In [1]:
```
from funcs.base import *

import openpyxl
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib. pyplot as plt
%matplotlib inline
sns.set(style="ticks")


from sklearn.metrics import *
from sklearn.neighbors import *
from sklearn.tree import *
from sklearn.linear_model import *
from sklearn.svm import *
from sklearn.ensemble import *

from sklearn.preprocessing import MinMaxScaler
```

1. Описание

In [2]:
```
database = pd.read_csv('epi_r.csv', sep=",")
```

In [3]:
```
database.shape, database.shape[0] * database.shape[1]
```

Out[3]: ((20052, 680), 13635360)

In [4]:
```
pd.set_option('display.max_columns', None)
database
```

Out[4]:

	title	rating	calories	protein	fat	sodium	#cakeweek	#wasteless	22-minute meals	3-ingredient recipes	30 days of groceries	advance prep required	alabama	alaska	alcoholic	almond	amaretto	anchovy	anise	anniversary	anthony bourdain	aperitif	appetizer	apple	apple juice	apricot	arizona	artichoke	arugula	asian pear	asparagus	aspe
0	Lentil, Apple, and Turkey Wrap	2.500	426.0	30.0	7.0	559.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
1	Boudin Blanc Terrine with Red Onion Confit	4.375	403.0	18.0	23.0	1439.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
2	Potato and Fennel Soup Hodge	3.750	165.0	6.0	7.0	165.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
3	Mahi-Mahi in Tomato Olive Sauce	5.000	NaN	NaN	NaN	NaN	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
4	Spinach Noodle Casserole	3.125	547.0	20.0	32.0	452.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
20047	Parmesan Puffs	3.125	28.0	2.0	2.0	64.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
20048	Artichoke and Parmesan Risotto	4.375	671.0	22.0	28.0	583.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0
20049	Turkey Cream Puff Pie	4.375	563.0	31.0	38.0	652.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
20050	Snapper on Angel Hair with Citrus Cream	4.375	631.0	45.0	24.0	517.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
20051	Baked Ham with Marmalade-Horseradish Glaze	4.375	560.0	73.0	10.0	3698.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0

20052 rows × 680 columns

2. Минимизация

2.1. Удаление строк

In [5]:
```
# Удаление пустых строк
buff = database.dropna(axis=0)
buff.shape
```

Out[5]: (15864, 680)

2.2. Формирование фреймов по уникальным значениям y

In [6]:
```
# Уникальные значения поля rating
l = np.unique(buff[['rating']])
print(f'Количество уникальных значений поля rating : {len(l)}\n{l}' )
```

```
Количество уникальных значений поля rating : 8
[0.    1.25  1.875 2.5   3.125 3.75  4.375 5.   ]
```

In [7]:
```
# Выбираем все уникальные строки с определенным rating
arrgroup = []
for i in l:
    arrgroup.append(buff[buff['rating'] == i])
```

In [8]:
```
# Узнаем размерность каждого dataframe
for i in arrgroup:
    print(i.shape)
```

```
(1296, 680)
(123, 680)
(81, 680)
(405, 680)
(1165, 680)
(4136, 680)
(6552, 680)
(2106, 680)
```

2.3. Выборка из каждого dataframe n% строк и добавляем в нывую таблицу

In [9]:
```
# В данной ситуации выбираем 33% строк
m_frac = 0.09
```

In [10]:
```
# Новый фрейм
newdata = pd.DataFrame()
```

In [11]:
```
# Осуществляем выборку по каждому диапазону
for i in arrgroup:
    newdata = newdata.append(i.sample(frac = m_frac))
```

In [12]:
```
newdata.shape
```

Out[12]: (1428, 680)

