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 **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

О Т Ч Е Т
по лабораторной работе № 5

Дисциплина: Технологии машинного обучения
Тема: «Ансамбли моделей машинного обучения»

Студент	ИУ5Ц-82Б		А.Н. Свинцов
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			Ю.Е. Гапанюк
		(Подпись, дата)	(И.О. Фамилия)
Преподаватель			А.Н. Нардид
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2023

Лабораторная работа №5

Ансамбли моделей машинного обучения

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

import seaborn as sns
import matplotlib.pyplot as plt


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

target_col = 'class'

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

```
     class cap-shape cap-surface cap-color bruises odor gill-
attachment  \
0        p         x           s         n       t    p
f
1        e         x           s         y       t    a
f
2        e         b           s         w       t    l
f
3        p         x           y         w       t    p
f
4        e         x           s         g       f    n
f
...    ...       ...         ...       ...     ...  ...             ..
.
8119     e         k           s         n       f    n
a
8120     e         x           s         n       f    n
a
8121     e         f           s         n       f    n
a
8122     p         k           y         n       f    y
f
8123     e         x           s         n       f    n
a

     gill-spacing gill-size gill-color  ... stalk-surface-below-
ring  \
0               c         n          k  ...                        s
```

```
1               c         b          k  ...                         s

2               c         b          n  ...                         s

3               c         n          n  ...                         s

4               w         b          k  ...                         s

...           ...       ...        ...  ...                       ...

8119            c         b          y  ...                         s

8120            c         b          y  ...                         s

8121            c         b          n  ...                         s

8122            c         n          b  ...                         k

8123            c         b          y  ...                         s


     stalk-color-above-ring stalk-color-below-ring veil-type veil-
color  \
0                         w                       w         p
w
1                         w                       w         p
w
2                         w                       w         p
w
3                         w                       w         p
w
4                         w                       w         p
w
...                     ...                     ...       ...        ..
.
8119                      o                       o         p
o
8120                      o                       o         p
n
8121                      o                       o         p
o
8122                      w                       w         p
w
8123                      o                       o         p
o

     ring-number ring-type spore-print-color population habitat
0              o         p                 k          s       u
```

```
1              o         p                    n           n       g
2              o         p                    n           n       m
3              o         p                    k           s       u
4              o         e                    n           a       g
...          ...       ...                  ...         ...     ...
8119           o         p                    b           c       l
8120           o         p                    b           v       l
8121           o         p                    b           c       l
8122           o         e                    w           v       l
8123           o         p                    o           c       l

[8124 rows x 23 columns]
```

Предварительная обработка

```
#Проверка на пропуски
data.isnull().sum()
```

```
class                       0
cap-shape                   0
cap-surface                 0
cap-color                   0
bruises                     0
odor                        0
gill-attachment             0
gill-spacing                0
gill-size                   0
gill-color                  0
stalk-shape                 0
stalk-root                  0
stalk-surface-above-ring    0
stalk-surface-below-ring    0
stalk-color-above-ring      0
stalk-color-below-ring      0
veil-type                   0
veil-color                  0
ring-number                 0
ring-type                   0
spore-print-color           0
population                  0
habitat                     0
dtype: int64
```

Пропусков нет

Категориальные признаки

```
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
```

Обучающая и тестовая выборка

```
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, random_state=1)

train_x.shape
```

```
(5686, 22)
```

```
test_x.shape
```

```
(2438, 22)
```

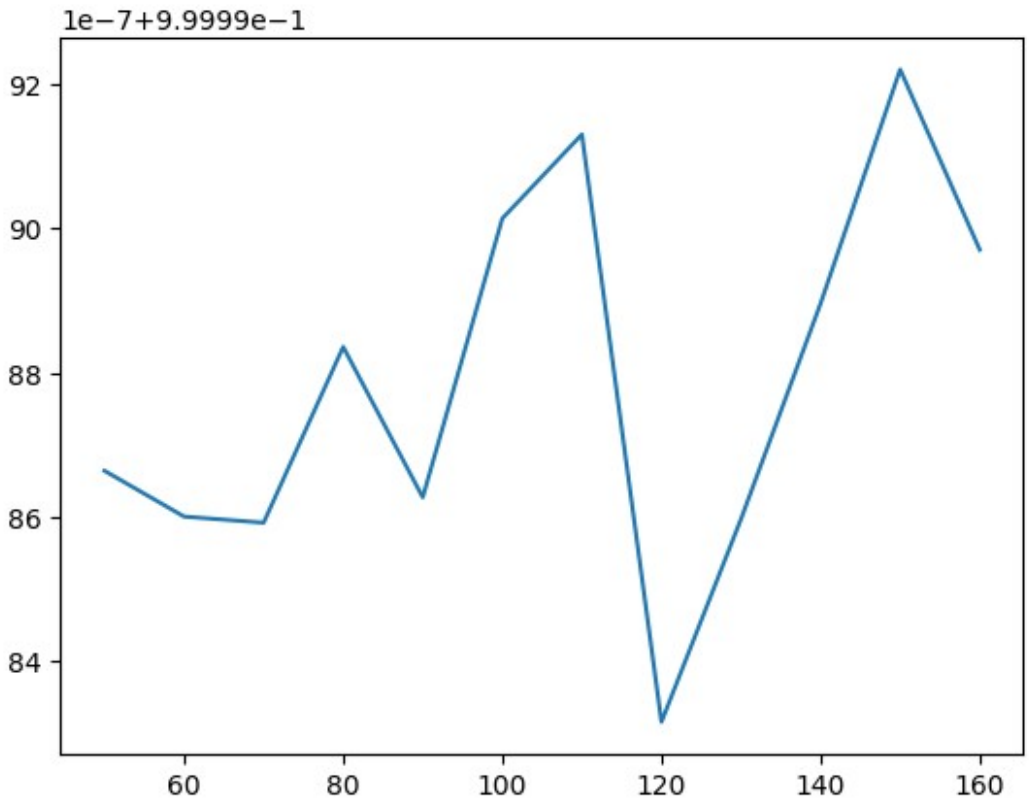
```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import median_absolute_error, r2_score,
precision_score

def test_model(model):
```

```
    print('precision: {}'.format(round(precision_score(test_y,
model.predict(test_x)), 2)))
```

