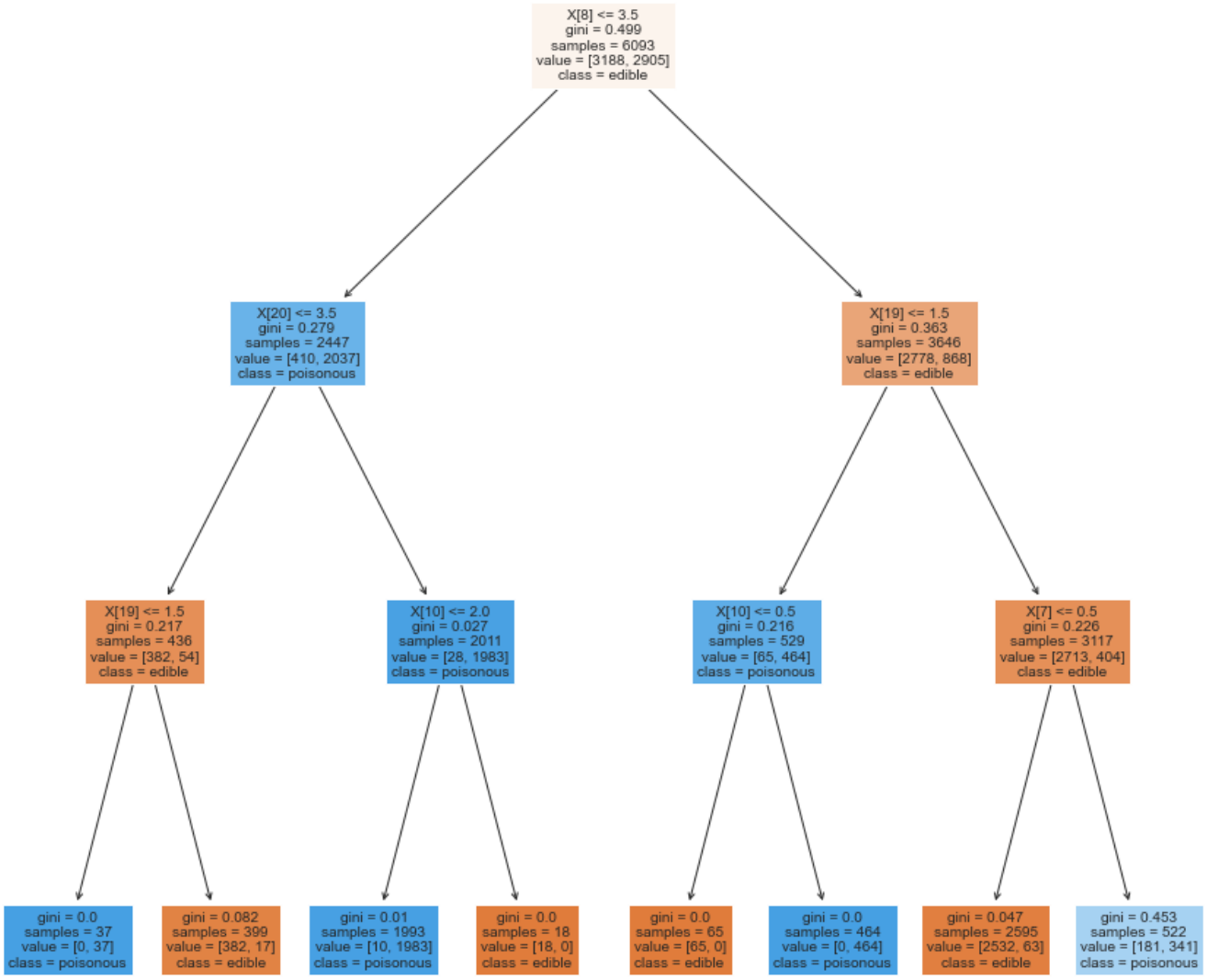
Out[16]:
```
DecisionTreeClassifier(random_state=1)
```

In [17]:
```
print(f1_score(Y_test, DT.predict(X_test), average='micro'))
precision_score(Y_test, DT.predict(X_test), average='micro')
```

Out[17]:
```
1.0
1.0
```

Можно сделать вывод, что дерево решений дает лучший результат

In [18]:
```
from sklearn import tree
fig, ax = plt.subplots(figsize=(15, 15))
clf = DecisionTreeClassifier(max_depth = 3,
                             random_state = 0)
clf.fit(X_train, Y_train)
cn=['edible', 'poisonous']
tree.plot_tree(clf, fontsize=10, class_names=cn, filled=True)
plt.show()
```


X[8] <= 3.5
gini = 0.499
samples = 6093
value = [3188, 2905]
class = edible
X[20] <= 3.5
gini = 0.279
samples = 2447
value = [410, 2037]
class = poisonous
X[19] <= 1.5
gini = 0.363
samples = 3646
value = [2778, 868]
class = edible
X[19] <= 1.5
gini = 0.217
samples = 436
value = [382, 54]
class = edible
X[10] <= 2.0
gini = 0.027
samples = 2011
value = [28, 1983]
class = poisonous
X[10] <= 0.5
gini = 0.216
samples = 529
value = [65, 464]
class = poisonous
X[7] <= 0.5
gini = 0.226
samples = 3117
value = [2713, 404]
class = edible
gini = 0.0
samples = 37
value = [0, 37]
class = poisonous
gini = 0.082
samples = 399
value = [382, 17]
class = edible
gini = 0.01
samples = 1993
value = [10, 1983]
class = poisonous
gini = 0.0
samples = 18
value = [18, 0]
class = edible
gini = 0.0
samples = 65
value = [65, 0]
class = edible
gini = 0.0
samples = 464
value = [0, 464]
class = poisonous
gini = 0.047
samples = 2595
value = [2532, 63]
class = edible
gini = 0.453
samples = 522
value = [181, 341]
class = poisonous