Задание: 1) Выберите набор данных (датасет) для решения задачи классификации или регресии. 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 две ансамблевые модели. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей. 5) Произведите для каждой модели подбор значений одного гиперпараметра. В зависимости от используемой библиотеки можно применять функцию GridSearchCV, использовать перебор параметров в цикле, или использовать другие методы. 6) Повторите пункт 4 для найденных оптимальных значений гиперпараметров. Сравните качество полученных моделей с качеством моделей, полученных в пункте 4.

In [8]:

```
import numpy as np #библиотека для работы с многомерными массивами данных
 и математическими операциями над ними
import pandas as pd #библиотека для анализа и обработки данных
from scipy import stats
from sklearn.datasets import load_iris #берём датасет
import matplotlib.pyplot as plt #простое рисование графиков
import seaborn as sns #удобные дефолтные настройки графиков из matpotlib
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import *
from sklearn.metrics import *
from sklearn.linear_model import LinearRegression # для линейной модели
from sklearn.svm import SVR # для SVM модели
from sklearn.tree import DecisionTreeRegressor # для дерева решений

from sklearn.ensemble import BaggingRegressor, RandomForestRegressor

%matplotlib inline
#для сохранения в ноутбуке вывода моих графиков
```

In [9]:

```
df = load_iris()
df = pd.DataFrame(data = np.c_[df['data'], df['target']], columns = df['fe
ature_names'] + ['target'])
df.head()
```

Out[9]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0

In [10]:

```
df.loc[:, df.columns!='target'] = df.loc[:, df.columns!='target'].apply(la
mbda x: x/x.max(), axis=0)
```

In [11]:

```
x_train, x_test, y_train, y_test = train_test_split(df.loc[:, df.columns!=
'target'],
                                                    df['target'],
                                                    test_size= 0.33)
```

In [12]:

```
def print_stat(model):
    print(mean_absolute_error(model.predict(x_test), y_test))
    print(r2_score(model.predict(x_test), y_test))
    print(median_absolute_error(model.predict(x_test), y_test))
```

In [13]:

```
base = LinearRegression()
BR = BaggingRegressor(base_estimator = base)
BR.fit(x_train, y_train)
```

Out[13]:

```
BaggingRegressor(base_estimator=LinearRegression(copy_X=Tr
ue, fit_intercept=True, n_jobs=1, normalize=False),
         bootstrap=True, bootstrap_features=False, max_fea
tures=1.0,
         max_samples=1.0, n_estimators=10, n_jobs=1, oob_s
core=False,
         random_state=None, verbose=0, warm_start=False)
```

In [14]:

```
print_stat(BR)
```

```
0.18002362402763242
0.9096246818373858
0.11587999684220415
```

In [15]:

```
RFR = RandomForestRegressor(max_depth=3, random_state=0,
                            n_estimators=100)
RFR.fit(x_train, y_train)
```

Out[15]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max
_depth=3,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=N
one,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=100,
n_jobs=1,
           oob_score=False, random_state=0, verbose=0, war
m_start=False)
```

In [16]:

```
print_stat(RFR)
```

```
0.054377203547699995
0.9364074105131873
0.0009150052172058887
```

In [17]:

```
base = LinearRegression()
BR = BaggingRegressor(base_estimator = base)
BR_GV = GridSearchCV(BR, {'n_jobs':range(1,10)}, cv=3).fit(x_train, y_trai
n).best_estimator_
```

In [18]:

```
print_stat(BR_GV)
```

```
0.18059613943213212
0.9092301257908989
0.11186795666308824
```

In [19]:

```
RFR = RandomForestRegressor(random_state=0,
                            n_estimators=100)
RFR_GV = GridSearchCV(RFR, {'max_depth':range(1,10)}, cv=3).fit(x_train, y
_train).best_estimator_
```

In [20]:

```
print_stat(RFR_GV)
```

```
0.051047142857142874
0.9391052844362164
0.0
```