In [13]:
```
newdata
```

Out[13]:

	title	rating	calories	protein	fat	sodium	#cakeweek	#wasteless	22-minute meals	3-ingredient recipes	30 days of groceries	advance prep required	alabama	alaska	alcoholic	almond	amaretto	anchovy	anise	anniversary	anthony bourdain	aperitif	appetizer	apple	apple juice	apricot	arizona	artichoke	arugula	asian pear	asparagus	aspen
7706	Carrot Tart with Ricotta and Herbs	0.0	360.0	11.0	27.0	160.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13464	Peach Patty-Cakes with Cream	0.0	599.0	8.0	27.0	472.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1943	Broiled Cod With Fennel and Orange	0.0	476.0	40.0	20.0	703.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
818	Roast Rack of Lamb	0.0	1004.0	47.0	87.0	916.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14780	Stuffing Fried Rice	0.0	795.0	16.0	26.0	983.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5607	Kabocha Squash and Pork Stir-Fry	5.0	258.0	10.0	21.0	660.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
528	Bee's Knees	5.0	136.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10938	Jerusalem Artichoke and Artichoke Heart Linguine	5.0	700.0	20.0	21.0	1869.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
16912	Roast Chicken With Harissa And Schmaltz	5.0	1248.0	91.0	87.0	22932.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1903	Roast Turkey with Savory Cranberry Sauce From ...	5.0	1512.0	179.0	55.0	2904.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

1428 rows × 680 columns

2.4. Делаем корреляцию данных и выясняем какие столбцы не влияют на таблицу

In [14]:
```
corr = newdata.corr()
```

In [15]:
```
# Сумма пропущенных значений
isnull = corr.isnull().sum()
summNan = isnull.to_dict()

keys = []
index = 0
# Если значение пропусков больше половины, то запоминае удаляемый столбец
for key, val in summNan.items():
    if val >= buff.shape[1] / 2:
        keys.append(key)
```

In [16]:
```
#isnull.to_dict()
```

2.5 Производим удаление ненужных столбцов

In [17]:
```
for i in keys:
    newdata = newdata.drop([i], axis = 1)
```

In [18]:
```
newdata.shape
```

Out[18]: (1428, 519)

Формирование данных

In [19]:
```
x = newdata.drop(['rating','title'], axis = 1)
y = newdata[['rating']]
mm = MinMaxScaler()
x=pd.DataFrame(mm.fit_transform(x))
y=pd.DataFrame(mm.fit_transform(y))
```

Будем рассматривать данную задачу со стороны классификации

In [20]:
```
from sklearn.utils.multiclass import type_of_target
from sklearn.preprocessing import LabelEncoder
type_of_target(y)
```

Out[20]: 'continuous'

In [21]:
```
one = LabelEncoder()
y = one.fit_transform(y.values.ravel())
```

In [22]:
```
type_of_target(y)
```

Out[22]: 'multiclass'

In [23]:
```
np.unique(y)
```

Out[23]: array([0, 1, 2, 3, 4, 5, 6, 7], dtype=int64)

In [24]:
```
y = pd.DataFrame(y)
```

Формирование моделей для решения задачи регрессии

In [25]:
```
metrics = [accuracy_score, balanced_accuracy_score]

models = [
    {
        'model': BaggingClassifier(n_estimators=5, random_state=10),
        'grid': {
            'n_estimators': np.arange(1, 5, 1),
            'random_state': np.arange(1, 5, 1)
        },
        'metrics': metrics
    },
    {
        'model': AdaBoostClassifier(n_estimators=5, random_state=10),
        'grid': {
            'n_estimators': np.arange(1, 5, 1),
            'random_state': np.arange(1, 5, 1)
        },
        'metrics': metrics
    }
]
```