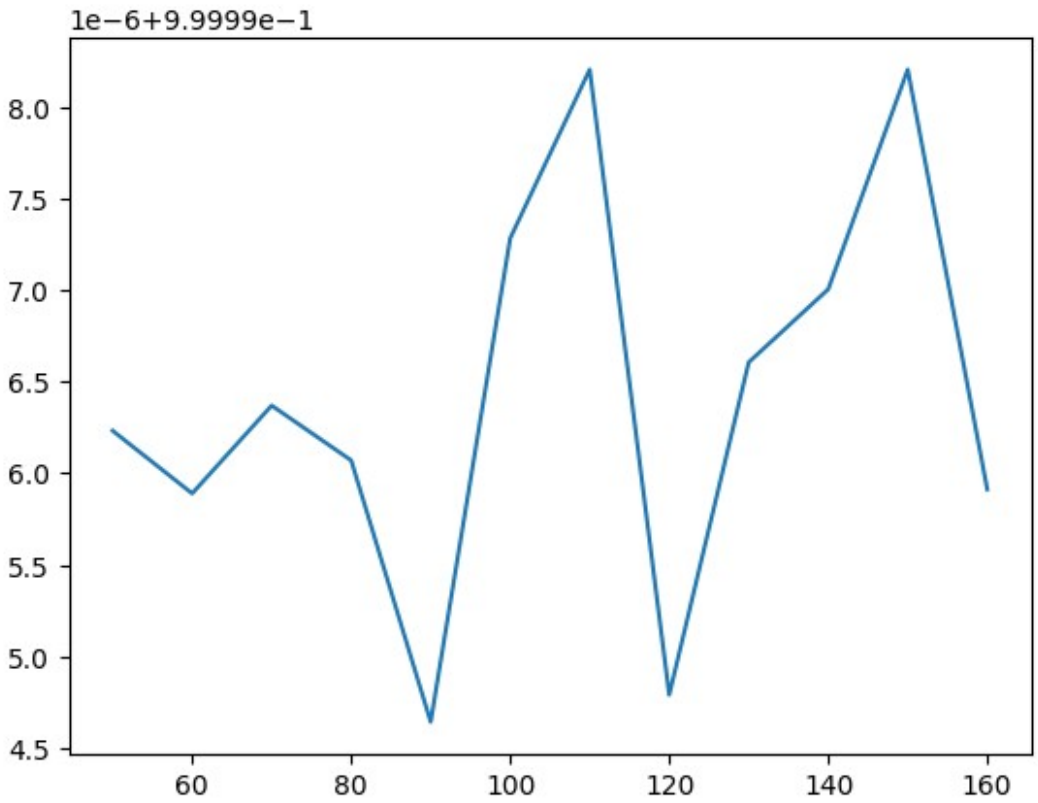
Обучение моделей

Случайный лес

```
from sklearn.ensemble import RandomForestRegressor

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

RandomForestRegressor(n_estimators=80)

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

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

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)

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')

reg = gs.best_estimator_

import matplotlib.pyplot as plt

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




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




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

```
RandomForestRegressor(n_estimators=150)
```

Градиентный бустинг

```
from sklearn.ensemble import GradientBoostingRegressor

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

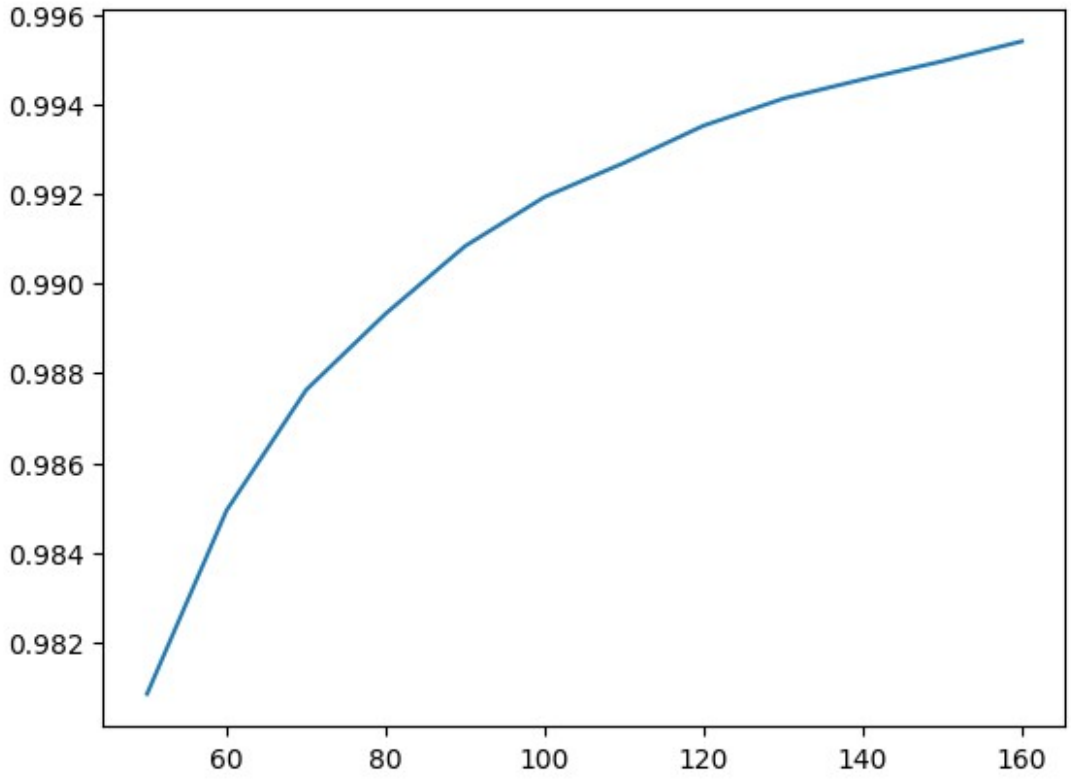
```
GradientBoostingRegressor(n_estimators=80)
```

```
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)
```

