Импорт необходимых модулей

In [1]:

```
!pip install missingno
```

```
Requirement already satisfied: missingno in c:\python37\lib\site-packages
(0.4.1)
Requirement already satisfied: matplotlib in c:\python37\lib\site-packages
(from missingno) (3.0.0)
Requirement already satisfied: scipy in c:\python37\lib\site-packages (from
missingno) (1.1.0)
Requirement already satisfied: seaborn in c:\python37\lib\site-packages (fro
m missingno) (0.9.0)
Requirement already satisfied: numpy in c:\python37\lib\site-packages (from
missingno) (1.16.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
c:\python37\lib\site-packages (from matplotlib->missingno) (2.2.2)
Requirement already satisfied: cycler>=0.10 in c:\python37\lib\site-packages
(from matplotlib->missingno) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\python37\lib\site-
packages (from matplotlib->missingno) (2.7.3)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\python37\lib\site-pac
kages (from matplotlib->missingno) (1.0.1)
Requirement already satisfied: pandas>=0.15.2 in c:\python37\lib\site-packag
es (from seaborn->missingno) (0.23.4)
Requirement already satisfied: six in c:\python37\lib\site-packages (from cy
cler>=0.10->matplotlib->missingno) (1.10.0)
Requirement already satisfied: setuptools in c:\python37\lib\site-packages
(from kiwisolver>=1.0.1->matplotlib->missingno) (40.5.0)
Requirement already satisfied: pytz>=2011k in c:\python37\lib\site-packages
(from pandas>=0.15.2->seaborn->missingno) (2017.2)

You are using pip version 18.1, however version 19.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip'
command.
```

In [2]:

```
import numpy as np
import pandas as pd
import missingno as msno
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="whitegrid")
```

Загрузка данных. Первый взгляд

In [3]:

```
beer_recipe = pd.read_csv('beer-recipes/recipeData.csv', index_col='BeerID', encoding=
beer_recipe.head()
```

Out[3]:

	Name	URL	Style	StyleID	Size(L)	OG	
BeerID							
1	Vanilla Cream Ale	/homebrew/recipe/view/1633/vanilla-cream-ale	Cream Ale	45	21.77	1.055	1
2	Southern Tier Pumking clone	/homebrew/recipe/view/16367/southern-tier-pumk...	Holiday/Winter Special Spiced Beer	85	20.82	1.083	1
3	Zombie Dust Clone - EXTRACT	/homebrew/recipe/view/5920/zombie-dust-clone-e...	American IPA	7	18.93	1.063	1
4	Zombie Dust Clone - ALL GRAIN	/homebrew/recipe/view/5916/zombie-dust-clone-a...	American IPA	7	22.71	1.061	1
5	Bakke Brygg Belgisk Blonde 50 l	/homebrew/recipe/view/89534/bakke-brygg-belgis...	Belgian Blond Ale	20	50.00	1.060	1

5 rows × 22 columns

In [4]:

```
print(beer_recipe.info(verbose=False))
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 73861 entries, 1 to 73861
Columns: 22 entries, Name to UserId
dtypes: float64(13), int64(2), object(7)
memory usage: 13.0+ MB
None
```

Детальный анализ данных

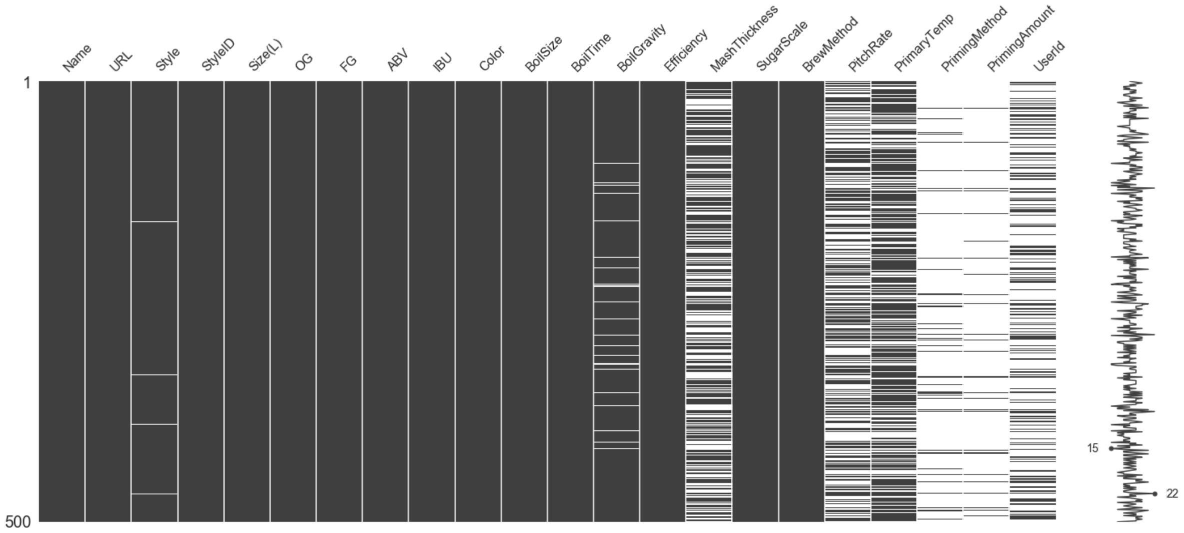
Пропуски

In [5]:

```
%matplotlib inline
msno.matrix(beer_recipe.sample(500))
```

Out[5]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x25b8b41eb00>
```



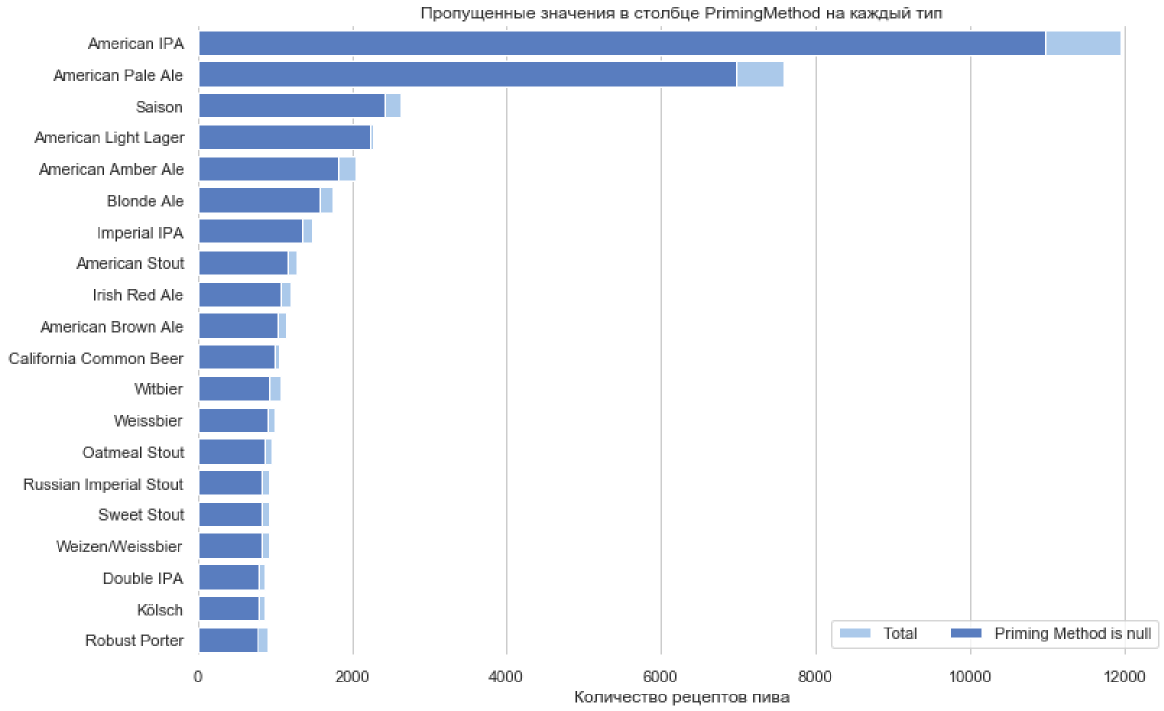
In [6]:

```
null_priming = beer_recipe['PrimingMethod'].isnull()
print('Priming Method пропущено в {} строк из {}, т.е. в {} % случаев'
      .format(null_priming.sum(), len(beer_recipe), round((null_priming.sum()/len(beer_
```

```
Priming Method пропущено в 67095 строк из 73861, т.е. в 90.84 % случаев
```

In [7]:

```
style_cnt = beer_recipe.loc[:,['Style','PrimingMethod']]
style_cnt['NullPriming'] = style_cnt['PrimingMethod'].isnull()
style_cnt['Count'] = 1
style_cnt_grp = style_cnt.loc[:,['Style','Count','NullPriming']].groupby('Style').sum(

style_cnt_grp = style_cnt_grp.sort_values('NullPriming', ascending=False)
style_cnt_grp.reset_index(inplace=True)

def stacked_bar_plot(df, x_total, x_sub_total, sub_total_label, y):

    f, ax = plt.subplots(figsize=(12, 8))

    sns.set_color_codes("pastel")
    sns.barplot(x=x_total, y=y, data=df, label="Total", color="b")

    sns.set_color_codes("muted")
    sns.barplot(x=x_sub_total, y=y, data=df, label=sub_total_label, color="b")

    ax.legend(ncol=2, loc="lower right", frameon=True)
    sns.despine(left=True, bottom=True)

    return f, ax

f, ax = stacked_bar_plot(style_cnt_grp[:20], 'Count', 'NullPriming', 'Priming Method i
ax.set(title='Пропущенные значения в столбце PrimingMethod на каждый тип', ylabel='', 
sns.despine(left=True, bottom=True)
```



Дизбаланс классов

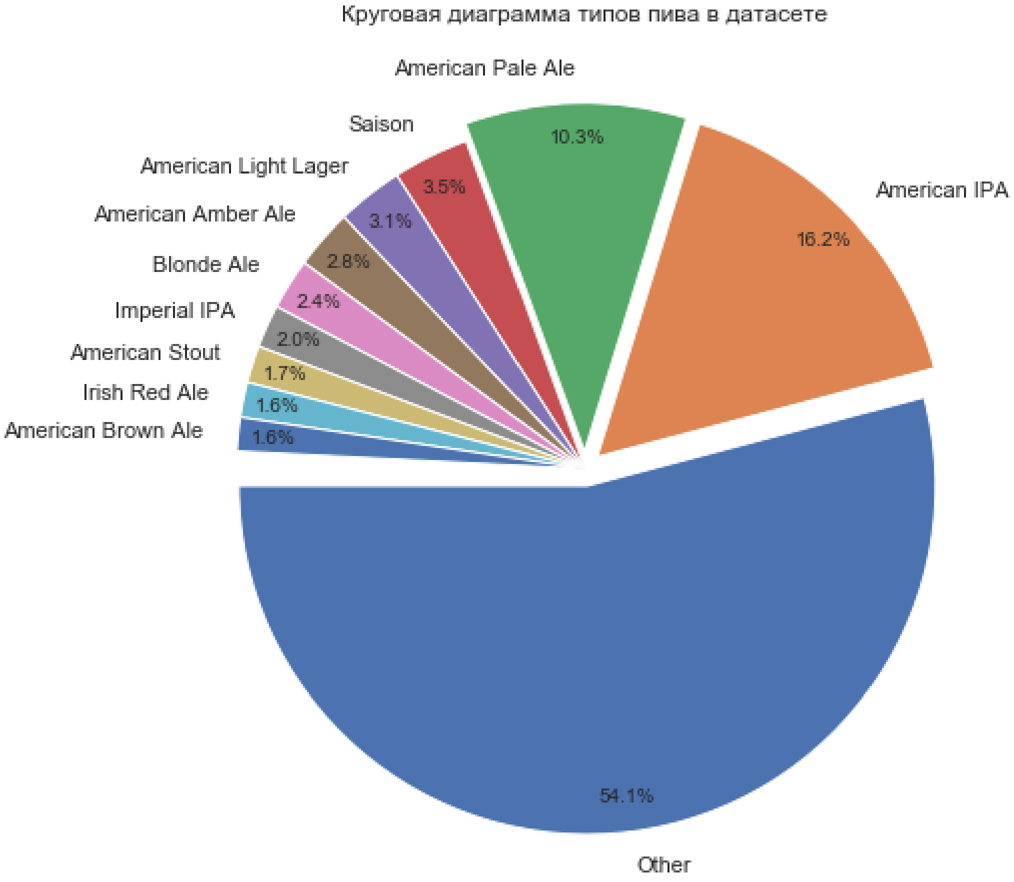
In [8]:

```
print('В датасете {} различных типов пива'.format(beer_recipe.StyleID.nunique()))
```

```
В датасете 176 различных типов пива
```

In [9]:

```
top10_style = list(style_cnt_grp['Style'][:10].values)

style_cnt_other = style_cnt_grp.loc[:, ['Style','Count']]
style_cnt_other.Style = style_cnt_grp.Style.apply(lambda x: x if x in top10_style else
style_cnt_other = style_cnt_other.groupby('Style').sum()

style_cnt_other['Ratio'] = style_cnt_other.Count.apply(lambda x: x/float(len(beer_reci
style_cnt_other = style_cnt_other.sort_values('Count', ascending=False)

f, ax = plt.subplots(figsize=(8, 8))
explode = (0.05, 0.05, 0.05, 0, 0, 0, 0, 0, 0, 0, 0)
plt.pie(x=style_cnt_other['Ratio'], labels=list(style_cnt_other.index), startangle = 1
plt.title('Круговая диаграмма типов пива в датасете')
plt.show()
```



Т.к. типов слишком много, для классификации я взял только 10 из 176 самых популярных классов, которые формируют из себя 46 процентов данных. При этом, чтобы уравнять распределения классов в данных, я отбросил долю данных о двух резко выделяющихся типах пива.

In [10]:

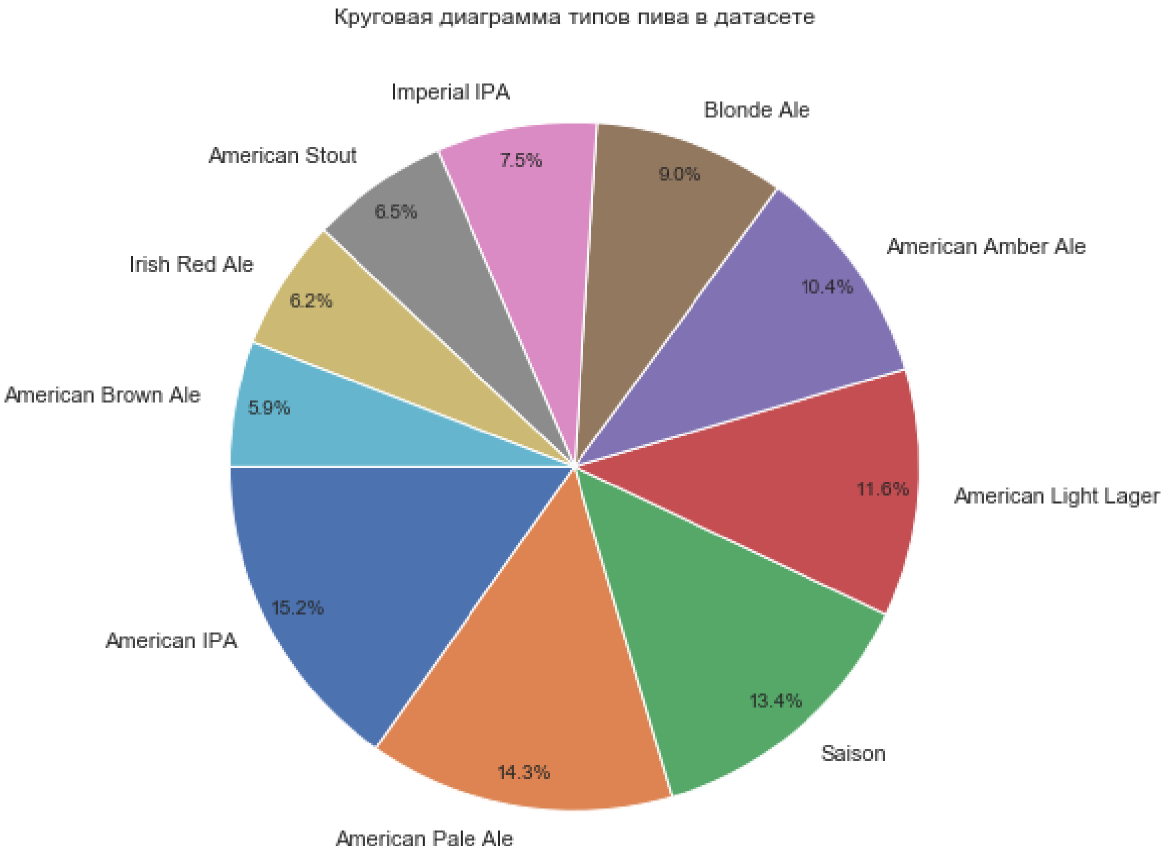
```
beer_recipe = beer_recipe[beer_recipe['Style'].isin(style_cnt_grp['Style'][:10].values
index_apa = beer_recipe[beer_recipe['Style']=='American Pale Ale']\
.index[:int(beer_recipe[beer_recipe['Style']=='American Pale Ale'].shape[0]*63//100)]
index_ipa = beer_recipe[beer_recipe['Style']=='American IPA']\
.index[:beer_recipe[beer_recipe['Style']=='American IPA'].shape[0]*75//100]
beer_recipe.drop(index_apa , inplace=True)
beer_recipe.drop(index_ipa , inplace=True)
```

In [11]:

```
style_cnt = beer_recipe.loc[:,['Style','PrimingMethod']]
style_cnt['NullPriming'] = style_cnt['PrimingMethod'].isnull()
style_cnt['Count'] = 1
style_cnt_grp = style_cnt.loc[:,['Style','Count','NullPriming']].groupby('Style').sum(

style_cnt_grp = style_cnt_grp.sort_values('NullPriming', ascending=False)
style_cnt_grp.reset_index(inplace=True)
```

In [12]:

```
top10_style = list(style_cnt_grp['Style'][:10].values)

style_cnt_other = style_cnt_grp.loc[:, ['Style','Count']]
style_cnt_other.Style = style_cnt_grp.Style.apply(lambda x: x if x in top10_style else
style_cnt_other = style_cnt_other.groupby('Style').sum()

style_cnt_other['Ratio'] = style_cnt_other.Count.apply(lambda x: x/float(len(beer_reci
style_cnt_other = style_cnt_other.sort_values('Count', ascending=False)

f, ax = plt.subplots(figsize=(8, 8))
explode = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
plt.pie(x=style_cnt_other['Ratio'], labels=list(style_cnt_other.index), startangle = 1
plt.title('Круговая диаграмма типов пива в датасете')
plt.show()
```



Категориальные фичи

In [13]:

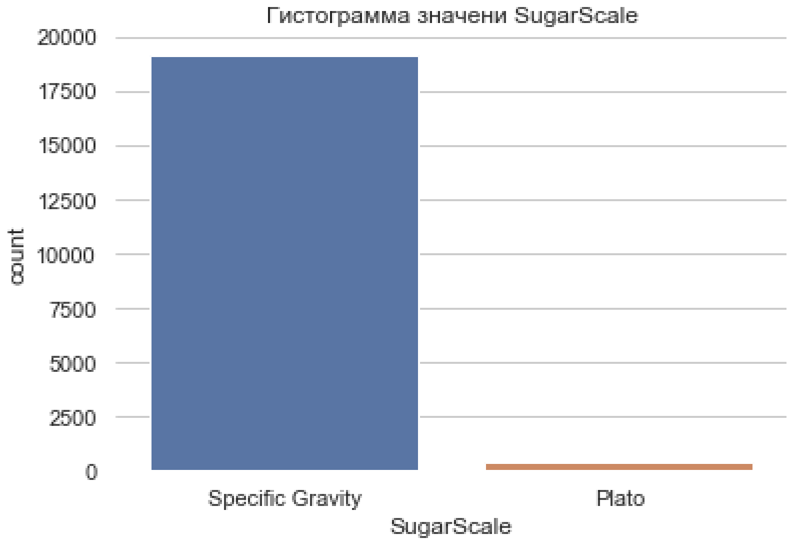
```
print( list(beer_recipe.select_dtypes(include=object).columns))
```

```
['Name', 'URL', 'Style', 'SugarScale', 'BrewMethod', 'PrimingMethod', 'Primi
ngAmount']
```

In [14]:

```
ax = sns.countplot(x='SugarScale', data=beer_recipe)
ax.set(title='Гистограмма значени SugarScale')
sns.despine(left=True, bottom=True)

print('В столбце SugarScale {} пропущенных значений'.format(beer_recipe.SugarScale.isn
```

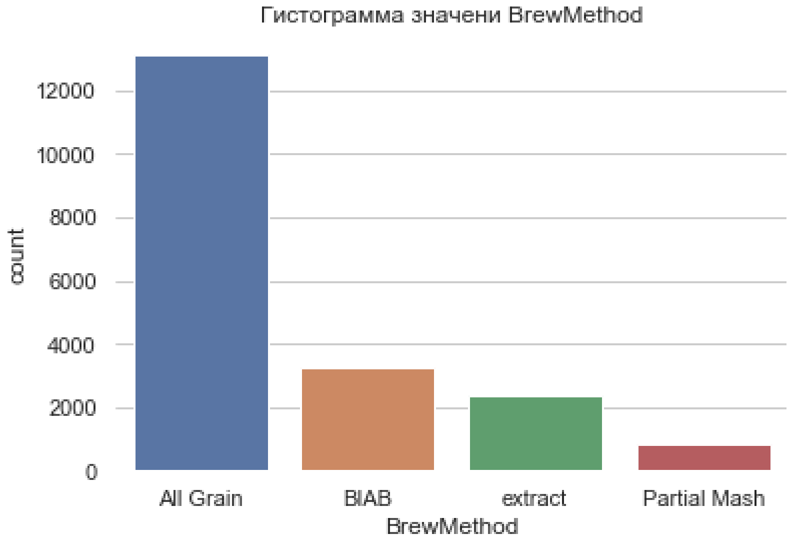
```
В столбце SugarScale 0 пропущенных значений
```



In [15]:

```
ax = sns.countplot(x='BrewMethod', data=beer_recipe)
ax.set(title='Гистограмма значени BrewMethod')
sns.despine(left=True, bottom=True)

print('В столбце BrewMethod {} пропущенных значений'.format(beer_recipe.BrewMethod.isn
```

```
В столбце BrewMethod 0 пропущенных значений
```



In [16]:

```
print('В столбце PrimingMethod {} пропущенных значений'.format(beer_recipe.PrimingMeth
print(beer_recipe.PrimingMethod.unique()[:20])
```

```
В столбце PrimingMethod 282 пропущенных значений
[nan 'Corn Sugar' 'corn sugar' 'Sugar' 'Dextrose' 'Sukkerlake'
 'Add in last 5 of boil' 'Cane Sugar' 'Forced CO2' 'Dme'
 'forced carbonation' 'Corn Sugar? Strong Ale yeast?' 'Dark Brown Sugar'
 'Honey' 'Light DME' 'sucrose' 'Table sugar' '3oz' ' Corn Sugar'
 'dextrose']
```

Числовые фичи

In [17]:

```
print(list(beer_recipe.select_dtypes(exclude=object)))
```

```
['StyleID', 'Size(L)', 'OG', 'FG', 'ABV', 'IBU', 'Color', 'BoilSize', 'BoilT
ime', 'BoilGravity', 'Efficiency', 'MashThickness', 'PitchRate', 'PrimaryTem
p', 'UserId']
```

In [18]:

```
def get_sg_from_plato(plato):
    sg = 1 + (plato / (258.6 - ( (plato/258.2) *227.1) ) )
    return sg

beer_recipe['OG_sg'] = beer_recipe.apply(lambda row: get_sg_from_plato(row['OG']) if r
beer_recipe['FG_sg'] = beer_recipe.apply(lambda row: get_sg_from_plato(row['FG']) if r
beer_recipe['BoilGravity_sg'] = beer_recipe.apply(lambda row: get_sg_from_plato(row['B
```

In [19]:

```
num_feats_list = ['Size(L)', 'OG_sg', 'FG_sg', 'ABV', 'IBU', 'Color', 'BoilSize', 'Boi
beer_recipe.loc[:, num_feats_list].describe().T
```

Out[19]:

	count	mean	std	min	25%	50%	75%	max
Size(L)	19577.0	46.851393	189.055791	1.000	18.93	20.820	24.000	6102.080
OG_sg	19577.0	1.058765	0.015172	1.000	1.05	1.057	1.065	1.500
FG_sg	19577.0	1.013183	0.004915	0.999	1.01	1.013	1.015	1.148
ABV	19577.0	5.987758	1.553305	0.000	5.11	5.780	6.660	49.960
IBU	19577.0	46.918919	42.742047	0.000	25.04	36.770	57.910	1359.420
Color	19577.0	11.829264	10.379289	0.000	5.15	7.770	14.300	108.650
BoilSize	19577.0	52.926640	201.348525	1.000	21.77	28.000	30.000	6454.130
BoilTime	19577.0	63.740512	12.369370	0.000	60.00	60.000	60.000	240.000
BoilGravity_sg	19001.0	1.051856	0.024441	1.000	1.04	1.046	1.056	1.500
Efficiency	19577.0	66.011748	14.202624	0.000	65.00	70.000	75.000	100.000
MashThickness	11857.0	2.135589	1.679638	0.300	1.50	1.500	3.000	63.000
PitchRate	8646.0	0.687480	0.332113	0.000	0.35	0.750	0.750	2.000
PrimaryTemp	13087.0	19.885641	3.800540	-17.780	18.33	20.000	21.000	85.000

In [20]:

```
vlow_scale_feats = ['OG_sg', 'FG_sg', 'BoilGravity_sg', 'PitchRate']
low_scale_feats = ['ABV', 'MashThickness']
mid_scale_feats = ['Color', 'BoilTime', 'Efficiency', 'PrimaryTemp']
high_scale_feats = ['IBU', 'Size(L)',  'BoilSize']
```

In [21]:

```
f, ax = plt.subplots(figsize=(12, 8))
ax = sns.boxplot(data=beer_recipe.loc[:, vlow_scale_feats], orient='h')
ax.set(title='График самых мелкомасштабных фич датасета')
sns.despine(left=True, bottom=True)
```

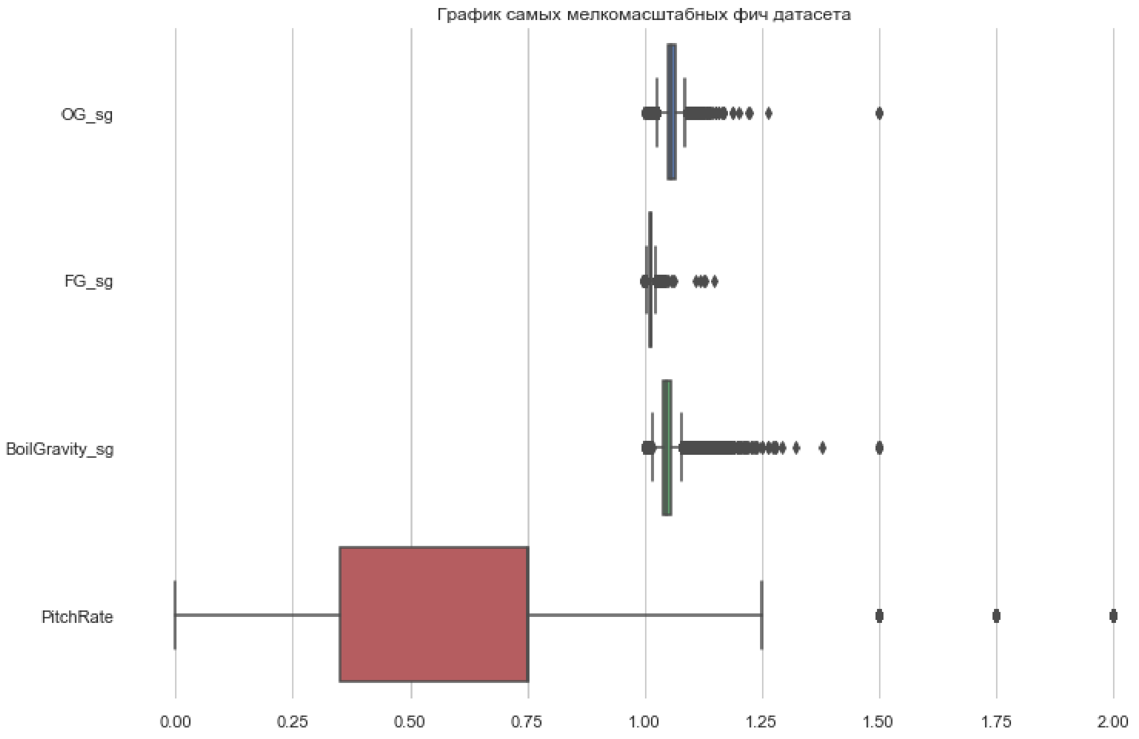


График самых мелкомасштабных фич датасета

In [22]:

```
f, ax = plt.subplots(figsize=(12, 8))
ax = sns.boxplot(data=beer_recipe.loc[:, low_scale_feats], orient='h')
ax.set(title='График мелкомасштабных фич датасета')
sns.despine(left=True, bottom=True)
```

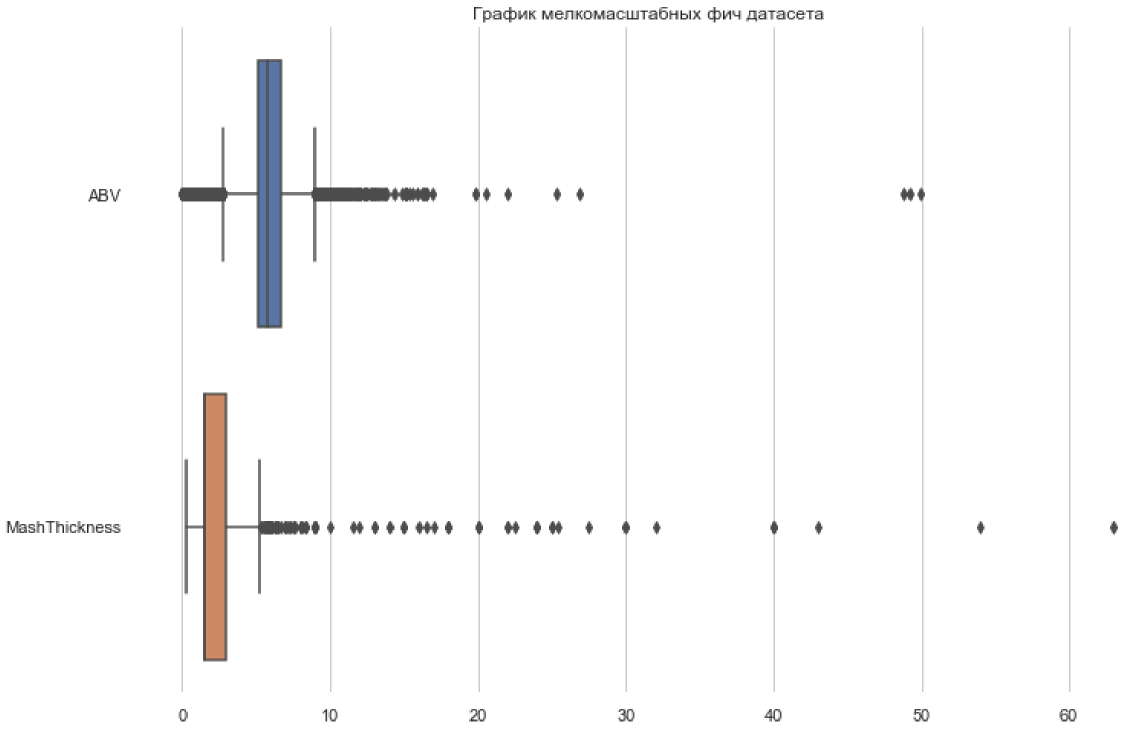
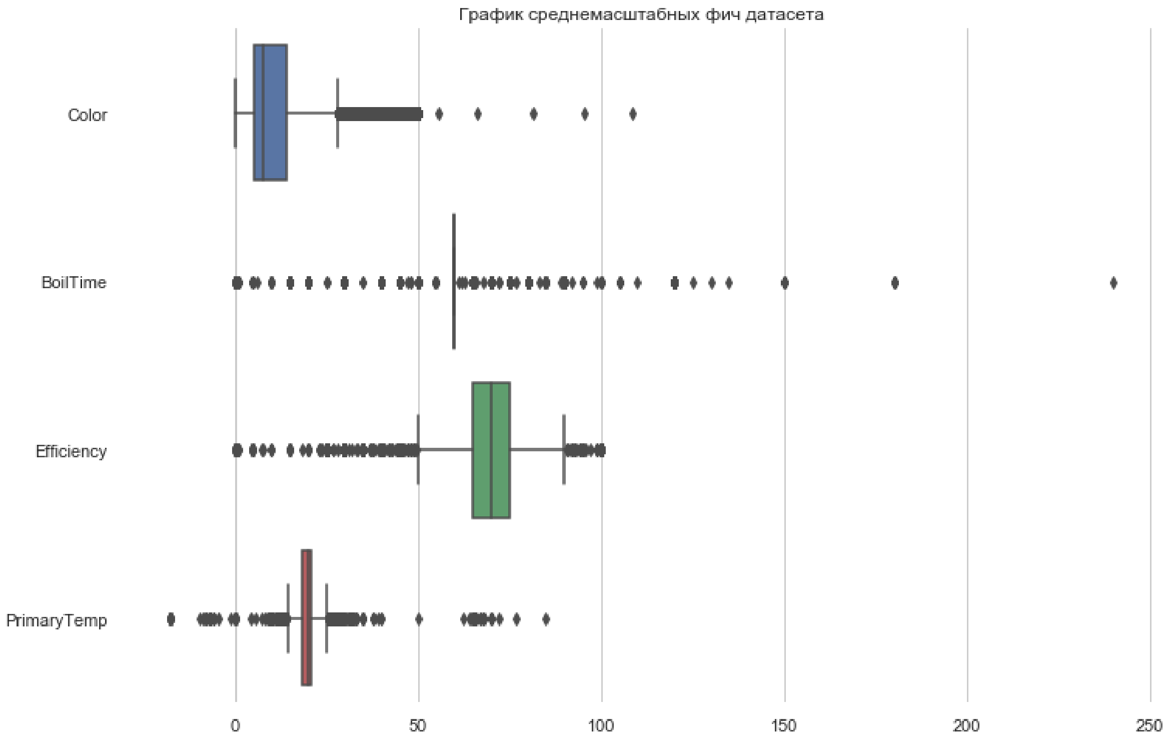


График мелкомасштабных фич датасета

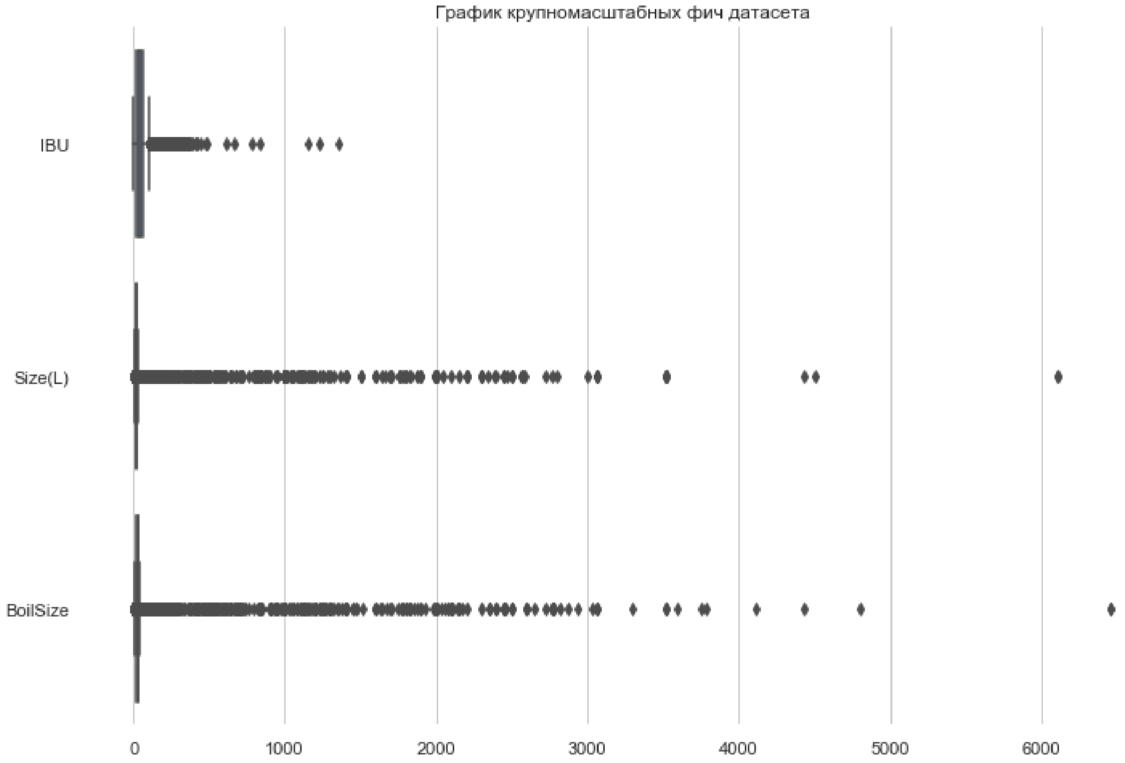
In [23]:

```
f, ax = plt.subplots(figsize=(12, 8))
ax = sns.boxplot(data=beer_recipe.loc[:, mid_scale_feats], orient='h')
ax.set(title='График среднемасштабных фич датасета')
sns.despine(left=True, bottom=True)
```



In [24]:

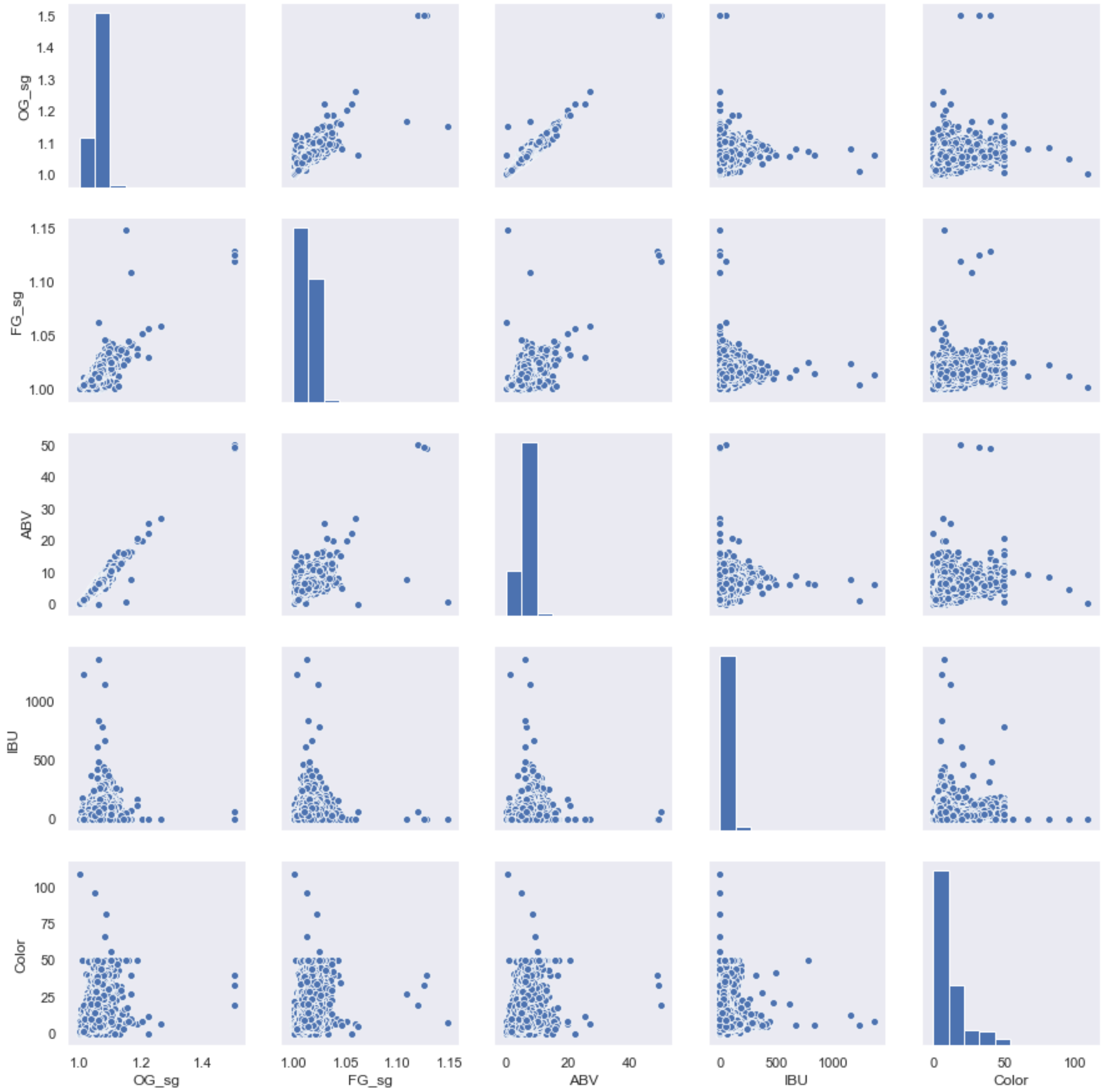
```
f, ax = plt.subplots(figsize=(12, 8))
ax = sns.boxplot(data=beer_recipe.loc[:, high_scale_feats], orient='h')
ax.set(title='График крупномасштабных фич датасета')
sns.despine(left=True, bottom=True)
```



Корреляция

In [25]:

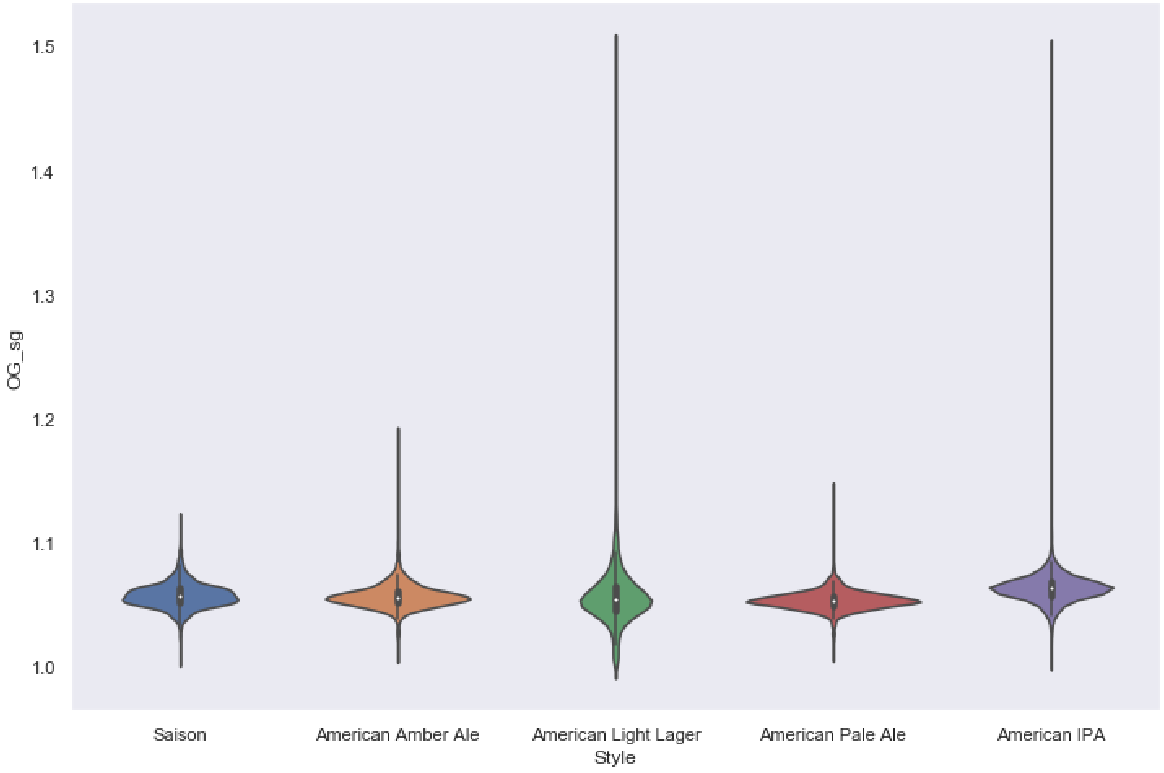
```
pairplot_df = beer_recipe.loc[:, ['Style','OG_sg','FG_sg','ABV','IBU','Color']]