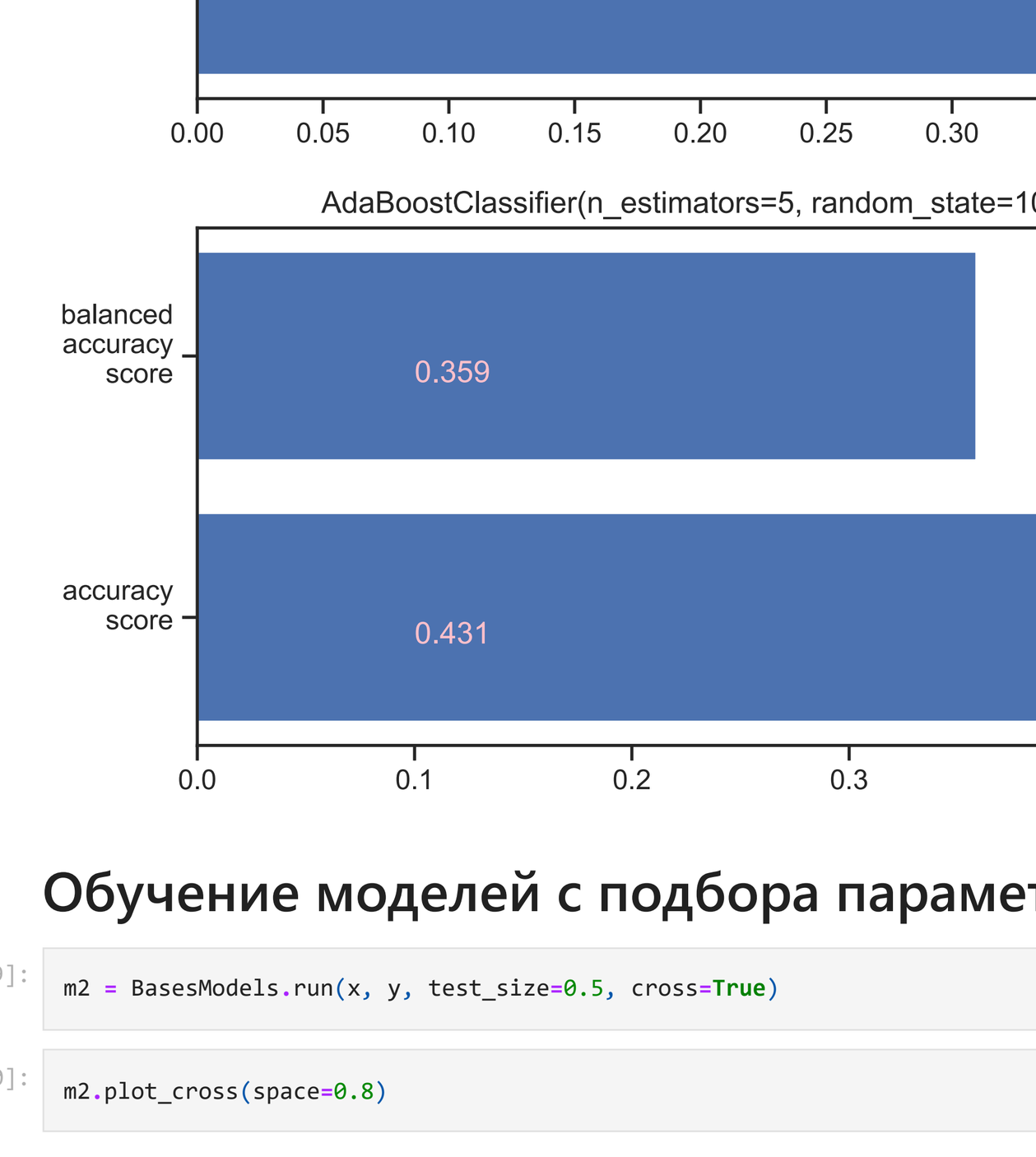
In [26]:
```
BasesModels = Analysing(models)
```

Обучение моделей без подбора параметров

In [27]:
```
m1 = BasesModels.run(x, y, test_size=0.25)
```

```
C:\Users\Vantus\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
C:\Users\Vantus\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\metrics\_classification.py:1850: UserWarning: y_pred contains classes not in y_true
  warnings.warn('y_pred contains classes not in y_true')
C:\Users\Vantus\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
C:\Users\Vantus\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\metrics\_classification.py:1850: UserWarning: y_pred contains classes not in y_true
  warnings.warn('y_pred contains classes not in y_true')
```

