Ансамбли моделей машинного обучения

In [23]:

```
import numpy as np
import pandas as pd
from typing import Dict, Tuple
from scipy import stats
from IPython.display import Image
from sklearn.datasets import load_iris, load_boston
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.ensemble import ExtraTreesClassifier, ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingClassifier, GradientBoostingRegressor
from sklearn.ensemble import BaggingClassifier
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_error, median_absolute_error, r2_score
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, LinearSVR
from sklearn.linear_model import LinearRegression
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

In [24]:

```
data=pd.read_csv('heart.csv', sep=",")
```

In [25]:

```
data.head()
```

Out[25]:

	age	sex	cp	trestbps	chol	restecg	thalach	slope	ca	thal	target
0	63	1	3	145	233	0	150	0	0	1	1
1	37	1	2	130	250	1	187	0	0	2	1
2	41	0	1	130	204	0	172	2	0	2	1
3	56	1	1	120	236	1	178	2	0	2	1
4	57	0	0	120	354	1	163	2	0	2	1

In [26]:

```
X_train, X_test, y_train, y_test = train_test_split(
    data, data['target'], test_size= 0.2, random_state= 1)
```

Обучение двух ансамблевых моделей

In [27]:

```
#случайный лес (n_estimators - число дерневьев)

randomforest = RandomForestClassifier(n_estimators=5, max_depth=1, random_state=0).fit(X_train, y_train)
```

In [28]:

```
target_randomforest = randomforest.predict(X_test)
```

In [29]:

In [29]:

```
accuracy_score(y_test, target_randomforest), \
precision_score(y_test, target_randomforest),  \
recall_score(y_test, target_randomforest)
```

Out[29]:

```
(0.8852459016393442, 0.8157894736842105, 1.0)
```

In [30]:

```
#градиентный бустинг
gradient_boosting = GradientBoostingClassifier(n_estimators=5, max_depth=1, learning_rate=0.01).fit(X_train, y_train)
```

In [31]:

```
target_gradient_boosting = gradient_boosting.predict(X_test)
```

In [32]:

```
accuracy_score(y_test, target_gradient_boosting), \
precision_score(y_test, target_gradient_boosting),  \
recall_score(y_test, target_gradient_boosting)
```

Out[32]:

```
(0.5081967213114754, 0.5081967213114754, 1.0)
```

Подбор гиперпараметра

In [33]:

```
parameters_random_forest = {'n_estimators':[1, 3, 5, 7, 10],
                            'max_depth':[1, 3, 5, 7, 10],
                            'random_state':[0, 2, 4, 6, 8, 10]}
best_random_forest = GridSearchCV(RandomForestClassifier(), parameters_random_forest, cv=3, scoring='accuracy')
best_random_forest.fit(X_train, y_train)
```

```
C:\Users\Dovlat\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py:841: DeprecationWarning: The default of the `iid` parameter will ch
ange from True to False in version 0.22 and will be removed in 0.24. This will change numeric results when test-set sizes are unequal.
  DeprecationWarning)
```

Out[33]:

```
GridSearchCV(cv=3, error_score='raise-deprecating',
       estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators='warn', n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False),
       fit_params=None, iid='warn', n_jobs=None,
       param_grid={'n_estimators': [1, 3, 5, 7, 10], 'max_depth': [1, 3, 5, 7, 10], 'random_state': [0, 2, 4, 6, 8, 10]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='accuracy', verbose=0)
```

In [34]:

```
parameters_gradient_boosting = {'n_estimators':[1, 3, 5, 7, 10],
                            'max_depth':[1, 3, 5, 7, 10],
                            'learning_rate':[0.001, 0.0025, 0.005, 0.0075, 0.01, 0.025]}
best_gradient_boosting = GridSearchCV(GradientBoostingClassifier(), parameters_gradient_boosting, cv=3, scoring='accuracy')
best_gradient_boosting.fit(X_train, y_train)
```

```
C:\Users\Dovlat\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py:841: DeprecationWarning: The default of the `iid` parameter will ch
ange from True to False in version 0.22 and will be removed in 0.24. This will change numeric results when test-set sizes are unequal.
  DeprecationWarning)
```

Out[34]:

```
GridSearchCV(cv=3, error_score='raise-deprecating',
       estimator=GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=3,
              max_features=None, max_leaf_nodes=None,
          max_features=None, max_leaf_nodes=None,
          min_impurity_decrease=0.0, min_impurity_split=None,
          min_samples_leaf=1, min_sampl...      subsample=1.0, tol=0.0001, validation_fraction=0.1,
          verbose=0, warm_start=False),
    fit_params=None, iid='warn', n_jobs=None,
    param_grid={'n_estimators': [1, 3, 5, 7, 10], 'max_depth': [1, 3, 5, 7, 10], 'learning_rate': [0.001, 0.0025, 0.005, 0.0075, 0.01, 0.025]},
    pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
    scoring='accuracy', verbose=0)
```

In [35]:

```
best_random_forest.best_params_
```

Out[35]:

```
{'max_depth': 1, 'n_estimators': 1, 'random_state': 6}
```

In [36]:

```
best_gradient_boosting.best_params_
```

Out[36]:

```
{'learning_rate': 0.025, 'max_depth': 1, 'n_estimators': 5}
```

Для оптимальных значений:

In [37]:

```
#случайный лес (n_estimators - число дерневьев)

randomforest2 = RandomForestClassifier(n_estimators=1, max_depth=1, random_state=6).fit(X_train, y_train)
```

In [38]:

```
target_randomforest2 = randomforest2.predict(X_test)
```

In [39]:

```
accuracy_score(y_test, target_randomforest2), \
precision_score(y_test, target_randomforest2),  \
recall_score(y_test, target_randomforest2)
```

Out[39]:

```
(1.0, 1.0, 1.0)
```

In [40]:

```
#градиентный бустинг
gradient_boosting2 = GradientBoostingClassifier(n_estimators=5, max_depth=1, learning_rate=0.025).fit(X_train, y_train)
```

In [41]:

```
target_gradient_boosting2 = gradient_boosting2.predict(X_test)
```

In [42]:

```
accuracy_score(y_test, target_gradient_boosting2), \
precision_score(y_test, target_gradient_boosting2),  \
recall_score(y_test, target_gradient_boosting2)
```

Out[42]:

```
(1.0, 1.0, 1.0)
```