sns.set(style="dark")
sns.pairplot(data=pairplot_df)
plt.show()
```



In [26]:

```
style_cnt_grp = style_cnt_grp.sort_values('Count', ascending=False)
top5_style = list(style_cnt_grp['Style'][:5].values)

top5_style_df = pairplot_df[pairplot_df['Style'].isin(top5_style)]

f, ax = plt.subplots(figsize=(12, 8))
sns.violinplot(x='Style', y='OG_sg',data=top5_style_df)
plt.show()
```

```
c:\python37\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Usin
g a non-tuple sequence for multidimensional indexing is deprecated; use `arr
[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted
as an array index, `arr[np.array(seq)]`, which will result either in an erro
r or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

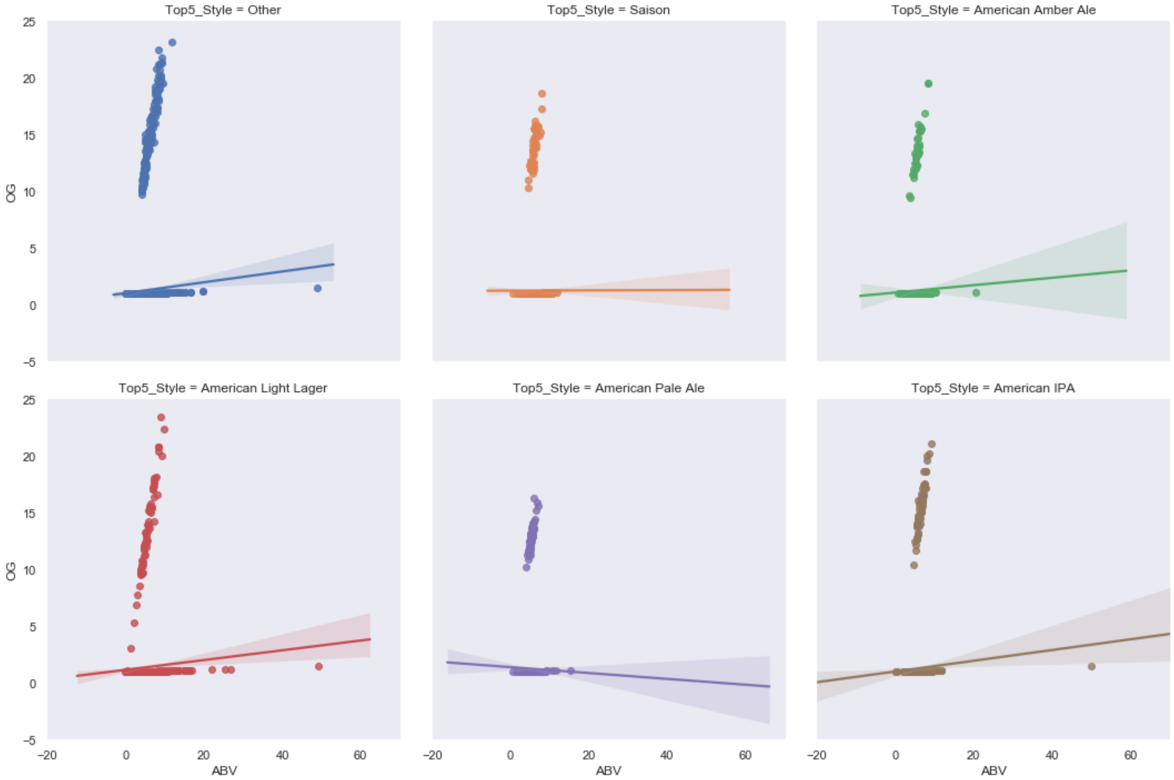


In [27]:

```
top5_style = list(style_cnt_grp['Style'][:5].values)
beer_recipe['Top5_Style'] = beer_recipe.Style.apply(lambda x: x if x in top5_style els

sns.lmplot(x='ABV', y='OG', hue='Top5_Style', col='Top5_Style', col_wrap=3, data=beer_
```

Out[27]:

```
<seaborn.axisgrid.FacetGrid at 0x25b8fc47860>
```

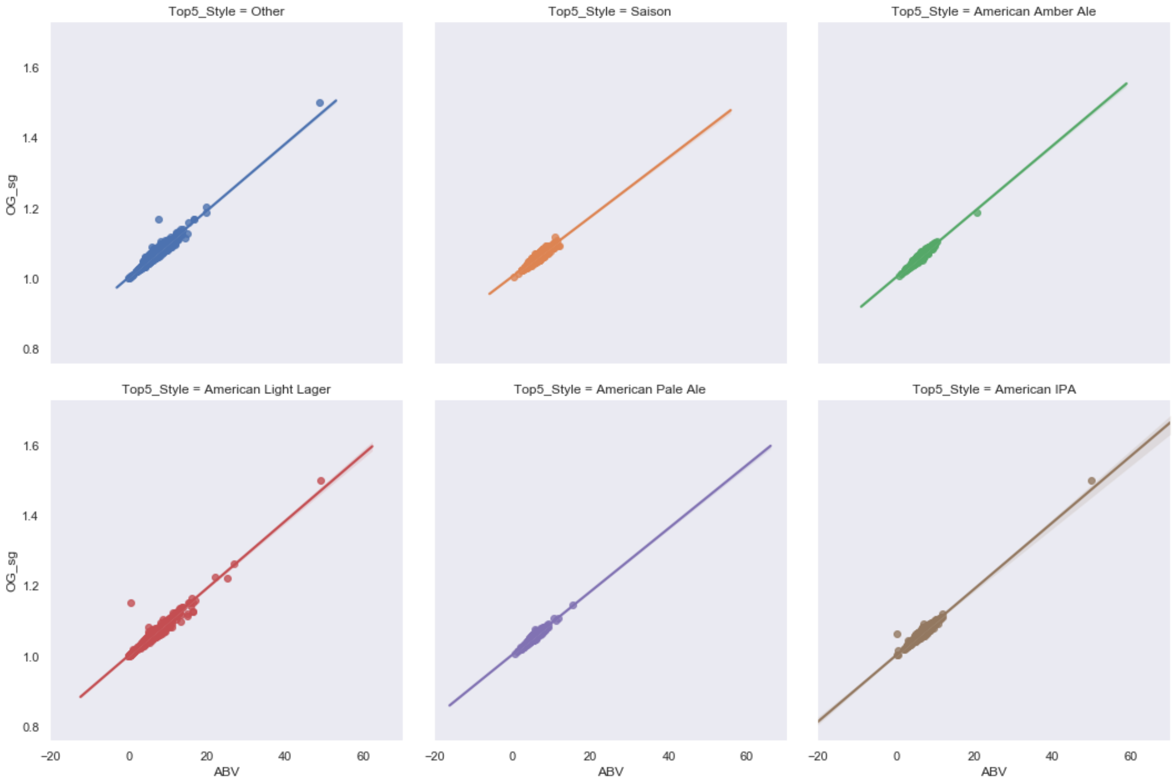


In [28]:

```
sns.lmplot(x='ABV', y='OG_sg', hue='Top5_Style', col='Top5_Style', col_wrap=3, data=be
```