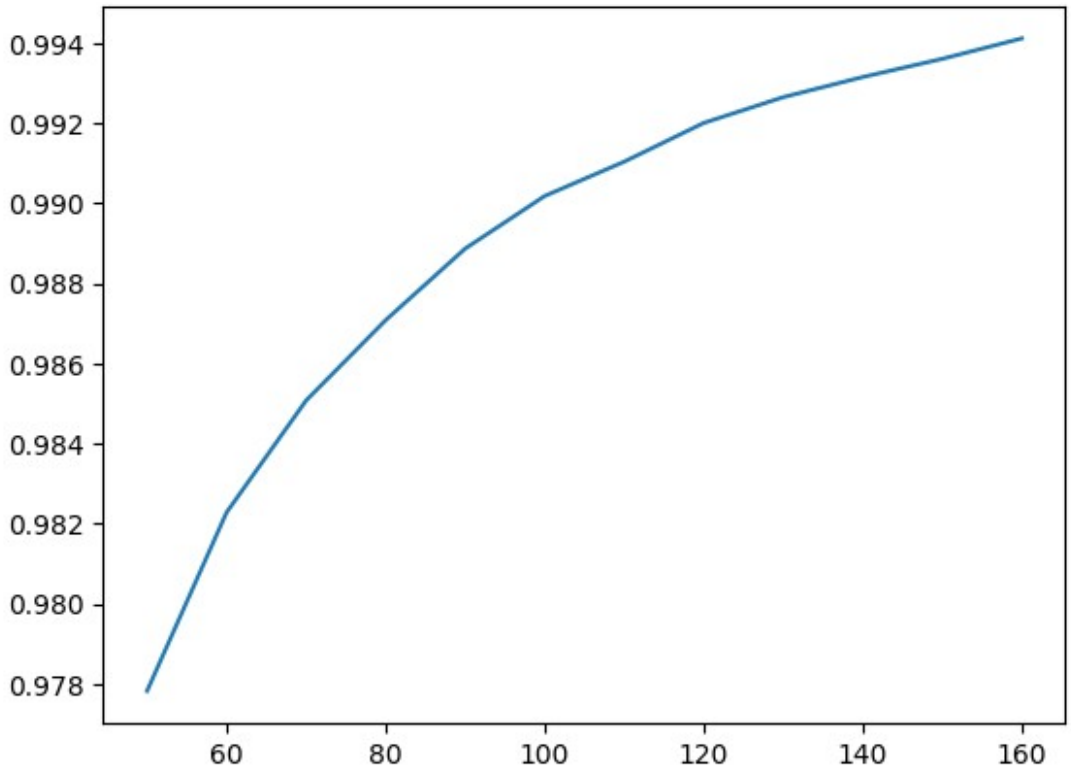
```
GridSearchCV(cv=ShuffleSplit(n_splits=10, random_state=None,
test_size=None, train_size=None),
             estimator=GradientBoostingRegressor(), n_jobs=-1,
             param_grid=[{'n_estimators': array([ 50,  60,  70,  80,
90, 100, 110, 120, 130, 140, 150, 160])}],
             return_train_score=True,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

```
reg = gs.best_estimator_
```

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




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




```
reg.fit(train_x, train_y)

GradientBoostingRegressor(n_estimators=160)
```

Стекинг

```
from sklearn.ensemble import RandomForestClassifier,
StackingClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression

base_learners = [
                 ('rf_1', RandomForestClassifier(n_estimators=10,
random_state=42)),
                 ('rf_2', GradientBoostingClassifier(n_estimators=80))

                ]

clf = StackingClassifier(estimators=base_learners,
final_estimator=LogisticRegression())

clf.fit(train_x, train_y)

StackingClassifier(estimators=[('rf_1',

RandomForestClassifier(n_estimators=10,

random_state=42)),
```

```
                                ('rf_2',

GradientBoostingClassifier(n_estimators=80))],
                   final_estimator=LogisticRegression())
```