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 [4]:
```
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 [5]:
```
data = pd.read_csv('./mushrooms.csv')
data.head()
```

Out[5]:

class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stal surfac belo ri

0	p	x	s	n	t	p	f	c	n	k	...
1	e	x	s	y	t	a	f	c	b	k	...
2	e	b	s	w	t	l	f	c	b	n	...
3	p	x	y	w	t	p	f	c	n	n	...
4	e	x	s	g	f	n	f	w	b	k	...

5 rows × 23 columns

In [6]:
```
data.shape
```

Out[6]:
```
(8124, 23)
```

Удаляем пустые значения и кодируем категориальные признаки

In [7]:
```
data = data.dropna(axis=1, how='any')
data.head()
```

Out[7]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stal surfac belo ri
0	p	x	s	n	t	p	f	c	n	k	...	
1	e	x	s	y	t	a	f	c	b	k	...	
2	e	b	s	w	t	l	f	c	b	n	...	
3	p	x	y	w	t	p	f	c	n	n	...	
4	e	x	s	g	f	n	f	w	b	k	...	

5 rows × 23 columns

In [8]:
```
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 [9]:

```
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 [10]:

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

In [11]:

```
X
```

Out[11]:

	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	...	su
0	5	2	4	1	6	1	0	1	4	0	...	
1	5	2	9	1	0	1	0	0	4	0	...	
2	0	2	8	1	3	1	0	0	5	0	...	

3	5	3	8	1	6	1	0	1	5	0	...
4	5	2	3	0	5	1	1	0	4	1	...
...	...	...	...	...	...	...	...	...	...	...	...
8119	3	2	4	0	5	0	0	0	11	0	...
8120	5	2	4	0	5	0	0	0	11	0	...
8121	2	2	4	0	5	0	0	0	5	0	...
8122	3	3	4	0	8	1	0	1	0	1	...
8123	5	2	4	0	5	0	0	0	11	0	...

8124 rows × 22 columns

In [12]:
```
Y
```

Out[12]:
```
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 [13]:
```
pd.DataFrame(X, columns=X.columns).describe()
```

Out[13]:

	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-
count	8124.000000	8124.000000	8124.000000	8124.000000	8124.000000	8124.000000	8124
mean	3.348104	1.827671	4.504677	0.415559	4.144756	0.974151	0
std	1.604329	1.229873	2.545821	0.492848	2.103729	0.158695	0
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0
25%	2.000000	0.000000	3.000000	0.000000	2.000000	1.000000	0
50%	3.000000	2.000000	4.000000	0.000000	5.000000	1.000000	0
75%	5.000000	3.000000	8.000000	1.000000	5.000000	1.000000	0
max	5.000000	3.000000	9.000000	1.000000	8.000000	1.000000	1

8 rows × 22 columns

Разделим выборку на обучающую и тестовую:

In [14]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25
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 [15]:
```
SGD = SGDClassifier(max_iter=10000)
SGD.fit(X_train, Y_train)
```

Out[15]:
```
SGDClassifier(max_iter=10000)
```

In [16]:
```
from sklearn.metrics import median_absolute_error, r2_score, presicion_s
f1_score(Y_test, SGD.predict(X_test), average='micro')
precision_score(Y_test, SGD.predict(X_test), average='micro')
```

Out[16]:
```
0.9512555391432792
```

SVM

In [17]:
```
SVC = SVC(kernel='rbf')
SVC.fit(X_train, Y_train)
```

Out[17]:
```
SVC()
```

In [18]:
```
f1_score(Y_test, SVC.predict(X_test), average='micro')
precision_score(Y_test, SVC.predict(X_test), average='micro')
```

Out[18]:
```
0.9862136878385032
```

Дерево решений

In [19]:
```
DT = DecisionTreeClassifier(random_state=1)
DT.fit(X_train, Y_train)
```

Out[19]:
```
DecisionTreeClassifier(random_state=1)
```

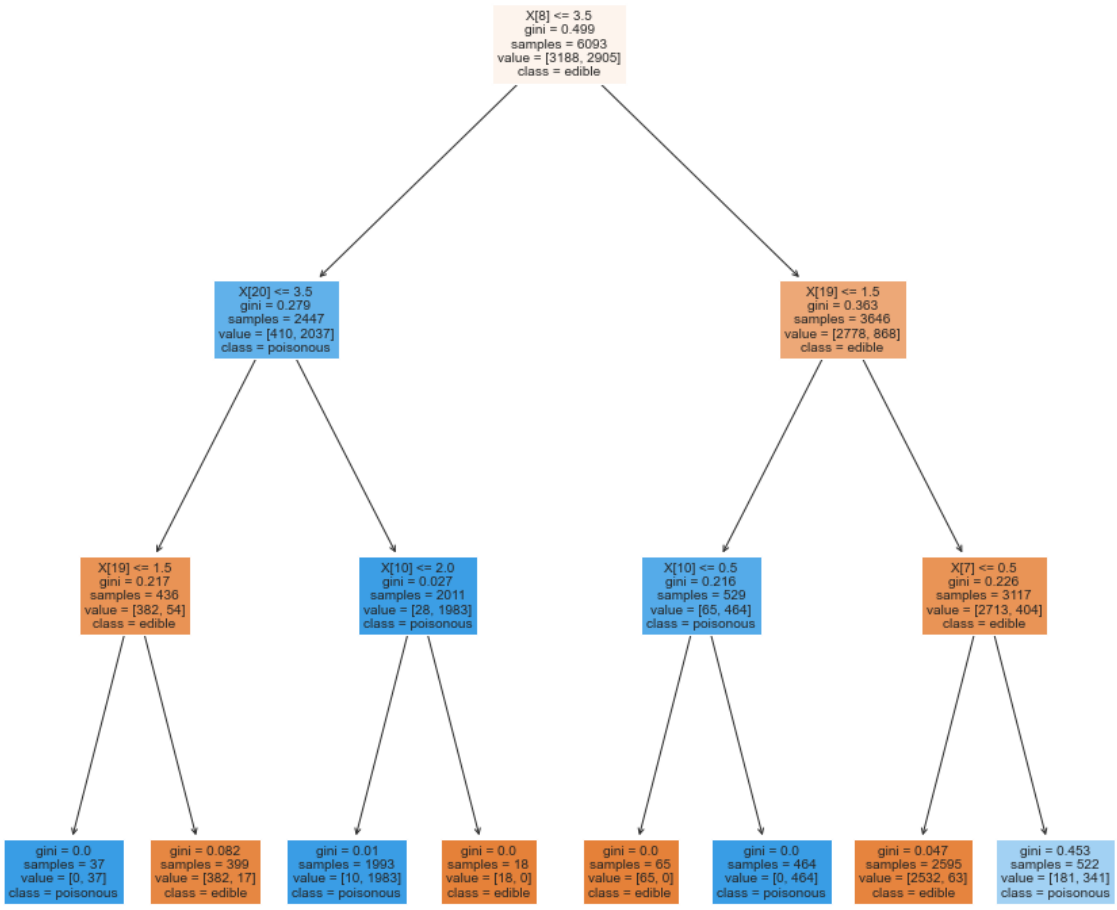
In [21]:
```
print(f1_score(Y_test, DT.predict(X_test), average='micro'))
precision_score(Y_test, DT.predict(X_test), average='micro')
```

Out[21]:
```
1.0
1.0
```

Можно сделать вывод, что дерево решений дает лучший результат

In [22]:
```
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()
```



In [ ]: