Задача:

1. Провести исследование данных.
2. Рассчитать регулярный прогноз (без промо) на 45, 46, 47, 48 календарные недели.
3. Оценка точности MAPE понедельно.

In [1]:

```
import pandas as pd
import numpy as np

import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

import os
```

In [2]:

```
# Загружаем файл
df_csv = pd.read_csv('005_example.csv', sep = ';')
print(len(df_csv))
df_csv.head()
```

```
529577
```

Out[2]:

	year	week	date_id	good_id	ship_store_id	price	promo	sale	promo_price	holiday
0	2018	38	16.09.2018	149045	40	699	0	1	699	0
1	2018	11	17.03.2018	149045	40	599	0	0	599	0
2	2018	12	23.03.2018	149045	40	599	0	0	599	0
3	2017	30	25.07.2017	149045	40	599	0	2	599	0
4	2018	40	04.10.2018	173544	13	1399	0	0	1399	0

1. date_id - дата
2. good_id - код товара
3. ship_store_id - код магазина
4. price - цена
5. promo - флаг промо
6. sale - продажи
7. promo_price - промо цена
8. holiday - флаг праздничного дня
9. owner_id - код региона

In [3]:

```
df = df_csv[['year', 'week', 'good_id', 'ship_store_id', 'owner_id', 'sale']]
print(len(df))
df.head()
```

529577

Out[3]:

	year	week	good_id	ship_store_id	owner_id	sale
0	2018	38	149045	40	2	1
1	2018	11	149045	40	2	0
2	2018	12	149045	40	2	0
3	2017	30	149045	40	2	2
4	2018	40	173544	13	2	0

In [4]:

```
# Создадим отдельный список для продаж без промо за 45-48 недели
df_no_promo = df_csv[(df_csv.promo == 0)&(df_csv.week.isin([45, 46, 47, 48]))][['year', 'ow
                                                                                'good_id',
print(len(df_no_promo))
df_no_promo.head()
```

48398

Out[4]:

	year	owner_id	ship_store_id	good_id	price
16	2019	2	63	173541	1299
17	2019	2	63	173541	1299
18	2018	2	63	173541	1299
20	2018	2	63	173541	1299
23	2018	2	63	173541	1299

In [5]:

```
# Если за 45-48 недели у товара в одном магазине была разная цена - усредним ее
df_no_promo = df_no_promo.groupby(['year', 'owner_id', 'ship_store_id', 'good_id'])['price'
print(len(df_no_promo))
df_no_promo.head()
```

1824

Out[5]:

	year	owner_id	ship_store_id	good_id	price
0	2017	2	2	19594	4999.0
1	2017	2	2	29712	2999.0
2	2017	2	2	96473	1999.0
3	2017	2	2	96474	2599.0
4	2017	2	2	96475	2599.0

In [6]:

```
# Сгруппируем продажи по неделям. После группировки sale - количество продаж товара за неде
df_prod = df.groupby(['year', 'owner_id', 'ship_store_id', 'good_id',
                      'week'])['sale'].apply(sum).reset_index()
print(len(df_prod))
df_prod.head()
```

77850

Out[6]:

	year	owner_id	ship_store_id	good_id	week	sale
0	2017	2	2	19594	5	3
1	2017	2	2	19594	6	0
2	2017	2	2	19594	7	0
3	2017	2	2	19594	8	0
4	2017	2	2	19594	9	3

In [7]:

```
# Теперь сделаем, так, что бы в sale были все продажи товара в магазине за год
df_group_sale = df_prod.groupby(['year', 'owner_id', 'ship_store_id',
                                 'good_id'])['sale'].apply(lambda s: s.values).reset_index(
print(len(df_group_sale))
df_group_sale.head()
```

1879

Out[7]:

	year	owner_id	ship_store_id	good_id	sale
0	2017	2	2	19594	[3, 0, 0, 0, 3, 1, 1, 4, 1, 0, 3, 1, 1, 2, 0, ...
1	2017	2	2	29712	[2, 5, 7, 6, 2, 2, 9, 3, 5, 1, 2, 3, 6, 6, 1, ...
2	2017	2	2	96473	[0, 0, 0, 3, 0, 1, 0, 1, 4, 0, 1, 0, 2, 0, 0, ...
3	2017	2	2	96474	[0, 2, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, ...
4	2017	2	2	96475	[1, 1, 1, 6, 1, 4, 7, 8, 4, 1, 2, 2, 4, 3, 4, ...

In [8]:

```
# В week будут храниться все номера недель в которые были продажи (в году)
df_group_week = df_prod.groupby(['year', 'owner_id', 'ship_store_id',
                                 'good_id'])['week'].apply(lambda s: s.values).reset_index(
print(len(df_group_week))
df_group_week.head()
```

1879

Out[8]:

	year	owner_id	ship_store_id	good_id	week
0	2017	2	2	19594	[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17...
1	2017	2	2	29712	[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17...
2	2017	2	2	96473	[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17...
3	2017	2	2	96474	[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17...
4	2017	2	2	96475	[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17...

In [9]:

```
df_group_s_w = df_group_sale[['year', 'owner_id', 'ship_store_id', 'good_id', 'sale']]
df_group_s_w['week'] = df_group_week['week']

df_group_s_w.head()
```

Out[9]:

	year	owner_id	ship_store_id	good_id	sale	week
0	2017	2	2	19594	[3, 0, 0, 0, 3, 1, 1, 4, 1, 0, 3, 1, 1, 2, 0, ...	[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17...
1	2017	2	2	29712	[2, 5, 7, 6, 2, 2, 9, 3, 5, 1, 2, 3, 6, 6, 1, ...	[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17...
2	2017	2	2	96473	[0, 0, 0, 3, 0, 1, 0, 1, 4, 0, 1, 0, 2, 0, 0, ...	[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17...
3	2017	2	2	96474	[0, 2, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, ...	[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17...
4	2017	2	2	96475	[1, 1, 1, 6, 1, 4, 7, 8, 4, 1, 2, 2, 4, 3, 4, ...	[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17...

In [10]:

```
# функция "weeks_of_sales" создает массив длиной 53, в котором индекс в массиве соответству
# значение соотвтествует количеству проданного товара за эту неделю
# 53 - количество недель в году
def weeks_of_sales (sales, weeks):
    sales53 = [0] * 53

    i = 0
    while i < len(weeks):
        sales53[weeks[i]-1] = sales[i]
        i += 1

    return sales53
```

In [11]:

```
# Проверим функцию
print(weeks_of_sales(df_group_s_w.sale[1], df_group_s_w.week[1]))
```

```
[0, 0, 0, 0, 2, 5, 7, 6, 2, 2, 9, 3, 5, 1, 2, 3, 6, 6, 1, 1, 5, 2, 8, 8, 3,
5, 9, 15, 4, 3, 7, 2, 7, 8, 7, 6, 6, 9, 2, 2, 5, 3, 4, 2, 4, 5, 2, 1, 0, 3,
10, 6, 2]
```

In [12]:

```
# С помощью функции "weeks_of_sales" сделаем, заменим значения в столбце "sale"
i = 0
while i < len(df_group_s_w):
    df_group_s_w.sale[i] = weeks_of_sales (df_group_s_w.sale[i], df_group_s_w.week[i])
    i += 1
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (http://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy)
  after removing the cwd from sys.path.
```

In [13]:

```
# Столбец "week" больше не нужен
del df_group_s_w['week']
df_group_s_w.head()
```

Out[13]:

	year	owner_id	ship_store_id	good_id	sale
0	2017	2	2	19594	[0, 0, 0, 0, 3, 0, 0, 0, 3, 1, 1, 4, 1, 0, 3, ...
1	2017	2	2	29712	[0, 0, 0, 0, 2, 5, 7, 6, 2, 2, 9, 3, 5, 1, 2, ...
2	2017	2	2	96473	[0, 0, 0, 0, 0, 0, 0, 3, 0, 1, 0, 1, 4, 0, 1, ...
3	2017	2	2	96474	[0, 0, 0, 0, 0, 2, 0, 1, 1, 1, 0, 0, 0, 0, 0, ...
4	2017	2	2	96475	[0, 0, 0, 0, 1, 1, 1, 6, 1, 4, 7, 8, 4, 1, 2, ...

In [14]:

```
# Оставим только товары, у которых на 45-48 неделях не было промо, и добавим цены товаров
df_group_s_w = df_group_s_w.merge(df_no_promo, on = ['year', 'owner_id', 'ship_store_id', '
```

Создание новых признаков

Прогноз будет осуществляться на 45-48 недели, следовательно, продажи за 49-53 тоже предполагаем неизвестными.

Будут созданы следующие фичи:

1. 'sale_44w' - продажи товара за первые 44 недели года (по магазину)
2. 'mean_44w' - среднее количество продаж в неделю за первые 44 недели года (по магазину)
3. 'w44' - продажи товара за 44 неделю (по магазину)
4. 'sale_44w_all' - среднее количество продаж по всем магазинам за первые 44 недели в году
5. 'sale_44w_region' - среднее количество продаж по магазинам внутри региона за первые 44 недели в году
6. 'mean_w44_all' - среднее количество продаж за 44 неделю по всем магазинам
7. 'mean_w44_region' - среднее количество продаж за 44 неделю по магазинам внутри региона

Значения 'sale_44w' и 'mean_44w' зависимы друг от друга, при обучении будет использоваться только 'mean_44w', 'sale_44w' пока нужен для простого вычисления других значений.

In [15]:

```
# Создадим пустые столбцы
df_group_s_w['sale_44w'] = np.nan
df_group_s_w['mean_44w'] = np.nan
df_group_s_w['w44'] = np.nan

df_group_s_w.head()
```

Out[15]:

	year	owner_id	ship_store_id	good_id	sale	price	sale_44w	mean_44w	w44
0	2017	2	2	19594	[0, 0, 0, 0, 3, 0, 0, 0, 3, 1, 1, 4, 1, 0, 3, ...	4999.0	NaN	NaN	NaN
1	2017	2	2	29712	[0, 0, 0, 0, 2, 5, 7, 6, 2, 2, 9, 3, 5, 1, 2, ...	2999.0	NaN	NaN	NaN
2	2017	2	2	96473	[0, 0, 0, 0, 0, 0, 0, 3, 0, 1, 0, 1, 4, 0, 1, ...	1999.0	NaN	NaN	NaN
3	2017	2	2	96474	[0, 0, 0, 0, 0, 2, 0, 1, 1, 1, 0, 0, 0, 0, 0, ...	2599.0	NaN	NaN	NaN
4	2017	2	2	96475	[0, 0, 0, 0, 1, 1, 1, 6, 1, 4, 7, 8, 4, 1, 2, ...	2599.0	NaN	NaN	NaN

In [ ]:

```
# Заполним пустые столбцы
i = 0
while i < len(df_group_s_w):
    df_group_s_w.sale_44w[i] = np.sum(df_group_s_w.sale[i][0:44])
    df_group_s_w.mean_44w[i] = np.mean(df_group_s_w.sale[i][0:44])
    df_group_s_w.w44[i] = df_group_s_w.sale[i][43]
    i += 1
```

In [17]:

```
df_group_s_w.head()
```

Out[17]:

	year	owner_id	ship_store_id	good_id	sale	price	sale_44w	mean_44w	w44
0	2017	2	2	19594	[0, 0, 0, 0, 3, 0, 0, 0, 3, 1, 1, 4, 1, 0, 3, ...	4999.0	63.0	1.431818	3.0
1	2017	2	2	29712	[0, 0, 0, 0, 2, 5, 7, 6, 2, 2, 9, 3, 5, 1, 2, ...	2999.0	193.0	4.386364	2.0
2	2017	2	2	96473	[0, 0, 0, 0, 0, 0, 0, 3, 0, 1, 0, 1, 4, 0, 1, ...	1999.0	70.0	1.590909	0.0
3	2017	2	2	96474	[0, 0, 0, 0, 0, 2, 0, 1, 1, 1, 0, 0, 0, 0, 0, ...	2599.0	37.0	0.840909	0.0
4	2017	2	2	96475	[0, 0, 0, 0, 1, 1, 1, 6, 1, 4, 7, 8, 4, 1, 2, ...	2599.0	150.0	3.409091	6.0

In [18]:

```
# средние продажи товара по всем магазинам за 44 недели
df_sale_44w_all = df_group_s_w.groupby(['year', 'good_id'])['sale_44w'].apply(np.mean).rese
df_sale_44w_all.rename(columns={'sale_44w': 'sale_44w_all'}, inplace=True)
df_sale_44w_all.head()
```

Out[18]:

	year	good_id	sale_44w_all
0	2017	19594	55.083333
1	2017	19595	8.000000
2	2017	29712	131.090909
3	2017	39176	111.000000
4	2017	85612	49.625000

In [19]:

```
# средние продажи товара по всем магазинам в регионе за 44 недели
df_sale_44w_region = df_group_s_w.groupby(['year', 'owner_id', 'good_id'])['sale_44w'].appl
df_sale_44w_region.rename(columns={'sale_44w': 'sale_44w_region'}, inplace=True)
df_sale_44w_region.head()
```

Out[19]:

	year	owner_id	good_id	sale_44w_region
0	2017	2	19594	59.090909
1	2017	2	19595	8.000000
2	2017	2	29712	158.461538
3	2017	2	39176	111.000000
4	2017	2	85612	78.500000

In [20]:

```
# средние продажи товара по всем магазинам за 44 неделю
df_mean_w44_all = df_group_s_w.groupby(['year', 'good_id'])['w44'].apply(np.mean).reset_ind
df_mean_w44_all.rename(columns={'w44': 'mean_w44_all'}, inplace=True)
df_mean_w44_all.head()
```

Out[20]:

	year	good_id	mean_w44_all
0	2017	19594	1.916667
1	2017	19595	0.000000
2	2017	29712	2.954545
3	2017	39176	3.500000
4	2017	85612	2.000000

In [21]:

```
# средние продажи товара по всем магазинам в регионе за 44 неделю
df_mean_w44_region = df_group_s_w.groupby(['year', 'owner_id', 'good_id'])['w44'].apply(np.
df_mean_w44_region.rename(columns={'w44': 'mean_w44_region'}, inplace=True)
df_mean_w44_region.head()
```

Out[21]:

	year	owner_id	good_id	mean_w44_region
0	2017	2	19594	2.000000
1	2017	2	19595	0.000000
2	2017	2	29712	3.307692
3	2017	2	39176	3.500000
4	2017	2	85612	3.000000

In [22]:

```
# Добавим вычисленные фитчи к исходной таблице
df_group_s_w = df_group_s_w.merge(df_sale_44w_all, on = ['year', 'good_id'], how = 'left')
df_group_s_w = df_group_s_w.merge(df_sale_44w_region, on = ['year', 'owner_id', 'good_id'],
df_group_s_w = df_group_s_w.merge(df_mean_w44_all, on = ['year', 'good_id'], how = 'left')
df_group_s_w = df_group_s_w.merge(df_mean_w44_region, on = ['year', 'owner_id', 'good_id'],
```

In [23]:

```
# Добавим столбцы со значениями продаж за 45-48 недели
df_group_s_w['w45'] = np.nan
df_group_s_w['w46'] = np.nan
df_group_s_w['w47'] = np.nan
df_group_s_w['w48'] = np.nan

i = 0
while i < len(df_group_s_w):
    df_group_s_w.w45[i] = df_group_s_w.sale[i][44]
    df_group_s_w.w46[i] = df_group_s_w.sale[i][45]
    df_group_s_w.w47[i] = df_group_s_w.sale[i][46]
    df_group_s_w.w48[i] = df_group_s_w.sale[i][47]
    i += 1

del df_group_s_w['sale']

df_group_s_w.head()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:9: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (http://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy)
  if __name__ == '__main__':
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:10: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (http://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy)
  # Remove the CWD from sys.path while we load stuff.
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:11: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (http://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy)
  # This is added back by InteractiveShellApp.init_path()
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:12: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (http://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy)
  if sys.path[0] == '':
```

Out[23]:

	year	owner_id	ship_store_id	good_id	price	sale_44w	mean_44w	w44	sale_44w_all

	year	owner_id	ship_store_id	good_id	price	sale_44w	mean_44w	w44	sale_44w_all
0	2017	2	2	19594	4999.0	63.0	1.431818	3.0	55.083333
1	2017	2	2	29712	2999.0	193.0	4.386364	2.0	131.090909
2	2017	2	2	96473	1999.0	70.0	1.590909	0.0	78.958333
3	2017	2	2	96474	2599.0	37.0	0.840909	0.0	58.055556
4	2017	2	2	96475	2599.0	150.0	3.409091	6.0	128.666667

In [24]:

```
df_group_s_w.describe()
```

Out[24]:

	year	owner_id	ship_store_id	good_id	price	sale_44w	m
count	1824.000000	1824.000000	1824.000000	1824.000000	1824.000000	1824.000000	182
mean	2018.311404	5.084978	50.852522	134984.467654	1708.859765	183.188048	
std	0.770807	4.942224	39.792242	40335.040712	1597.358739	278.755128	
min	2017.000000	2.000000	2.000000	19594.000000	119.000000	0.000000	
25%	2018.000000	2.000000	13.000000	123647.000000	791.500000	46.750000	
50%	2019.000000	2.000000	45.000000	147603.000000	1199.000000	98.500000	
75%	2019.000000	6.000000	69.000000	173540.000000	2199.000000	197.250000	
max	2019.000000	21.000000	150.000000	185371.000000	10999.000000	2685.000000	6

Создание обучающего и тестового набора

In [25]:

```
train = df_group_s_w[df_group_s_w.year != 2019][['mean_44w','w44','sale_44w_region','mean_w
                                                 'w45','w46','w47','w48']].sample(frac=1).r
test = df_group_s_w[df_group_s_w.year == 2019][['mean_44w','w44', 'sale_44w_region','mean_w
                                               'w45','w46','w47','w48']].sample(frac=1).re
```

In [26]:

```
#train = df_group_s_w[df_group_s_w.year != 2019].iloc[:, 4:].sample(frac=1).reset_index(dro
#test = df_group_s_w[df_group_s_w.year == 2019].iloc[:, 4:].sample(frac=1).reset_index(drop
```

In [27]:

```
train.head()
```

Out[27]:

	mean_44w	w44	sale_44w_region	mean_w44_all	mean_w44_region	w45	w46	w47	w48
0	5.090909	16.0	95.857143	6.111111	5.142857	17.0	19.0	13.0	14.0
1	2.295455	0.0	101.000000	1.666667	0.000000	0.0	1.0	1.0	1.0
2	5.681818	5.0	199.937500	13.956522	12.875000	9.0	8.0	13.0	16.0
3	0.000000	0.0	0.266667	0.088235	0.200000	0.0	7.0	7.0	9.0
4	0.272727	7.0	12.000000	2.407407	7.000000	2.0	0.0	0.0	0.0

In [28]:

```
test.head()
```

Out[28]:

	mean_44w	w44	sale_44w_region	mean_w44_all	mean_w44_region	w45	w46	w47	w48
0	4.045455	10.0	84.800000	5.189189	4.4	13.0	12.0	8.0	8.0
1	2.613636	2.0	115.000000	2.266667	2.0	3.0	6.0	3.0	2.0
2	3.295455	2.0	89.166667	2.266667	2.5	3.0	2.0	4.0	1.0
3	4.295455	6.0	189.000000	17.948718	6.0	48.0	16.0	14.0	2.0
4	5.704545	13.0	251.000000	9.326087	13.0	0.0	3.0	2.0	4.0

In [29]:

```
x_train = train.iloc[:, :-4]
y_train = train.iloc[:, -4:]
```

In [30]:

```
x_test = test.iloc[:, :-4]
y_test = test.iloc[:, -4:]
```

Линейная регрессия

In [31]:

```
from sklearn.linear_model import LinearRegression
```

In [32]:

```
model_lr = LinearRegression()
```

In [33]:

```
model_lr.fit(x_train, y_train)
```

Out[33]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fal
se)
```

In [34]:

```
predicted_lr = model_lr.predict(x_test)
print(predicted_lr)
```

```
[[7.12159802 5.64145398 6.83570703 7.68992217]
 [2.43616078 3.30721891 5.04827476 5.7619938 ]
 [2.39579776 3.77575849 5.7382371  6.4864276 ]
 ...
 [2.90812892 3.28115153 4.98250858 5.5220058 ]
 [3.89985472 4.6408563  6.32791849 7.35568876]
 [1.63444855 3.19982778 5.11801442 5.76656545]]
```

In [35]:

```
# Функция "rounding" округляет значения в массиве до целого и заменяет отрицательные значен
def rounding(m):
    # округлим значения до целого
    m = np.around(m, decimals=0)
    # заменим отрицательные значения на 0, если они есть
    m[m < 0] = 0
    return m
```

In [36]:

```
def mape(predict, fact):
    mae = 0
    n = 0
    eps = 0.000001

    l = len(predict)
    i = 0
    while i < l:
        m = len(predict[i])
        j = 0
        while j < m:
            mae += abs(predict[i][j] - fact[i][j])/(max(predict[i][j], fact[i][j]) + eps)
            n += 1
            j += 1
        i += 1
    return mae/n
```

In [37]:

```
mape_lr = mape(rounding(predicted_lr), y_test.values)
print("Точность:", 1 - mape_lr)
```

```
Точность: 0.43640098962749796
```

XGBoost

In [38]:

```
import xgboost as xgb
from sklearn.model_selection import GridSearchCV
```

In [39]:

```
# функция, возвращающая лучшую модель из GridSearchCV
def best_xgb(x, y):
    params = {'min_child_weight':[4,5], 'gamma':[i/10.0 for i in range(3,6)], 'subsample':[
              'colsample_bytree':[i/10.0 for i in range(6,11)], 'max_depth': [2,3,4]}

    model = xgb.XGBRegressor()
    grid = GridSearchCV(model, params)
    grid.fit(x, y)

    return grid.best_estimator_
```

In [ ]:

```
# выберем лучшие модели для каждой прогнозируемой недели
model_xgb45 = best_xgb(x_train, y_train.iloc[:, 0])
model_xgb46 = best_xgb(x_train, y_train.iloc[:, 1])
model_xgb47 = best_xgb(x_train, y_train.iloc[:, 2])
model_xgb48 = best_xgb(x_train, y_train.iloc[:, 3])
```

In [41]:

```
# сделаем прогноз
predicted_xgb45 = model_xgb45.predict(x_test)
predicted_xgb46 = model_xgb45.predict(x_test)
predicted_xgb47 = model_xgb45.predict(x_test)
predicted_xgb48 = model_xgb45.predict(x_test)
```

In [42]:

```
#объединим все спронозированные значения в один массив
predicted_xgb = np.column_stack((predicted_xgb45, predicted_xgb46, predicted_xgb47, predict
print(predicted_xgb[:10])
```

```
[[ 6.0300364  6.0300364  6.0300364  6.0300364]
 [ 2.219733   2.219733   2.219733   2.219733 ]
 [ 2.9897344  2.9897344  2.9897344  2.9897344]
 [11.480563  11.480563  11.480563  11.480563 ]
 [10.261178  10.261178  10.261178  10.261178 ]
 [ 7.0328746  7.0328746  7.0328746  7.0328746]
 [ 8.504426   8.504426   8.504426   8.504426 ]
 [ 9.642557   9.642557   9.642557   9.642557 ]
 [ 9.852744   9.852744   9.852744   9.852744 ]
 [ 3.7435224  3.7435224  3.7435224  3.7435224]]
```

In [43]:

```
mape_xgb = mape(rounding(predicted_xgb), y_test.values)
print("Точность:", 1 - mape_xgb)
```

```
Точность: 0.4407295487079046
```

LightGBM

In [44]:

```
import lightgbm as lgbm
```

In [45]:

```
# функция, возвращающая лучшую модель из GridSearchCV
def best_lgbm(x, y):
    params = {'n_estimators':range(50, 600, 50), 'num_leaves':range(8, 80, 8)}

    model = lgbm.LGBMRegressor()
    grid = GridSearchCV(model, params)
    grid.fit(x, y)

    return grid.best_estimator_
```

In [ ]:

```
# выберем лучшие модели для каждой прогнозируемой недели
model_lgbm45 = best_lgbm(x_train, y_train.iloc[:, 0])
model_lgbm46 = best_lgbm(x_train, y_train.iloc[:, 1])
model_lgbm47 = best_lgbm(x_train, y_train.iloc[:, 2])
model_lgbm48 = best_lgbm(x_train, y_train.iloc[:, 3])
```

In [47]:

```
# сделаем прогноз
predicted_lgbm45 = model_lgbm45.predict(x_test)
predicted_lgbm46 = model_lgbm45.predict(x_test)
predicted_lgbm47 = model_lgbm45.predict(x_test)
predicted_lgbm48 = model_lgbm45.predict(x_test)
```

In [48]:

```
#объединим все спронозированные значения в один массив
predicted_lgbm = np.column_stack((predicted_lgbm45, predicted_lgbm46, predicted_lgbm47, pre
print(predicted_lgbm[:10])
```

```
[[ 5.97989262  5.97989262  5.97989262  5.97989262]
 [ 2.86441205  2.86441205  2.86441205  2.86441205]
 [ 3.34463651  3.34463651  3.34463651  3.34463651]
 [13.14794339 13.14794339 13.14794339 13.14794339]
 [ 9.9728818   9.9728818   9.9728818   9.9728818 ]
 [ 5.77873795  5.77873795  5.77873795  5.77873795]
 [ 1.2334733   1.2334733   1.2334733   1.2334733 ]
 [ 8.71866948  8.71866948  8.71866948  8.71866948]
 [ 9.31894317  9.31894317  9.31894317  9.31894317]
 [ 3.46810568  3.46810568  3.46810568  3.46810568]]
```

In [49]:

```
mape_lgbm = mape(rounding(predicted_lgbm), y_test.values)
print("Точность:", 1 - mape_lgbm)
```

```
Точность: 0.43900713111428546
```

Ансамбль моделей

In [50]:

```
k_lr = 0
k_xgb = 0
k_lgbm = 0
mp = 1

i = j = k = 0
for i  in range(0, 100, 5):
    for j in range(0, 100, 5):
        for k in range(0, 100, 5):
            pr = (predicted_lr * i + predicted_xgb * j + predicted_lgbm * k) / 100
            m = mape(rounding(pr), y_test.values)
            if m < mp:
                mp = m
                k_lr = i / 100
                k_xgb = j / 100
                k_lgbm = k / 100
```

In [51]:

```
print("Коэффициенты:", k_lr, k_xgb, k_lgbm)
print("Точность:", 1 - mp)
```

```
Коэффициенты: 0.55 0.1 0.15
Точность: 0.45280465572171014
```

In [ ]:

```

```