# **Введение**

Вы приступаете к изучению второго модуля, посвященного временным рядам и их анализу.

В этом модуле вы:

* закрепите на практике навыки прошлого модуля по визуализации и подготовке данных, моделированию временных рядов;
* познакомитесь детальнее с задачей прогноза временного ряда;
* узнаете классификацию методов;
* изучите различные методы прогнозирования временных рядов, такие как наивное предсказание, скользящее среднее, взвешенное скользящее среднее, экспоненциальное сглаживание, двойное экспоненциальное сглаживание и тройное экспоненциальное сглаживание;
* узнаете, что такое регрессионный анализ, познакомитесь с основными методами регрессии, начиная с линейной и полиномиальной регрессии в контексте временных рядов;
* научитесь реализовывать предсказания на практике с помощью библиотеки SCIKIT-TIME (SKTIME).

После прохождения модуля вы сможете точно выбирать подход, наиболее подходящий для вашей задачи и данных и обеспечивающий наилучшее качество прогноза.

Со списком дополнительной литературы и словарем модуля вы можете ознакомиться в последних юнитах.

Итак, приступим!

# Юнит 10. Практика

Начнем этот модуль с закрепления на практике знаний, которые вы получили благодаря изучению прошлого модуля. Практическое применение полученных знаний играет важную роль в обучении и помогает вам лучше усвоить материал.

В данном модуле мы предлагаем вам ознакомиться с тетрадями в Google Colab, которые позволят применить полученные знания в области визуализации и предварительной работы с данными, а также в моделировании временных рядов.

**Визуализация и предварительная работа с данными**

Первая тетрадь посвящена визуализации и предварительной обработке данных — важным этапам в анализе данных. Эти процессы позволяют нам получить представление о структуре данных, выявить возможные аномалии и подготовить данные для дальнейшего анализа.

Предлагаем вам ознакомиться с данной тетрадью в Google Colab: <https://colab.research.google.com/drive/1iXuR09afBT16T1RqV-0G9HpMRIT71zly>

Визуализация временных рядов

Проведем импорт необходимых библиотек

try:  
 import pandas as pd   
except:  
 !pip install pandas  
finally:  
 import pandas as pd

import matplotlib.pyplot as plt  
import matplotlib.dates as mdates#Date Parser  
  
import seaborn as sns  
sns.set\_style('white')  
sns.set(rc={'figure.figsize':(11, 4)})

Загрузка данных в Pandas

урок вдохновлён данным репозиторием <https://github.com/jenfly/opsd>

В качестве набор данных для практики рассмотрим часть набор данных [открытые данные энергетических систем](https://open-power-system-data.org/). Мы будем работать с данным относящимися ко временным рядам (<https://data.open-power-system-data.org/time_series/>). По приведенной ссылке можно найти описание набора данных. Также набор можно найти на странице официального репозитория: <https://github.com/Open-Power-System-Data/time_series>.

Для начала давайте попробуем загрузить последнюю версию набора данных.

# Download hourly data from OPSD website  
url = 'https://data.open-power-system-data.org/time\_series/2020-10-06/'  
datafile = url + 'time\_series\_60min\_singleindex.csv'  
df\_all = pd.read\_csv(datafile, index\_col='utc\_timestamp', parse\_dates=True, low\_memory=False)  
df\_all.head()

[5 rows x 299 columns]

Из всего набора выделим только данные относящиеся к целевой стране. В качестве примера рассмотрим германию. В качестве периода анализа возьмём период с 2015 по 2019 годы.

def extract\_country(df\_all, country\_code, year\_min=None, year\_max=None):  
 """Extract data for a single country"""  
   
 # List of columns to extract  
 columns = [col for col in df\_all.columns if col.startswith(country\_code)]  
   
 # Extract columns and remove country codes from column labels  
 columns\_map = {col : col[3:] for col in columns}  
 df\_out = df\_all[columns].rename(columns=columns\_map)  
   
 # Exclude years outside of specified range, if any  
 if year\_min is not None:  
 df\_out = df\_out[df\_out.index.year >= year\_min]  
 if year\_max is not None:  
 df\_out = df\_out[df\_out.index.year <= year\_max]  
   
 return df\_out  
df\_hrly = extract\_country(df\_all, country\_code='DE', year\_min=2015, year\_max=2019)  
df\_hrly

[43824 rows x 41 columns]

Нам понадобятся не все колонки, поэтому выделим необходимые, кроме того приведем колонки к более интерпретируемому виду

def transform\_dataframe(df, cols\_map):  
 # Rename columns for convenience  
 df = df[list(cols\_map.keys())].rename(columns=cols\_map)  
 # Convert from MW to GW  
 df = df / 1000  
 df = df.resample('D').sum(min\_count=24)  
 df = df.rename\_axis('Date')  
 df.index = df.index.strftime('%Y-%m-%d')  
 return df  
  
cols\_map = {'load\_actual\_entsoe\_transparency' : 'Consumption',  
 'wind\_generation\_actual' : 'Wind',  
 'solar\_generation\_actual' : 'Solar'}  
df\_daily = transform\_dataframe(df\_hrly, cols\_map)  
  
# Compute wind + solar generation  
df\_daily['Wind+Solar'] = df\_daily[['Wind', 'Solar']].sum(axis=1, skipna=False)  
df\_daily.to\_csv('de\_data.csv')  
df\_daily.head()

Consumption Wind Solar Wind+Solar  
Date   
2015-01-01 1088.317 325.165 NaN NaN  
2015-01-02 1246.588 603.554 7.757 611.311  
2015-01-03 1117.554 462.955 7.237 470.192  
2015-01-04 1081.980 385.023 19.982 405.005  
2015-01-05 1325.920 216.540 26.522 243.062

Теперь набор представляет собой ежедневное потребление электричества (в гига-Ваттах в час) в Германии. Набор включает следующие временные ряды в виде колонок:

* Date — дата в формате гггг-мм-дд;
* Consumption — Общее потребление, ГВт/ч;
* Wind — Потребление ветряной энергии, ГВт/ч;
* Solar — Потребление солнечной энергии, ГВт/ч;
* Wind+Solar — Потребление энергии из альтернативных источников, ГВт/ч.

Проведем анализ сформированного набора данных

path\_ts = 'de\_data.csv'  
  
df = pd.read\_csv(path\_ts)  
  
df.sample(5, random\_state=0)

Date Consumption Wind Solar Wind+Solar  
793 2017-03-04 1224.314 253.250 99.948 353.198  
789 2017-02-28 1509.014 602.221 49.109 651.330  
118 2015-04-29 1365.365 153.288 188.720 342.008  
318 2015-11-15 1176.237 530.791 21.279 552.070  
891 2017-06-10 1132.742 89.475 214.438 303.913

Размер набора данных

print(df.shape)

(1826, 5)

Информация о колонках в наборе

df.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 1826 entries, 0 to 1825  
Data columns (total 5 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 Date 1826 non-null object   
 1 Consumption 1826 non-null float64  
 2 Wind 1819 non-null float64  
 3 Solar 1818 non-null float64  
 4 Wind+Solar 1817 non-null float64  
dtypes: float64(4), object(1)  
memory usage: 71.5+ KB

Описание суммарных статистик данных

df.describe()

Consumption Wind Solar Wind+Solar  
count 1826.000000 1819.000000 1818.000000 1817.000000  
mean 1340.623081 267.842699 103.386051 371.257950  
std 156.544793 192.922094 66.797470 178.690531  
min 934.864000 16.482000 4.989000 27.529000  
25% 1213.262000 120.906500 38.559500 242.188000  
50% 1374.256000 211.989000 101.705500 338.443000  
75% 1455.366500 364.868000 159.751250 470.921000  
max 1636.406000 998.899000 264.538000 1034.494000

типы данных

df.dtypes

Date object  
Consumption float64  
Wind float64  
Solar float64  
Wind+Solar float64  
dtype: object

Введение индексов-дат

df.Date = pd.to\_datetime(df.Date)  
df.set\_index('Date', inplace=True)  
df.sample(15, random\_state=0)

Consumption Wind Solar Wind+Solar  
Date   
2017-03-04 1224.314 253.250 99.948 353.198  
2017-02-28 1509.014 602.221 49.109 651.330  
2015-04-29 1365.365 153.288 188.720 342.008  
2015-07-11 1122.407 50.020 193.951 243.971  
2017-06-23 1407.411 427.695 169.412 597.107

df.info()

<class 'pandas.core.frame.DataFrame'>  
DatetimeIndex: 1826 entries, 2015-01-01 to 2019-12-31  
Data columns (total 4 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 Consumption 1826 non-null float64  
 1 Wind 1819 non-null float64  
 2 Solar 1818 non-null float64  
 3 Wind+Solar 1817 non-null float64  
dtypes: float64(4)  
memory usage: 71.3 KB

Отметим, что на самом деле можно было сразу загрузить данные в таком виде, чтобы индексы были датами

df = pd.read\_csv(path\_ts, parse\_dates=['Date'], index\_col="Date")  
df.head()

Consumption Wind Solar Wind+Solar  
Date   
2015-01-01 1088.317 325.165 NaN NaN  
2015-01-02 1246.588 603.554 7.757 611.311  
2015-01-03 1117.554 462.955 7.237 470.192  
2015-01-04 1081.980 385.023 19.982 405.005  
2015-01-05 1325.920 216.540 26.522 243.062

Данные обработанные в форме дат DateTimeIndex позволяют работать с индексом как с датой

print(df.index.day)  
print(df.index.weekday)  
print(df.index.year)

Int64Index([ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,  
 ...  
 22, 23, 24, 25, 26, 27, 28, 29, 30, 31],  
 dtype='int64', name='Date', length=1826)  
Int64Index([3, 4, 5, 6, 0, 1, 2, 3, 4, 5,  
 ...  
 6, 0, 1, 2, 3, 4, 5, 6, 0, 1],  
 dtype='int64', name='Date', length=1826)  
Int64Index([2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015,  
 ...  
 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019],  
 dtype='int64', name='Date', length=1826)

Теперь мы можем обращаться к данным по дате

df.loc['2017-08-10']

Consumption 1342.278  
Wind 100.274  
Solar 71.162  
Wind+Solar 171.436  
Name: 2017-08-10 00:00:00, dtype: float64

или по выборке дат

df.loc['2014-12-31':'2015-01-22']

Consumption Wind Solar Wind+Solar  
Date   
2015-01-01 1088.317 325.165 NaN NaN  
2015-01-02 1246.588 603.554 7.757 611.311  
2015-01-03 1117.554 462.955 7.237 470.192  
2015-01-04 1081.980 385.023 19.982 405.005  
2015-01-05 1325.920 216.540 26.522 243.062  
2015-01-06 1331.916 117.227 32.888 150.115  
2015-01-18 1139.695 127.215 31.259 158.474  
2015-01-19 1457.299 31.620 23.681 55.301  
2015-01-20 1489.253 16.738 10.791 27.529  
2015-01-21 1486.707 50.390 22.120 72.510  
2015-01-22 1496.905 62.295 12.119 74.414

а также обращаться к конкретной колонке по дате

df.loc['2015-12-20':'2015-12-25', 'Wind']

Date  
2015-12-20 370.889  
2015-12-21 583.385  
2015-12-24 450.480  
2015-12-25 420.503  
Name: Wind, dtype: float64

Или использовать колонку как переменную

df.Wind.loc['2015-12-20':'2015-12-25']

Date  
2015-12-20 370.889  
2015-12-24 450.480  
2015-12-25 420.503  
Name: Wind, dtype: float64

а также к каждой колонке можно обращаться по ключу

df[['Wind']].loc['2015-12-20':'2015-12-25']

Wind  
Date   
2015-12-23 552.048  
2015-12-24 450.480  
2015-12-25 420.503

df['Wind'].loc['2015-12-20':'2015-12-25']

Date  
2015-12-20 370.889  
2015-12-25 420.503  
Name: Wind, dtype: float64

также можно обращаться по индексу чрез метод iloc

df.iloc[0:2,0:3]

Consumption Wind Solar  
Date   
2015-01-01 1088.317 325.165 NaN  
2015-01-02 1246.588 603.554 7.757

Данные можно представлять с нужной частотой при помощи метода asfreq, например с частотой D - день, W,M,Y для недели, месяца и года соответственно.

df[['Wind']].loc['2015-10-20':'2015-12-25'].asfreq('W')

Wind  
Date   
2015-10-25 107.528  
2015-12-06 589.628  
2015-12-13 288.092  
2015-12-20 370.889

можно сделать обращение по месяцу с периодом 1 неделя

df.loc['2016-02'].asfreq('W')

Consumption Wind Solar Wind+Solar  
Date   
2016-02-07 1158.086 515.965 38.073 554.038  
2016-02-14 1179.309 280.405 26.907 307.312  
2016-02-21 1192.219 641.776 22.521 664.297  
2016-02-28 1155.177 187.644 71.204 258.848

или точно также с использованием открытой нотации обращения к массиву

df.loc['2015':].asfreq('Y')

Consumption Wind Solar Wind+Solar  
Date   
2015-12-31 1122.732 306.450 20.226 326.676  
2019-12-31 1124.933 425.225 36.562 461.787

Полагаем, что в данном случае будет визуально правильней поменять индексы на значение года

df.loc['2012':].asfreq('Y').set\_index(df.loc['2012':].asfreq('Y').index.year)

Consumption Wind Solar Wind+Solar  
Date   
2015 1122.732 306.450 20.226 326.676  
2018 1148.098 245.130 9.327 254.457  
2019 1124.933 425.225 36.562 461.787

Для индексов в формате дат также доступно группирование методом groupby. Группирование groupby происходит по заданным периодам, например W, 'Y' или A (год), '2y' (по 2 года) и т.д.

Часто после использования методов groupby, asfreq, а также groupby используется некоторая функция итога, например, sum, mean, median or std.

df.groupby(pd.Grouper(freq='1y')).sum()

Consumption Wind Solar Wind+Solar  
Date   
2015-12-31 479496.047 77172.286 34779.771 111419.348  
2016-12-31 486995.370 74996.242 34146.371 109089.308  
2017-12-31 492116.437 102670.983 35882.978 138553.961  
2018-12-31 498895.226 108564.492 41231.973 149796.465  
2019-12-31 490474.666 123801.867 41914.747 165716.614

df.resample('1w').median().head(3)

Consumption Wind Solar Wind+Solar  
Date   
2015-01-04 1102.9355 423.989 7.757 470.192  
2015-01-11 1331.9160 440.318 19.811 448.917  
2015-01-18 1462.7150 406.982 21.405 425.088

df.asfreq('1w').head(3)

Consumption Wind Solar Wind+Solar  
Date   
2015-01-04 1081.980 385.023 19.982 405.005  
2015-01-11 1162.956 630.378 19.811 650.189  
2015-01-18 1139.695 127.215 31.259 158.474

The full list of frequencies with its description can be find in this book <https://jakevdp.github.io/PythonDataScienceHandbook/03.11-working-with-time-series.html>

Также, полагаем, что в наборе достаточно много значений NaN. Давайте их уберем. Для этого в Pandas есть несколько инструментов, в том числе ffill, bfill для заполнения пропусков соответственно следующими или предыдущими значениями, например можно использовать метод так .asfreq('D', method='ffill'). Также возможны использования методов удаления пропусков dropna или заполнения заданными значениями filna. Если используется метод dropna, то из данных будет удалена вся строка с пропуском.

Давайте для начала посмотрим сколько у нас NaN значений. Для этого можно использовать или метод isnull или isna .

df.isna().sum()

Consumption 0  
Wind 7  
Solar 8  
Wind+Solar 9  
dtype: int64

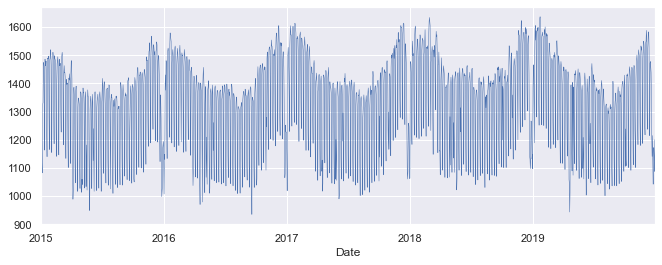
df.fillna(0, inplace=True)  
df.head(3)

Consumption Wind Solar Wind+Solar  
Date   
2015-01-01 1088.317 325.165 0.000 0.000  
2015-01-02 1246.588 603.554 7.757 611.311  
2015-01-03 1117.554 462.955 7.237 470.192

Визуализация временного ряда

Для начала давайте посмотрим на наиболее простой тип визуализации для одного из столбцов наших данных

df['Consumption'].plot(linewidth=0.5);

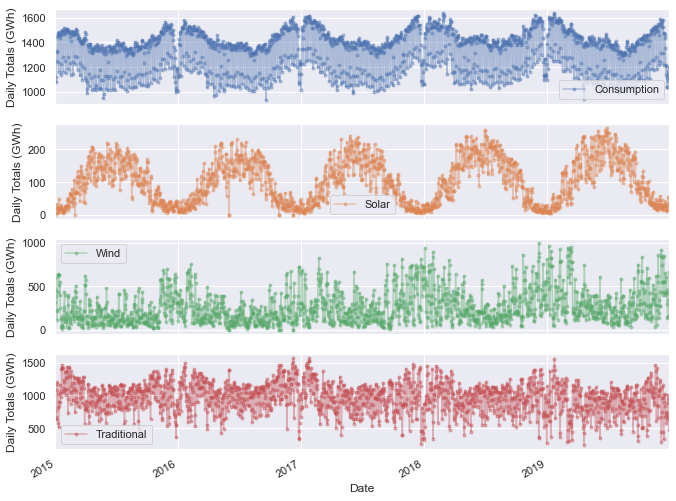


В наших данных есть составляющее общего потребления, солнечной и ветряной энергии. Полагаем, что нам может потребоваться столбец, соответствующий другим источникам (не альтернативным).

df['Traditional'] = df['Consumption'] - df['Solar'] - df['Wind']

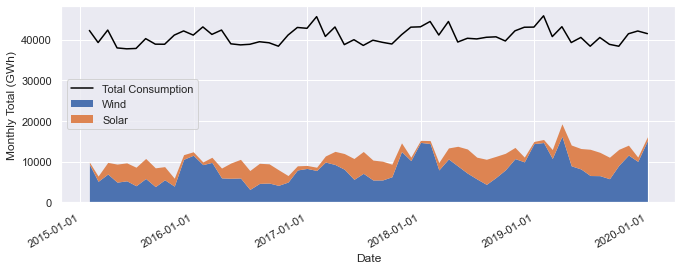
Также давайте попробуем визуализировать и остальные столбцы

cols\_plot = ['Consumption', 'Solar', 'Wind','Traditional']  
axes = df[cols\_plot].plot(marker='.', alpha=0.4, linestyle='-', figsize=(11, 9), subplots=True)  
for ax in axes:  
 ax.set\_ylabel('Daily Totals (GWh)')



Теперь попробуем провести визуализацию на одном графике, с периодом 1 месяц

df\_monthly = df.resample('M').sum(min\_count=7)  
  
fig, ax = plt.subplots()  
ax.plot(df\_monthly['Consumption'], color='black', label='Total Consumption')  
  
df\_monthly[['Wind', 'Solar']].plot.area(ax=ax, linewidth=0)  
  
ax.xaxis.set\_major\_locator(mdates.YearLocator())  
ax.legend()  
ax.set\_ylabel('Monthly Total (GWh)');



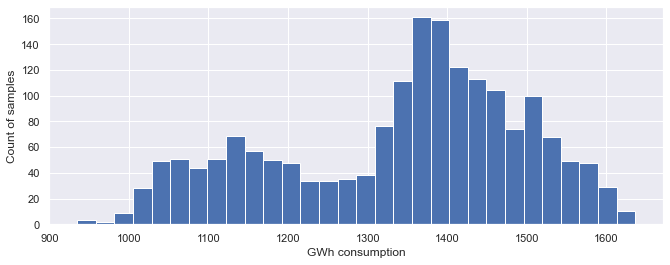
Анализ полученных графиков показывает следующее:

* Все три графика осциллируют во времени в течение года. Рискнем предположить, что это связано с сезонностью и изменением погоды.
* Потребление электроэнергии выше зимой и ниже летом.
* Потребление энергии имеет 2 сезонных составляющих:
  + основная, с примерным диапазоном значений 1300-1500 ГВт
  + дополнительная со значениями порядка 1100 ГВт, которая предположительно связана с изменением потребления в течение неделе. Это также подтверждается заметным снижением потребления в начале каждого года.
* Пик производства солнечной энергии приходится на лето.
* Пик производства ветряной энергии приходится на зиму, причем колебания этого ряда куда более подвержены дисперсии. Полагаем, что это связано с погодным фактором.
* Значение альтернативных источников энергии растет, но очень медленно.
* Общее электропотребление, а также потребление из альтернативных источников имеют растущий тренд, тогда как тренд традиционных источников - спадающий.

Анализ общего потребления

Давайте подробней изучить сезонность по подробнее. Для анализа гипотезы о наличии двух составляющих в общем потреблении давайте посмотрим на распределение значений.

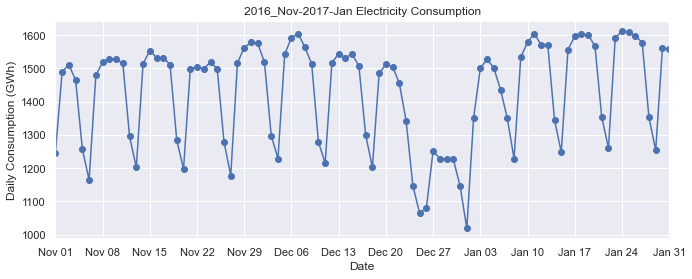
ax = df.Consumption.hist(bins=30)  
# ax = df.Consumption.plot(kind='kde' )  
ax.set\_ylabel('Count of samples')  
ax.set\_xlabel('GWh consumption')  
plt.show()



В распределении мы видим две составляющих, предположительно соответствующих двум периодам в данных. Пики этих составляющих приходятся на 1100 и 1400 ГВт.

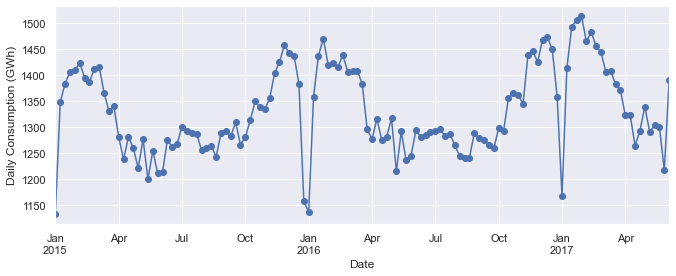
построим график потребления для периода в 2 месяца, включая начала и конец года

ax = df.loc['2016-11':'2017-01', 'Consumption'].plot(marker='o', linestyle='-')  
ax.set\_ylabel('Daily Consumption (GWh)')  
ax.set\_title('2016\_Nov-2017-Jan Electricity Consumption')  
# For more convinient ticks (week ticks)  
ax.xaxis.set\_major\_locator(mdates.WeekdayLocator())  
# Format 3-letter month name and day number  
ax.xaxis.set\_major\_formatter(mdates.DateFormatter('%b %d'))  
plt.show()



Действительно у нас имеется две сезонных компоненты. Спад потребления связан с выходными). Однако, Потребление во время смены годов можно считать или аномалией или третьей составляющей (что возможно правильней). Проверим гипотезу о 3 составляющей. Для этого используем скользящее среднее.

ax = df.loc['2013-10':'2017-05', 'Consumption'].\  
 resample('W').mean().plot(marker='o', linestyle='-',linewidth=1.5)  
ax.set\_ylabel('Daily Consumption (GWh)')  
plt.show()

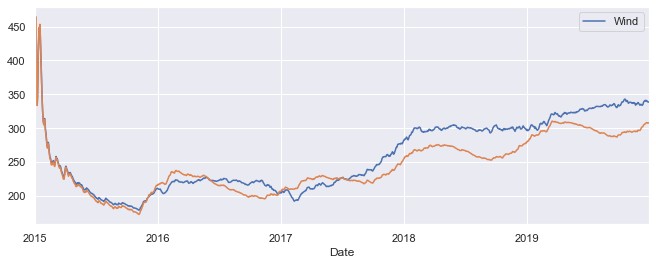


Анализ потребления ветровой энергии

Также давайте проверим наличие тренда у ветровой составляющей (полагаем, что тут тренд не очевиден). Для этого используем скользящее вреднее при помощи метода rolling. Данный метод можно выполнить с заданным периодом, в частях года или днях. Отметим, что также можно воспользоваться методом скользящего среднего

