Лабораторная работа

Ансамбли моделей машинного обучения.

Цель лабораторной работы: изучение ансамблей моделей машинного обучения.

Задание: Выберите набор данных (датасет) для решения задачи классификации или регресии.

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 ансамблевые модели:

одну из моделей группы бэггинга (бэггинг или случайный лес или сверхслучайные деревья); одну из моделей группы бустинга; одну из моделей группы стекинга. (+1 балл на экзамене) Дополнительно к указанным моделям обучите еще две модели:

Модель многослойного персептрона. По желанию, вместо библиотеки scikit-learn возможно использование библиотек TensorFlow, PyTorch или других аналогичных библиотек. Модель МГУА с использованием библиотеки - https://github.com/kvoyager/GmdhPy (https://github.com/kvoyager/GmdhPy) (или аналогичных библиотек). Найдите такие параметры запуска модели, при которых она будет по крайней мере не хуже, чем одна из предыдущих ансамблевых моделей. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.

Ввод [1]:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder


# скроем предупреждения о возможных ошибках для лучшей читаемости
import warnings
warnings.filterwarnings('ignore')

target_col = 'class'
```

Ввод [2]:

```
data = pd.read_csv('./mushrooms.csv')
data
```

Out[2]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk surface below rin
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	...	
...	...	...	...	...	...	...	...	...	...	...	...	.
8119	e	k	s	n	f	n	a	c	b	y	...	
8120	e	x	s	n	f	n	a	c	b	y	...	
8121	e	f	s	n	f	n	a	c	b	n	...	
8122	p	k	y	n	f	y	f	c	n	b	...	
8123	e	x	s	n	f	n	a	c	b	y	...	

8124 rows × 23 columns

Предварительная обработка

Удаляем столбцы с пустыми значениями:

Ввод [3]:

```
data = data.dropna(axis=1, how='any')
data
```

Out[3]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk surface below rin
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	...	
...	...	...	...	...	...	...	...	...	...	...	...	.
8119	e	k	s	n	f	n	a	c	b	y	...	
8120	e	x	s	n	f	n	a	c	b	y	...	
8121	e	f	s	n	f	n	a	c	b	n	...	
8122	p	k	y	n	f	y	f	c	n	b	...	
8123	e	x	s	n	f	n	a	c	b	y	...	

8124 rows × 23 columns

Ввод [4]:

```
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:

Ввод [5]:

```
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

Ввод [6]:

```
from sklearn.model_selection import train_test_split

data_x = data.loc[:, data.columns != target_col]
data_y = data[target_col]

train_x, test_x, train_y, test_y = train_test_split(data_x, data_y, test_size=0.3, r
```

Ввод [7]:

```
train_x.shape
```

Out[7]:

```
(5686, 22)
```

Ввод [8]:

```
test_x.shape
```

Out[8]:

```
(2438, 22)
```

Ввод [9]:

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import median_absolute_error, r2_score

def test_model(model):
    print('mean_absolute_error: {}'.format(round(mean_absolute_error(test_y, model.p
    print('median_absolute_error: {}'.format(round(median_absolute_error(test_y, mod
    print('r2_score: {}'.format(round(r2_score(test_y, model.predict(test_x)), 2)))
```

Обучение моделей

Случайный лес

Ввод [10]:

```
from sklearn.ensemble import RandomForestRegressor

ran_80 = RandomForestRegressor(n_estimators=80)
ran_80.fit(train_x, train_y)
```

Out[10]:

```
RandomForestRegressor(n_estimators=80)
```

Ввод [11]:

```
test_model(ran_80)
```

```
mean_absolute_error: 0.0
median_absolute_error: 0.0
r2_score: 1.0
```

Ввод [12]:

```
param_range = np.arange(50, 170, 10)
tuned_parameters = [{'n_estimators': param_range}]
tuned_parameters
```

Out[12]:

```
[{'n_estimators': array([ 50,  60,  70,  80,  90, 100, 110, 120, 130,
140, 150, 160])}]
```