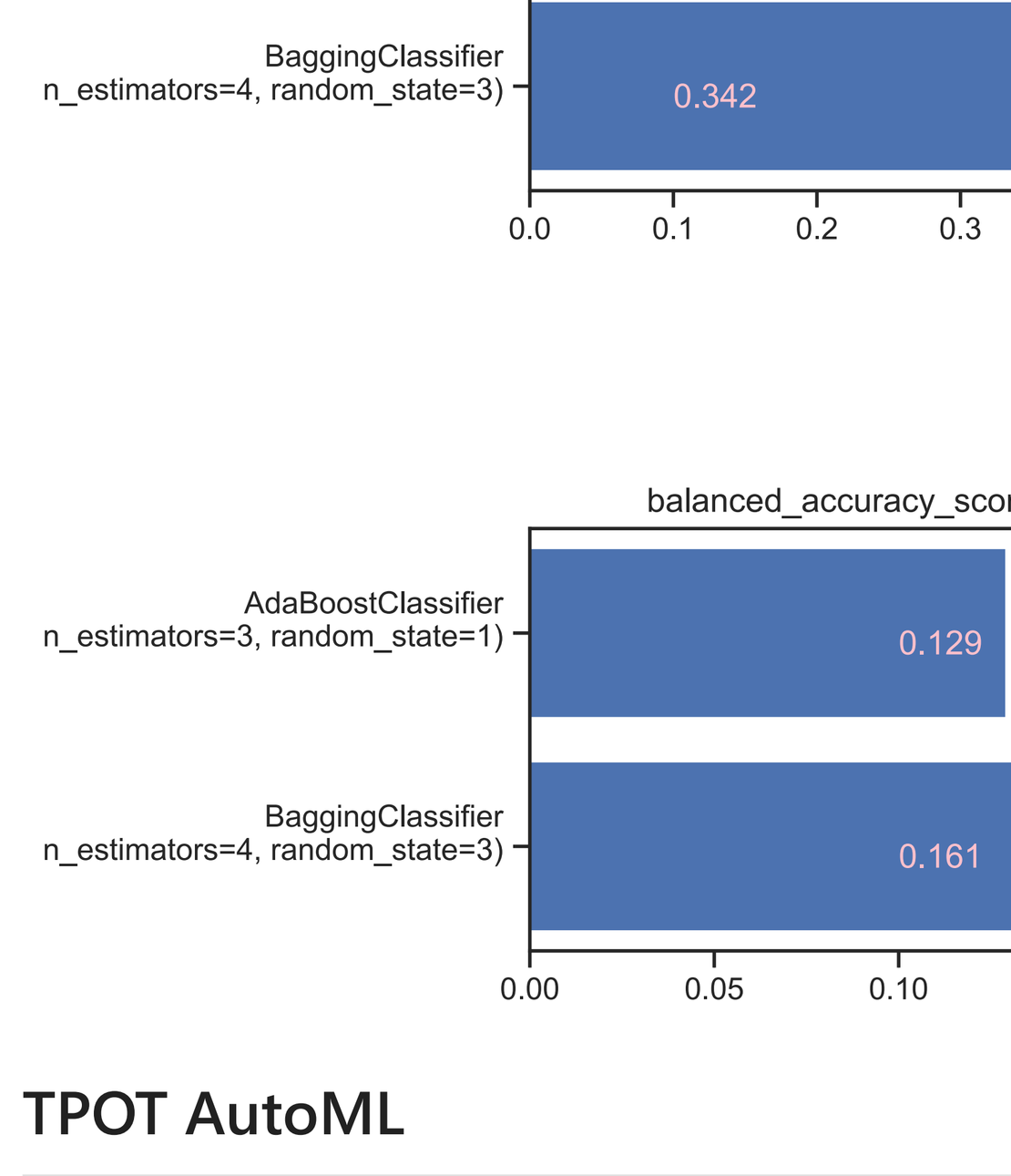
In [28]:
```
m1.plot(space=0.25)
```



Обучение моделей с подбора параметров

In [29]:
```
m2 = BasesModels.run(x, y, test_size=0.5, cross=True)
```

In [30]:
```
m2.plot_cross(space=0.8)
```



TPOT AutoML

In [31]:
```
# from tpot import *
# from sklearn.model_selection import train_test_split
```

Деление выборки

In [32]:
```
# train_x, test_x, train_y, test_y = train_test_split(x, y, test_size=0.25, random_state=1)
```

Создание экземпляра

In [33]:
```
# tp = TPOTClassifier()
```

In [1]:
```
# tp.fit(train_x, train_y)
```

Формирование выводов о качестве построенных моделей на основе выбранных метрик.

Web отчет