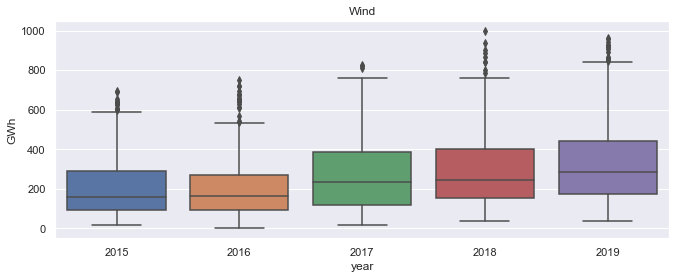
df[['Wind']].rolling('365d').mean().plot( linewidth=1.5, );  
df.Wind.ewm(halflife=365, min\_periods=0,adjust=True).mean().plot()

<AxesSubplot:xlabel='Date'>



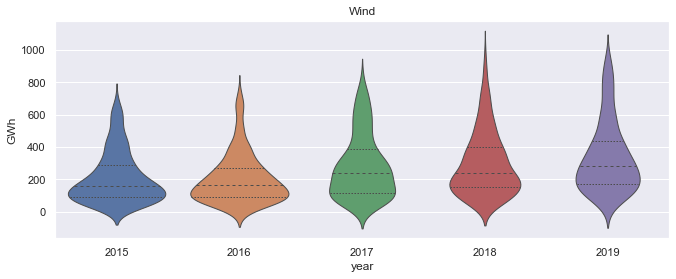
Также проверим тренд при помощи построения BBox

ax = sns.boxplot(data=df, x=df.index.year, y='Wind')  
ax.set\_ylabel('GWh')  
ax.set\_xlabel('year')  
ax.set\_title('Wind')  
plt.show()



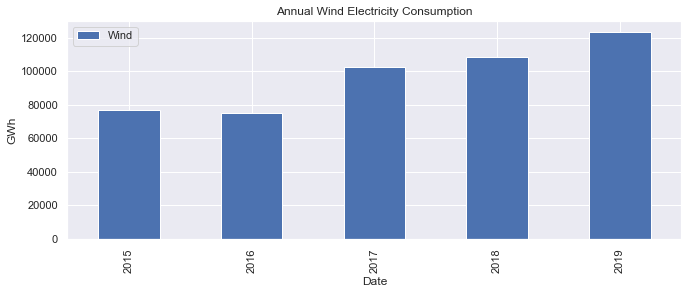
Отметим, что по мимо BBox можно также воспользоваться альтернативным отображением

ax=sns.violinplot(data=df, x=df.index.year, y='Wind',  
 split=True, inner="quart", linewidth=1, )  
ax.set\_ylabel('GWh')  
ax.set\_xlabel('year')  
ax.set\_title('Wind')  
plt.show()



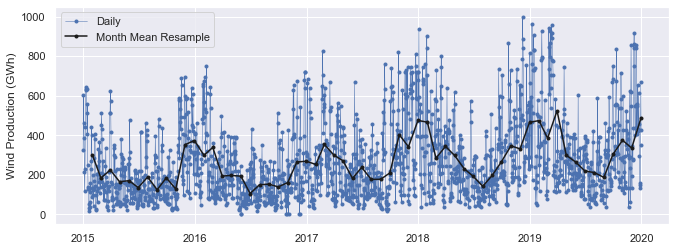
Однако, также можно проверить гипотезу при помощи bar-plot

df\_test = df[['Wind']].resample('Y').sum()  
  
ax = df\_test.set\_index(df\_test.index.year).plot.bar()  
ax.set\_title('Annual Wind Electricity Consumption')  
ax.set\_ylabel('GWh');



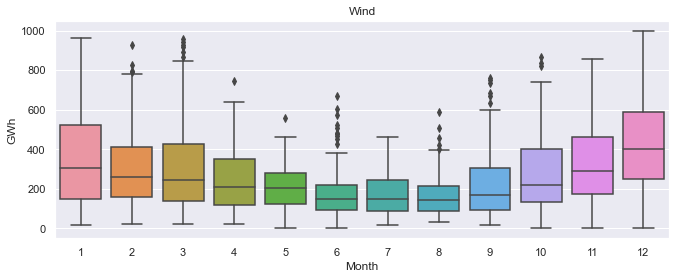
Также давайте проверим наличие сезонных составляющих у ряда с ветряной энергией.

start, end = '2015-01', '2019-12'  
  
fig, ax = plt.subplots()  
ax.plot(df.loc[start:end, 'Wind'],marker='.', linestyle='-', linewidth=0.5, label='Daily')  
ax.plot(df.resample('M').mean().loc[start:end, 'Wind'], marker='o', markersize=3, linestyle='-', label='Month Mean Resample', color='k')  
ax.set\_ylabel('Wind Production (GWh)')  
ax.legend();



Вероятно, сезонность нестабильна, чтобы это проверить давайте построим boxplot месячных значений

ax = sns.boxplot(data=df, x=df.index.month, y='Wind')  
ax.set\_ylabel('GWh')  
ax.set\_xlabel('Month')  
ax.set\_title('Wind')  
plt.show()

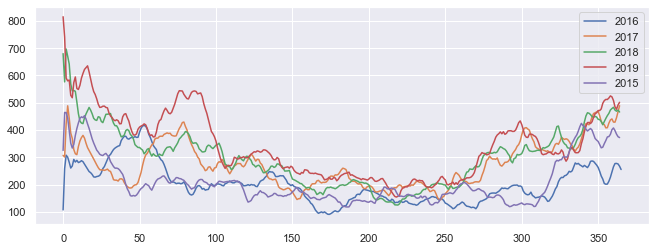


На boxplot видно большое количество выброс и изменение среднего значения и дисперсии.

В дополнение мы можем проверить нестационарность сезонности ветровой составляющей с использованием скользящего среднего по годам. Пример показан ниже.

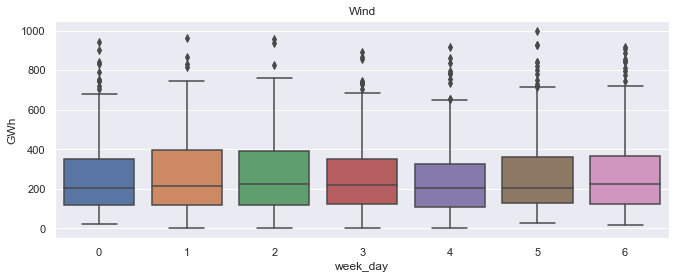
for year in list(set(df.index.year)):  
 plt.plot(df.loc[str(year):str(year), 'Wind'].rolling('30d').mean().values, label=year)  
plt.legend()

<matplotlib.legend.Legend at 0x1a014397eb8>



Вероятно, нестационарность сезонности увеличивается от года к году. Также оценим сезонность дней недели.

ax = sns.boxplot(data=df, x=df.index.weekday, y='Wind');  
ax.set\_ylabel('GWh')  
ax.set\_xlabel('week\_day')  
ax.set\_title('Wind')  
plt.show()



на графике не видно явной сезонности или нестационарности, однако число выбросов достаточно большое.