Out[28]:

```
<seaborn.axisgrid.FacetGrid at 0x25b9045f940>
```



Построение моделей

Предобработка данных

In [29]:

```python
from sklearn.preprocessing import LabelEncoder, Imputer
from sklearn.model_selection import train_test_split

features_list= ['StyleID', #целевой признак
                'OG_sg','FG_sg','ABV','IBU','Color',
                'SugarScale', 'BrewMethod',
                'Size(L)', 'BoilSize', 'BoilTime', 'BoilGravity_sg',
                'Efficiency', 'MashThickness', 'PitchRate', 'PrimaryTemp'
                ]

clf_data = beer_recipe.loc[:, features_list]

# Кодирование категориальных значений
cat_feats_to_use = list(clf_data.select_dtypes(include=object).columns)
for feat in cat_feats_to_use:
    encoder = LabelEncoder()
    clf_data[feat] = encoder.fit_transform(clf_data[feat])

#Заполнение пропусков
num_feats_to_use = list(clf_data.select_dtypes(exclude=object).columns)
for feat in num_feats_to_use:
    imputer = Imputer(strategy='median')
    clf_data[feat] = imputer.fit_transform(clf_data[feat].values.reshape(-1,1))

# Выделение целевого признака
X = clf_data.iloc[:, 1:]
y = clf_data.iloc[:, 0]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2, stratify=y, ra
```

```
c:\python37\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWa
rning: Class Imputer is deprecated; Imputer was deprecated in version 0.20 a
nd will be removed in 0.22. Import impute.SimpleImputer from sklearn instea
d.
  warnings.warn(msg, category=DeprecationWarning)
c:\python37\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWa
rning: Class Imputer is deprecated; Imputer was deprecated in version 0.20 a
nd will be removed in 0.22. Import impute.SimpleImputer from sklearn instea
d.
  warnings.warn(msg, category=DeprecationWarning)
c:\python37\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWa
rning: Class Imputer is deprecated; Imputer was deprecated in version 0.20 a
nd will be removed in 0.22. Import impute.SimpleImputer from sklearn instea
d.
  warnings.warn(msg, category=DeprecationWarning)
c:\python37\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWa
rning: Class Imputer is deprecated; Imputer was deprecated in version 0.20 a
nd will be removed in 0.22. Import impute.SimpleImputer from sklearn instea
d.
  warnings.warn(msg, category=DeprecationWarning)
c:\python37\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWa
rning: Class Imputer is deprecated; Imputer was deprecated in version 0.20 a
nd will be removed in 0.22. Import impute.SimpleImputer from sklearn instea
d.
  warnings.warn(msg, category=DeprecationWarning)
c:\python37\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWa
rning: Class Imputer is deprecated; Imputer was deprecated in version 0.20 a
nd will be removed in 0.22. Import impute.SimpleImputer from sklearn instea
d.
```

```
  warnings.warn(msg, category=DeprecationWarning)
c:\python37\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWa
rning: Class Imputer is deprecated; Imputer was deprecated in version 0.20 a
nd will be removed in 0.22. Import impute.SimpleImputer from sklearn instea
d.
  warnings.warn(msg, category=DeprecationWarning)
c:\python37\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWa
rning: Class Imputer is deprecated; Imputer was deprecated in version 0.20 a
nd will be removed in 0.22. Import impute.SimpleImputer from sklearn instea
d.
  warnings.warn(msg, category=DeprecationWarning)
c:\python37\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWa
rning: Class Imputer is deprecated; Imputer was deprecated in version 0.20 a
nd will be removed in 0.22. Import impute.SimpleImputer from sklearn instea
d.
  warnings.warn(msg, category=DeprecationWarning)
c:\python37\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWa
rning: Class Imputer is deprecated; Imputer was deprecated in version 0.20 a
nd will be removed in 0.22. Import impute.SimpleImputer from sklearn instea
d.
  warnings.warn(msg, category=DeprecationWarning)
c:\python37\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWa
rning: Class Imputer is deprecated; Imputer was deprecated in version 0.20 a
nd will be removed in 0.22. Import impute.SimpleImputer from sklearn instea
d.
  warnings.warn(msg, category=DeprecationWarning)
c:\python37\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWa
rning: Class Imputer is deprecated; Imputer was deprecated in version 0.20 a
nd will be removed in 0.22. Import impute.SimpleImputer from sklearn instea
d.
  warnings.warn(msg, category=DeprecationWarning)
c:\python37\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWa
rning: Class Imputer is deprecated; Imputer was deprecated in version 0.20 a
nd will be removed in 0.22. Import impute.SimpleImputer from sklearn instea
d.
  warnings.warn(msg, category=DeprecationWarning)
c:\python37\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWa
rning: Class Imputer is deprecated; Imputer was deprecated in version 0.20 a
nd will be removed in 0.22. Import impute.SimpleImputer from sklearn instea
d.
  warnings.warn(msg, category=DeprecationWarning)
c:\python37\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWa
rning: Class Imputer is deprecated; Imputer was deprecated in version 0.20 a
nd will be removed in 0.22. Import impute.SimpleImputer from sklearn instea
d.
  warnings.warn(msg, category=DeprecationWarning)
c:\python37\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWa
rning: Class Imputer is deprecated; Imputer was deprecated in version 0.20 a
nd will be removed in 0.22. Import impute.SimpleImputer from sklearn instea
d.
  warnings.warn(msg, category=DeprecationWarning)
```

In [30]:

```
#Проверим формат данных и наличие пропусков
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19577 entries, 7 to 73861
Data columns (total 15 columns):
OG_sg            19577 non-null float64
FG_sg            19577 non-null float64
ABV              19577 non-null float64
IBU              19577 non-null float64
Color            19577 non-null float64
SugarScale       19577 non-null float64
BrewMethod       19577 non-null float64
Size(L)          19577 non-null float64
BoilSize         19577 non-null float64
BoilTime         19577 non-null float64
BoilGravity_sg   19577 non-null float64
Efficiency       19577 non-null float64
MashThickness    19577 non-null float64
PitchRate        19577 non-null float64
PrimaryTemp      19577 non-null float64
dtypes: float64(15)
memory usage: 3.0 MB
```

Масштабирование

In [31]:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [32]:

```
sanity_df = pd.DataFrame(X_train, columns = X.columns)
sanity_df.describe().T
```

Out[32]:

	count	mean	std	min	25%	50%	75%	
OG_sg	15661.0	-5.290619e-15	1.000032	-3.797645	-0.569259	-0.117284	0.399257	28.4
FG_sg	15661.0	5.991589e-15	1.000032	-2.830207	-0.637818	-0.039894	0.411515	26.8
ABV	15661.0	3.933597e-16	1.000032	-3.783844	-0.557280	-0.127914	0.427737	27.7
IBU	15661.0	-6.987011e-17	1.000032	-1.099991	-0.514494	-0.238732	0.256329	30.7
Color	15661.0	1.919160e-16	1.000032	-1.139966	-0.642957	-0.389636	0.238368	9.3
SugarScale	15661.0	3.783875e-16	1.000032	-6.743279	0.148296	0.148296	0.148296	0.1
BrewMethod	15661.0	-6.351828e-18	1.000032	-0.602041	-0.602041	-0.602041	0.377346	2.3
Size(L)	15661.0	7.712934e-18	1.000032	-0.241550	-0.145732	-0.135632	-0.118638	32.3
BoilSize	15661.0	-3.493506e-17	1.000032	-0.257687	-0.151818	-0.121569	-0.111486	32.2
BoilTime	15661.0	1.179625e-16	1.000032	-5.158404	-0.303884	-0.303884	-0.303884	14.2
BoilGravity_sg	15661.0	4.422234e-15	1.000032	-2.122995	-0.481865	-0.235696	0.133559	18.3
Efficiency	15661.0	-2.300269e-16	1.000032	-4.645258	-0.070782	0.281101	0.632984	2.3
MashThickness	15661.0	-2.395547e-16	1.000032	-1.181302	-0.283420	-0.283420	-0.238526	45.7
PitchRate	15661.0	3.112396e-16	1.000032	-3.231898	0.118578	0.118578	0.118578	5.7
PrimaryTemp	15661.0	-4.042485e-16	1.000032	-12.077784	-0.293056	0.027355	0.027355	20.8

Обучение базовых моделей

In [33]:

```python
from sklearn.linear_model import SGDClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from time import time

sgd = SGDClassifier()
knc = KNeighborsClassifier()
dtc = DecisionTreeClassifier()
rfc = RandomForestClassifier()
gbc = GradientBoostingClassifier()
models = [sgd, knc, dtc, rfc, gbc]

for model in models:
    start = time()
    model.fit(X_train, y_train)
    stop = time()
    duration = stop-start
    print(type(model).__name__, ': время обучения - ', duration)
```

```
c:\python37\lib\site-packages\sklearn\linear_model\stochastic_gradient.py:14
4: FutureWarning: max_iter and tol parameters have been added in SGDClassifi
er in 0.19. If both are left unset, they default to max_iter=5 and tol=None.
If tol is not None, max_iter defaults to max_iter=1000. From 0.21, default m
ax_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)

SGDClassifier : время обучения -  0.07810401916503906
KNeighborsClassifier : время обучения -  0.12496709823608398
DecisionTreeClassifier : время обучения -  0.22611451148986816

c:\python37\lib\site-packages\sklearn\ensemble\forest.py:248: FutureWarning:
The default value of n_estimators will change from 10 in version 0.20 to 100
in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)

RandomForestClassifier : время обучения -  0.33602404594421387
GradientBoostingClassifier : время обучения -  17.34536123275757
```