Ввод [13]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import ShuffleSplit

gs = GridSearchCV(RandomForestRegressor(), tuned_parameters,
                  cv=ShuffleSplit(n_splits=10), scoring="r2",
                  return_train_score=True, n_jobs=-1)
gs.fit(data_x, data_y)
```

Out[13]:

```
GridSearchCV(cv=ShuffleSplit(n_splits=10, random_state=None, test_size
=None, train_size=None),
             estimator=RandomForestRegressor(), n_jobs=-1,
             param_grid=[{'n_estimators': array([ 50,  60,  70,  80,
90, 100, 110, 120, 130, 140, 150, 160])}],
             return_train_score=True, scoring='r2')
```

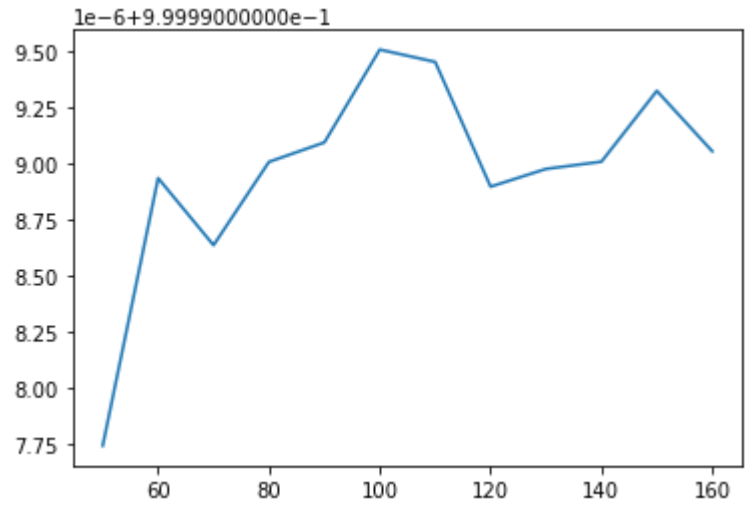
Ввод [14]:

```
reg = gs.best_estimator_
```

Ввод [15]:

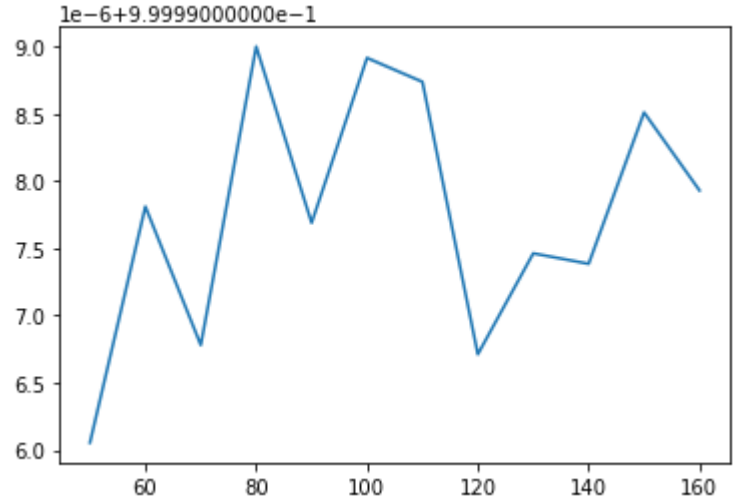
```
import matplotlib.pyplot as plt

plt.plot(param_range, gs.cv_results_["mean_train_score"]);
```



Ввод [16]:

```
plt.plot(param_range, gs.cv_results_["mean_test_score"]);
```



Ввод [17]:

```
reg.fit(train_x, train_y)
test_model(reg)
```

```
mean_absolute_error: 0.0
median_absolute_error: 0.0
r2_score: 1.0
```

Градиентный бустинг

Ввод [18]:

```
from sklearn.ensemble import GradientBoostingRegressor

gr_80 = GradientBoostingRegressor(n_estimators=80)
gr_80.fit(train_x, train_y)
```

Out[18]:

```
GradientBoostingRegressor(n_estimators=80)
```

Ввод [19]:

```
test_model(gr_80)
```

```
mean_absolute_error: 0.02
median_absolute_error: 0.0
r2_score: 0.99
```

Ввод [20]:

```
gs = GridSearchCV(GradientBoostingRegressor(), tuned_parameters,
                  cv=ShuffleSplit(n_splits=10), scoring="r2",
                  return_train_score=True, n_jobs=-1)
gs.fit(data_x, data_y)
```

Out[20]:

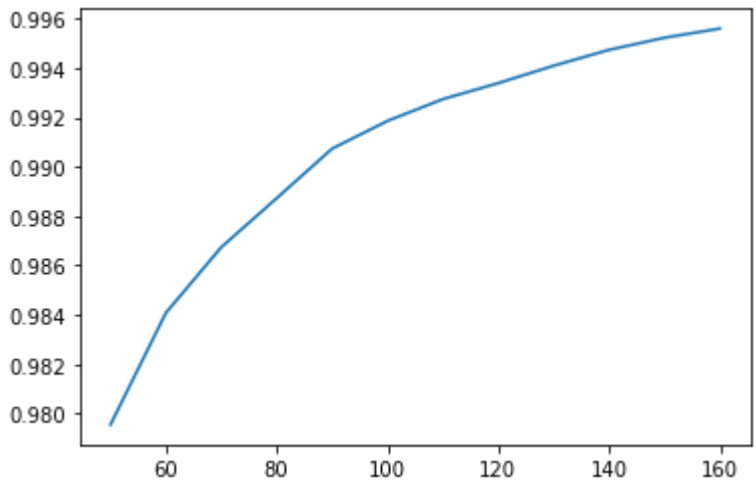
```
GridSearchCV(cv=ShuffleSplit(n_splits=10, random_state=None, test_size
=None, train_size=None),
             estimator=GradientBoostingRegressor(), n_jobs=-1,
             param_grid=[{'n_estimators': array([ 50,  60,  70,  80,
90, 100, 110, 120, 130, 140, 150, 160])}],
             return_train_score=True, scoring='r2')
```

Ввод [21]:

```
reg = gs.best_estimator_
```

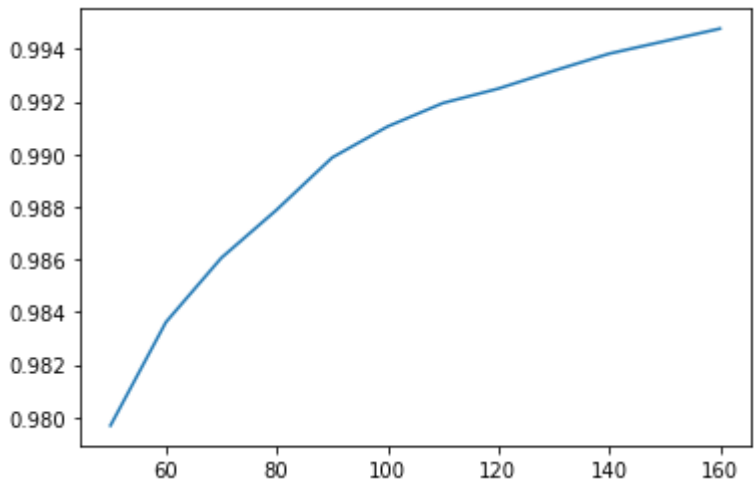
Ввод [22]:

```
plt.plot(param_range, gs.cv_results_["mean_train_score"]);
```



Ввод [23]:

```
plt.plot(param_range, gs.cv_results_["mean_test_score"]);
```



Ввод [24]:

```
reg.fit(train_x, train_y)
test_model(reg)
```

```
mean_absolute_error: 0.01
median_absolute_error: 0.0
r2_score: 0.99
```

Ввод [ ]:

```

```