В качестве упражнения слушателю предлагается провести анализ солнечной энергии и традиционной энергии

# Юнит 11 Моделирование временных рядов

**(ОПЦИОНАЛЬНАЯ ПРАКТКА, ЕЕ ПРОВЕРЯТЬ НЕ БУДУ)**

Вторая тетрадь посвящена моделированию временных рядов.

Моделирование временных рядов — это одна из важных задач анализа данных, которая позволяет предсказывать поведение данных в будущем на основе прошлых наблюдений.

Предлагаем вам ознакомиться с данной тетрадью в Google Colab:

<https://colab.research.google.com/drive/13-O7EjeGlD9SZ-IfllhFnIucyk5RpktY>

Моделирование временных рядов

Импорт данных

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
import matplotlib.dates as mdates  
# Use seaborn style defaults and set the default figure size  
sns.set(rc={'figure.figsize':(16, 4)})

Детерминированные модели

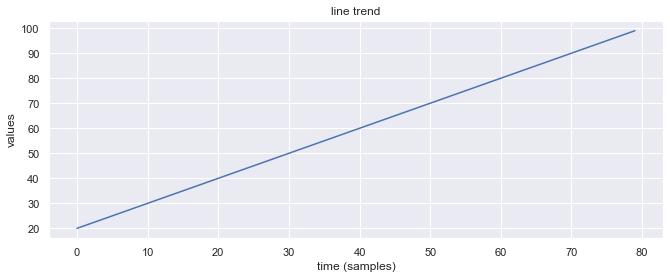
Простейшим случаем детерминированного временного ряда является одномерная (одномерная) зависимость значения от времени, представленная в следующей форме

где: - это временной ряд - набор выборок, проиндексированных некоторой переменной , обычно – это временные отметки, если временной шаг дискретный, он также может быть обозначен как (номер выборки), в этом случае в реальном времени - значение шага будет соответствовать , где - период шага (период дискретизации, с которым берутся отсчеты). - некоторый начальный постоянный уровень, - это наличие некоторого тренда, который является частью зависимости с медленным изменением. - это сезонность или некоторые «относительно быстро изменяющиеся» периодические составляющие - это относительно быстро меняющаяся часть взаимосвязи. - это некоторые периодические компоненты с "относительно медленным изменением" с нерегулярным периодом и относительно высокой интенсивностью. Часто в тренд включаются циклическая и 𝑎0 части, в этом случае модель может быть задана как

Исследование тренда

Сначала промоделируем временной ряд как имеющий только линейный тренд, взятый с единым периодом выборки.

ts = np.arange(20,100)  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='time (samples)', ylabel='values',  
 title='line trend')  
plt.show()

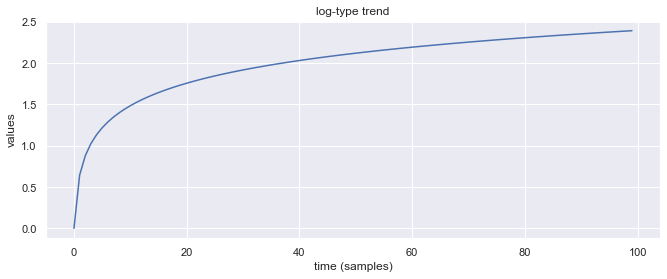


Есть несколько простейших типов трендов, которые могут быть представлены во временных рядах:

* Линейный тренд
* параболический тренд
* полиномиальный тренд
* гиперболический тренд
* экспоненциальный тренд
* насыщение (логистический) тренд
* логарифмический тренд
* циклический тренд (огибающая)
* многие другие функции, которые, как правило, сглажены, очень медленно меняются или даже монотонны.

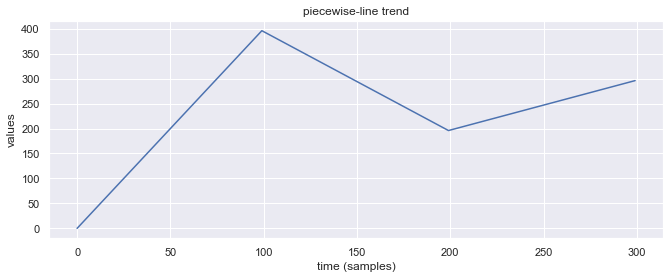
Теперь мы можем попробовать логарифмический тренд с основанием ( Число Эйлера, натуральный логарифм) и .

N\_OF\_SAMPLES=100 # Number of samples  
a = 4#const  
c = 0.4   
n = np.arange(N\_OF\_SAMPLES)  
ts = c\*np.log(1+a\*(n))  
  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='time (samples)', ylabel='values',  
 title='log-type trend')  
plt.show()



Для многих реальных временных рядов кусочно-монотонное поведение является естественным, поэтому часто необходимо моделировать кусочно-монотонный тренд с одной или несколькими точками перегиба.

N\_OF\_SAMPLES=100 # Number of samples  
a = 4#const  
n = np.arange(N\_OF\_SAMPLES)  
ts1 =a\*n  
a = 2#const  
n = np.arange(1,N\_OF\_SAMPLES+1)  
ts2 = ts1[-1]-a\*n  
a = 1#const  
n = np.arange(1,N\_OF\_SAMPLES+1)  
ts3 = ts2[-1]+a\*n  
ts = np.concatenate((ts1,ts2,ts3))  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='time (samples)',   
 ylabel='values',  
 title='piecewise-line trend')  
plt.show()



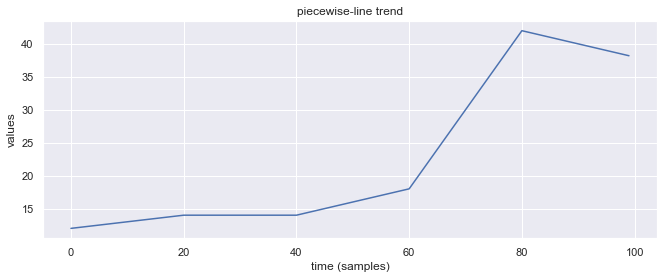
Давайте теперь смоделируем поведение кусочно-линейного тренда, предложенное моделью Prophet,

где матрица изменения роста, описывающая точки перегиба (матрица с единицами),  
 постоянная скорости роста, смещение, вектор изменения скорости роста, коэффициенты изменения роста , точки перегиба.