Проверка результатов

In [34]:

```
from sklearn.metrics import classification_report

for model in models:
    y_pred = model.predict(X_test)
    print(type(model).__name__, ':\n',
          classification_report(y_test, y_pred,
                                target_names=beer_recipe[beer_recipe['StyleID'].isin(y

```

```
SGDClassifier :
                     precision    recall  f1-score   support

        Imperial IPA       0.15      0.09      0.11       408
              Saison       0.29      0.14      0.19       230
          Blonde Ale       0.30      0.40      0.34       597
  American Brown Ale       0.16      0.08      0.11       455
  American Amber Ale       0.26      0.18      0.21       561
      American Stout       0.76      0.76      0.76       254
        Irish Red Ale       0.34      0.11      0.16       351
American Light Lager       0.54      0.87      0.67       296
   American Pale Ale       0.13      0.29      0.18       241
        American IPA       0.47      0.66      0.55       523

           micro avg       0.34      0.34      0.34      3916
           macro avg       0.34      0.36      0.33      3916
        weighted avg       0.32      0.34      0.32      3916

KNeighborsClassifier :
                     precision    recall  f1-score   support

        Imperial IPA       0.41      0.53      0.46       408
              Saison       0.52      0.53      0.52       230
          Blonde Ale       0.50      0.60      0.55       597
  American Brown Ale       0.39      0.32      0.35       455
  American Amber Ale       0.41      0.48      0.44       561
      American Stout       0.75      0.77      0.76       254
        Irish Red Ale       0.45      0.40      0.43       351
American Light Lager       0.75      0.70      0.73       296
   American Pale Ale       0.43      0.30      0.35       241
        American IPA       0.62      0.48      0.54       523

           micro avg       0.50      0.50      0.50      3916
           macro avg       0.52      0.51      0.51      3916
        weighted avg       0.51      0.50      0.50      3916

DecisionTreeClassifier :
                     precision    recall  f1-score   support

        Imperial IPA       0.42      0.44      0.43       408
              Saison       0.51      0.51      0.51       230
          Blonde Ale       0.49      0.49      0.49       597
  American Brown Ale       0.33      0.31      0.32       455
  American Amber Ale       0.43      0.43      0.43       561
      American Stout       0.76      0.74      0.75       254
        Irish Red Ale       0.46      0.48      0.47       351
American Light Lager       0.68      0.69      0.68       296
   American Pale Ale       0.37      0.40      0.39       241
        American IPA       0.53      0.52      0.52       523
```

```
           micro avg       0.48      0.48      0.48      3916
           macro avg       0.50      0.50      0.50      3916
        weighted avg       0.48      0.48      0.48      3916

RandomForestClassifier :
                       precision    recall  f1-score   support

        Imperial IPA       0.47      0.56      0.51       408
              Saison       0.57      0.60      0.59       230
          Blonde Ale       0.55      0.62      0.59       597
  American Brown Ale       0.44      0.35      0.39       455
  American Amber Ale       0.52      0.55      0.53       561
      American Stout       0.77      0.81      0.79       254
       Irish Red Ale       0.55      0.51      0.53       351
American Light Lager       0.71      0.76      0.74       296
   American Pale Ale       0.49      0.41      0.45       241
        American IPA       0.64      0.58      0.61       523

           micro avg       0.56      0.56      0.56      3916
           macro avg       0.57      0.57      0.57      3916
        weighted avg       0.56      0.56      0.56      3916

GradientBoostingClassifier :
                       precision    recall  f1-score   support

        Imperial IPA       0.55      0.62      0.58       408
              Saison       0.60      0.71      0.65       230
          Blonde Ale       0.60      0.63      0.62       597
  American Brown Ale       0.55      0.35      0.43       455
  American Amber Ale       0.57      0.62      0.60       561
      American Stout       0.76      0.89      0.82       254
       Irish Red Ale       0.64      0.60      0.62       351
American Light Lager       0.74      0.81      0.77       296
   American Pale Ale       0.56      0.51      0.53       241
        American IPA       0.67      0.63      0.65       523

           micro avg       0.62      0.62      0.62      3916
           macro avg       0.62      0.64      0.63      3916
        weighted avg       0.62      0.62      0.61      3916
```

Подбор параметров моделей

In [*]:

```
#Подбор параметров моделей (длительный процесс, поэтому данные о подборе сохраняются в
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import ShuffleSplit

sgd_parameters = {'loss' : ['hinge', 'log', 'modified_huber', 'squared_hinge', 'percep
                  'penalty': ['l1', 'l2'],
                  'alpha': [0.00001, 0.0001, 0.001],
                  'max_iter': [500, 1000, 2000]}
knc_parameters = {'n_neighbors' : [3, 5, 10],
                  'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute']}
dtc_parameters = {'criterion' : ['gini', 'entropy'],
                  'max_leaf_nodes': [5, 10, 15, None],
                  'random_state': [35],
                  'max_depth': [5, 10, 15, None]}
frc_parameters = {'max_leaf_nodes': [5, 10, 15, None],
                  'random_state': [35],
                  'n_estimators': [10, 50, 100],
                  'max_depth': [5, 10, 15, None]}
gbc_parameters = {'loss' : ['deviance', 'exponential'],
                  'learning_rate': [0.01, 0.1, 1],
                  'random_state': [35],
                  'n_estimators': [60, 100, 120],
                  'max_depth': [2, 3, 4]}
parameters = [sgd_parameters, knc_parameters, dtc_parameters, frc_parameters, gbc_para
cv = ShuffleSplit(n_splits = 5, test_size = 0.2)
grid_list=[]
for i in range(len(models)):
    m = models[i]
    p = parameters[i]
    grid = GridSearchCV(m, p, cv = cv)
    grid.fit(X_train, y_train)
    grid_list.append(grid)
```

In [ ]:

```
#Сохранение моделей с подобранными параметрами
from sklearn.externals import joblib
for i in len(models):
    m = models[i]
    grid = grid_list[i]
    joblib.dump(grid.best_estimator_, type(m).__name__ +'_best_params.pkl')
```

In [ ]:

```
#Загрузка моделей с подобранными параметрами
best_models = []
for m in models:
    best_models.append(joblib.load(type(m).__name__ +'_best_params.pkl'))
```

Итоговые результаты

In [ ]:

```
for grid in grid_list:
    y_pred = grid.best_estimator_.predict(X_test)
    print(grid.best_estimator_, ':\n',
          classification_report(y_test, y_pred,
                                target_names=beer_recipe[beer_recipe['StyleID'].isin(y
```

In [ ]:

```
from sklearn.model_selection import learning_curve

def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None, n_jobs=None, train_
    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training examples")
    plt.ylabel("Score")
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    plt.grid()

    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color="g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
             label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
             label="Cross-validation score")

    plt.legend(loc="best")
    return plt
```

In [ ]:

```
title = "Learning Curve"
best_model = 
cv = ShuffleSplit(n_splits=30, test_size=0.2, random_state=35)
plot_learning_curve(estimator, title, X, y, cv=cv, n_jobs=4)

plt.show()
```

In [ ]:

```python
from sklearn.model_selection import validation_curve

def plot_val_curve(X, y, model, param_name, param_range, scorer):
    X, y = X, y
    cv = ShuffleSplit(n_splits=30, test_size=0.2, random_state=35)
    train_scores, test_scores = validation_curve(model, X, y, param_name=param_name, p
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)

    plt.title("Validation Curve")
    plt.xlabel(param_name)
    plt.ylabel("Score")
    plt.ylim(0.0, 1.1)
    lw = 2
    plt.semilogx(param_range, train_scores_mean, label="Training score",
                 color="darkorange", lw=lw)
    plt.fill_between(param_range, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.2,
                     color="darkorange", lw=lw)
    plt.semilogx(param_range, test_scores_mean, label="Cross-validation score",
                 color="navy", lw=lw)
    plt.fill_between(param_range, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.2,
                     color="navy", lw=lw)
    plt.legend(loc="best")
    return plt
```

In [ ]:

```python
from sklearn.metrics import make_scorer, accuracy_score

param_range = list(range(2, 10, 1))
param_name = "max_depth"
acc_scorer = make_scorer(accuracy_score)
val_curve(X, y, best_model, param_name, param_range, acc_scorer)

plt.show()
```