Ансамбли моделей машинного обучения

In [70]:

```
import numpy as np
import pandas as pd
from typing import Dict, Tuple
from scipy import stats
from scipy.optimize import fmin_tnc
from IPython.display import Image
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classification_repor
t
from sklearn.metrics import confusion_matrix
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_err
or, median_absolute_error, r2_score, root_mean_squared_error
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import SGDRegressor
from sklearn.linear_model import SGDClassifier
from sklearn import linear_model
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import GradientBoostingRegressor
import xgboost as xgb
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

In [6]:

```
mpg = pd.read_csv('C:\\MGTU\\6 semestr\\TMO\\auto-mpg.csv')
```

In [7]:

```
mpg.head()
```

Out[7]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	**18.0**	**8**	**307.0**	**130**	**3504**	**12.0**	**70**	**1**	**chevrolet chevelle malibu**
1	**15.0**	**8**	**350.0**	**165**	**3693**	**11.5**	**70**	**1**	**buick skylark 320**
2	**18.0**	**8**	**318.0**	**150**	**3436**	**11.0**	**70**	**1**	**plymouth satellite**
3	**16.0**	**8**	**304.0**	**150**	**3433**	**12.0**	**70**	**1**	**amc rebel sst**
4	**17.0**	**8**	**302.0**	**140**	**3449**	**10.5**	**70**	**1**	**ford torino**

In [8]:

```
mpg.dtypes
```

Out[8]:

```
mpg             float64
cylinders         int64
displacement    float64
horsepower       object
weight            int64
acceleration    float64
model year        int64
```

```
model year          int64
origin              int64
car name           object
dtype: object
```

In [9]:

```
mpg = mpg[mpg['horsepower'] != '?']
mpg['horsepower'] = mpg['horsepower'].astype(float)
```

In [10]:

```
mpg.dtypes
```

Out[10]:

```
mpg              float64
cylinders          int64
displacement     float64
horsepower       float64
weight             int64
acceleration     float64
model year         int64
origin             int64
car name          object
dtype: object
```

In [11]:

```
mpg = mpg.drop(columns=['car name'])
```

In [12]:

```
X = mpg.drop(columns=['mpg'])  # Признаки
y = mpg['mpg']  # Целевая переменная

# Разделение данных на обучающую и тестовую выборки
mpg_X_train, mpg_X_test, mpg_y_train, mpg_y_test = train_test_split(X, y, test_size=0.2,
random_state=1)
```

Bagging

In [53]:

```
br_mpg = BaggingRegressor(n_estimators=1000, oob_score = False, random_state=10)
br_mpg.fit(mpg_X_train,mpg_y_train )
br_mpg_predict =  br_mpg.predict(mpg_X_test)
```

In [54]:

```
root_mean_squared_error(mpg_y_test, br_mpg_predict)
```

Out[54]:

```
3.1432575417344863
```

Random Forest

In [55]:

```
rfr_mpg = RandomForestRegressor(n_estimators=1000, oob_score = True, random_state=10)
rfr_mpg.fit(mpg_X_train,mpg_y_train )
rfr_mpg_predict =  rfr_mpg.predict(mpg_X_test)
```

In [56]:

```
root_mean_squared_error(mpg_y_test, rfr_mpg_predict)
```

Out[56]:

3.151838599486819

In [89]:

```
rfr_mpg1 = RandomForestRegressor(n_estimators=1000, oob_score = True, max_features = 2 ,
min_samples_leaf = 5, random_state=10)
rfr_mpg1.fit(mpg_X_train,mpg_y_train )
rfr_mpg_predict1 =  rfr_mpg1.predict(mpg_X_test)
```

In [90]:

```
root_mean_squared_error(mpg_y_test, rfr_mpg_predict1)
```

Out[90]:

3.55224352114633

Extremely Randomized Trees

In [59]:

```
etr_mpg = ExtraTreesRegressor(n_estimators=1000, random_state=10)
etr_mpg.fit(mpg_X_train,mpg_y_train )
etr_mpg_predict =  etr_mpg.predict(mpg_X_test)
```

In [60]:

```
root_mean_squared_error(mpg_y_test, etr_mpg_predict)
```

Out[60]:

3.312486072538161

AdaBoost

In [64]:

```
abr_mpg = AdaBoostRegressor(n_estimators=1000, random_state=10)
abr_mpg.fit(mpg_X_train,mpg_y_train )
abr_mpg_predict =  abr_mpg.predict(mpg_X_test)
```

In [65]:

```
root_mean_squared_error(mpg_y_test, abr_mpg_predict)
```

Out[65]:

3.620154134360523

GradientBoosting

In [67]:

```
gbr_mpg = GradientBoostingRegressor(n_estimators=1000, random_state=10)
gbr_mpg.fit(mpg_X_train,mpg_y_train )
gbr_mpg_predict =  gbr_mpg.predict(mpg_X_test)
```

In [68]:

```
root_mean_squared_error(mpg_y_test, gbr_mpg_predict)
```

Out[68]:

3.1436494559859263

XGBoost

In [77]:

```
dtrain = xgb.DMatrix(mpg_X_train, label=mpg_y_train)
dtest = xgb.DMatrix(mpg_X_test, label=mpg_y_test)

params = {
    'objective': 'reg:squarederror', # функция потерь для задачи регрессии
    'eval_metric': 'rmse'
}

num_rounds = 1000
model = xgb.train(params, dtrain, num_rounds)

y_pred = model.predict(dtest)
```

In [78]:

```
root_mean_squared_error(mpg_y_test, y_pred)
```

Out[78]:

```
3.2797096825132277
```