*Заметим*:

В простейшем случае модель сводится к $y(t) = (k)t + m, $ для временного ряда без точек перегиба . В модели Facebook Prophet предложили рассматривать логистическую модель как альтернативу линейной, в этом случае тренд можно представить в виде

N\_OF\_SAMPLES=100 # Number of samples  
  
k = 0.1  
m = 12  
  
n = np.arange(N\_OF\_SAMPLES)  
  
inflection\_points = np.array([20, 40, 60, 80])#change points  
  
a = np.zeros(shape=(inflection\_points.size, N\_OF\_SAMPLES)) # the matrix of growth changing   
  
# fill matrix  
# n[:,None] -mean add new dimention,   
#(n[:,None] > inflection\_points) is the logic operation to fill matrix with false, true  
#(n[:,None] > inflection\_points)\*1 prodece 1 for true and 0 for false  
a = ((n[:,None] > inflection\_points) \* 1).T  
  
  
delta = np.array([-0.1, 0.2, 1, -1.4])#vector with growth rate adjustments  
  
growth = (k + np.dot(a.T,delta))   
  
gamma = -inflection\_points \* delta  
offset = m + np.dot(a.T,gamma)  
  
ts = growth\* n + offset  
  
  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='time (samples)',   
 ylabel='values',  
 title='piecewise-line trend')  
plt.show()



Сезонность

Простейшую сезонную часть временного ряда можно представить в виде

где: — интенсивность сезонной составляющей; — сезонный период (месяц, день, неделя и т. д.); — начальный сдвиг (начальная фаза) сезонности; — период выборки; и — сезонная частота () и частота дискретизации . Давайте смоделируем эту серию.

Примечание В соответствии с теоремой Шеннона-Найквиста-Котельникова минимальное значение должно быть

Для оценки полученного количества периодов используйте

N\_OF\_SAMPLES=365 # Number of samples  
  
n = np.arange(N\_OF\_SAMPLES)  
  
a = 1  
  
  
Ts = 1/365  
  
T =1/3  
  
theta = np.pi/2  
  
print('number of periods = ',N\_OF\_SAMPLES\*Ts/T)  
  
ts = a\*np.sin(2\*np.pi\*n\*Ts/T+theta)  
  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='time (samples)',   
 ylabel='values',  
 title='seasonal part')  
plt.show()

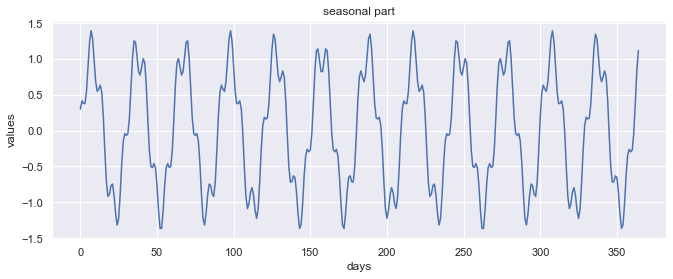
number of periods = 3.0



Давайте теперь смоделируем более сложную сезонность в году, например месяц и неделю. Мы начнем с аддитивной модели:

Отметим, что тут в отличии от циклического тренда период сезонности достаточно быстрый и в общем случае ожидается постоянным.

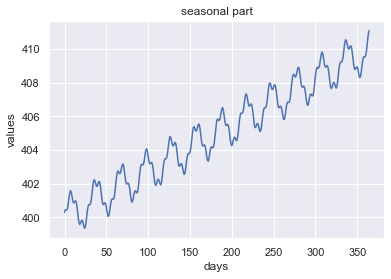
N\_OF\_DAYS=365# Number of samples  
  
days = np.arange(N\_OF\_DAYS)  
  
a\_w = 0.3 #weak influence  
  
a\_m = 1.1 #month influence  
  
T\_w = 7/365  
  
T\_m = 30/365  
  
Ts = 1/365  
  
theta\_w = np.pi/2  
  
theta\_m = 0  
  
ts = a\_w\*np.sin(2\*np.pi\*days\*Ts/T\_w + theta\_w)+a\_m\*np.sin(2\*np.pi\*days\*Ts/T\_m + theta\_m)   
  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='days',   
 ylabel='values',  
 title='seasonal part')  
plt.show()



Теперь мы можем моделировать аддитивные и мультипликативные временные ряды:

В нашем случае это будет реализовано как:

YEAR = 365  
  
WEEK = 7  
  
MONTH = 30  
  
N\_OF\_DAYS=YEAR# Number of samples  
  
days = np.arange(N\_OF\_DAYS)  
  
a\_w = 0.3 #weak influence  
  
a\_m = 1.1 #month influence  
  
T\_w = WEEK/YEAR  
  
T\_m = MONTH/YEAR  
  
Ts = 1/YEAR  
  
theta\_w = np.pi/2  
  
theta\_m = 0  
  
a\_trend = 10 #slope  
  
bias\_trend = 400  
  
trend = a\_trend\*days\*Ts+bias\_trend   
  
seasonality = a\_w\*np.sin(2\*np.pi\*days\*Ts/T\_w + theta\_w)+a\_m\*np.sin(2\*np.pi\*days\*Ts/T\_m + theta\_m)   
  
ts =trend + seasonality  
  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='days',   
 ylabel='values',  
 title='seasonal part')  
plt.show()

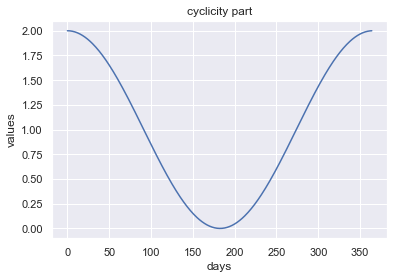


Цикличность тренда

Помимо тренда и сезонности мы можем добавить некоторую цикличность (как альтернативу можно рассматривать как дополнение трендового поведения).

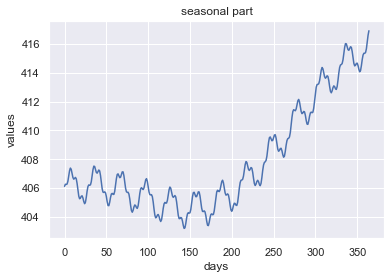
Давайте смоделируем это как некоторую зависимость год-сезон. Например, в приведенном ниже примере мы моделируем падение продаж в середине года (летом).

a\_cycl = 1  
T\_cycl = 1  
cyclicity = a\_cycl +a\_cycl \*np.sin(2\*np.pi\*days\*Ts/T\_cycl + np.pi/2)  
  
ts =cyclicity  
  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='days',   
 ylabel='values',  
 title='cyclicity part')  
plt.show()



Теперь мы можем добавить цикличность к линейному тренду

YEAR = 365  
  
WEEK = 7  
  
MONTH = 30  
  
N\_OF\_DAYS=YEAR# Number of samples  
  
days = np.arange(N\_OF\_DAYS)  
  
a\_w = 0.3 #weak influence  
  
a\_m = 1.1 #month influence  
  
T\_w = WEEK/YEAR  
  
T\_m = MONTH/YEAR  
  
Ts = 1/YEAR  
  
theta\_w = np.pi/2  
  
theta\_m = 0  
  
a\_trend = 10 #slope  
  
bias\_trend = 400  
  
trend = a\_trend\*days\*Ts+bias\_trend   
  
seasonality = a\_w\*np.sin(2\*np.pi\*days\*Ts/T\_w + theta\_w)+a\_m\*np.sin(2\*np.pi\*days\*Ts/T\_m + theta\_m)   
  
a\_cycl = 2.91  
T\_cycl = 1  
cyclicity = a\_cycl+a\_cycl \*np.sin(2\*np.pi\*days\*Ts/T\_cycl + np.pi/2)  
  
ts =trend + seasonality + cyclicity  
  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='days',   
 ylabel='values',  
 title='seasonal part')  
plt.show()



Особые события

Помимо тренда и регулярной сезонности, в модель временных рядов могут быть введены "особые события".

Например, если мы моделируем временные ряды продаж, будет интересно добавить некоторые изменения спроса в рабочие дни.

В простейшем случае это можно сделать следующим образом:

где - означает остаток деления; — дельта-функция Кронекера,

i — номер дня ().

*Примечание* Если вы хотите считать дни не с понедельника, используйте

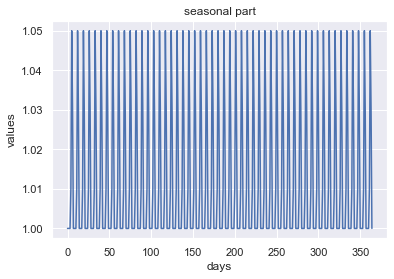
где – количество дней, на которые нужно сдвинуться. Давайте проверим этот результат

N\_OF\_DAYS =14  
shift = 1  
days = np.arange(1,N\_OF\_DAYS+1)  
print((days-1+(shift-1))%7+1)

[1 2 3 4 5 6 7 1 2 3 4 5 6 7]

Для моделирования поведения дней недели мы будем использовать набор из 7 коэффициентов

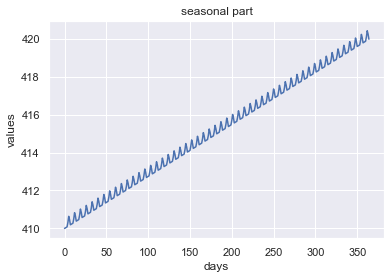
N\_OF\_DAYS=365  
  
days = np.arange(N\_OF\_SAMPLES)  
  
# week days coefficients  
a\_week = np.array([1, 1, 1, 1, 1.01, 1.05, 1.03])  
  
#for the number of days multiples of the week  
week\_days = list(a\_week)\*int(N\_OF\_DAYS/7)   
  
# add rest of the days  
week\_days = np.array([\*week\_days,\*a\_week[:N\_OF\_DAYS%7]])  
  
#check that week\_days size equal to N\_OF\_DAYS  
assert week\_days.size==N\_OF\_DAYS  
  
ts = week\_days  
  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='days',   
 ylabel='values',  
 title='seasonal part')  
plt.show()



Для имитации влияния дней недели на линейный тренд можно ввести следующую модель

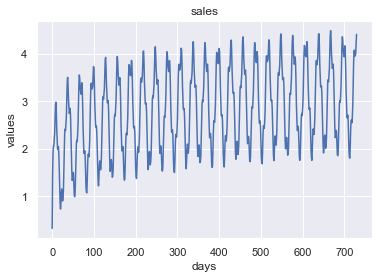
где модель дней недели с трендом

N\_OF\_DAYS = 365  
days = np.arange(N\_OF\_SAMPLES)  
  
a\_trend = 10 #slope  
bias\_trend = 400  
week\_coefficients = np.array([1, 1, 1, 1, 1.02, 1.05, 1.03])  
  
a\_week = week\_coefficients\*a\_trend  
  
week\_days = np.array([\*list(a\_week)\*int(N\_OF\_DAYS/7), \*a\_week[:N\_OF\_DAYS%7]])  
  
trend = a\_trend\*n\*Ts+bias\_trend   
  
ts =week\_days + trend   
  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='days',   
 ylabel='values',  
 title='seasonal part')  
plt.show()



Давайте теперь добавим часть сезонности и создадим модель на два года с логистическим трендом.

YEAR = 365  
  
WEEK = 7  
  
MONTH = 30  
  
N\_OF\_DAYS=YEAR\*2# Number of samples  
  
days = np.arange(N\_OF\_DAYS)  
  
a\_w = 0.3 #weak influence  
  
a\_m = 1.1 #month influence  
  
T\_w = WEEK/YEAR  
  
T\_m = MONTH/YEAR  
  
Ts = 1/YEAR  
  
a\_trend = 5   
c\_trend = 0.34   
  
week\_coefficients = np.array([0.95, 1, 1, 1, 1, 1.25, 1.03])  
  
a\_week = week\_coefficients\*c\_trend  
  
trend = c\_trend\*np.log(1+a\_trend\*days)  
  
seasonality = a\_w\*np.sin(2\*np.pi\*days\*Ts/T\_w )+a\_m\*np.sin(2\*np.pi\*days\*Ts/T\_m )   
  
  
week\_days = np.array([\*list(a\_week)\*int(N\_OF\_DAYS/7), \*a\_week[:N\_OF\_DAYS%7]])  
  
  
ts = week\_days + trend + seasonality   
  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='days',   
 ylabel='values',  
 title='sales')  
plt.show()



Моделирование шумов

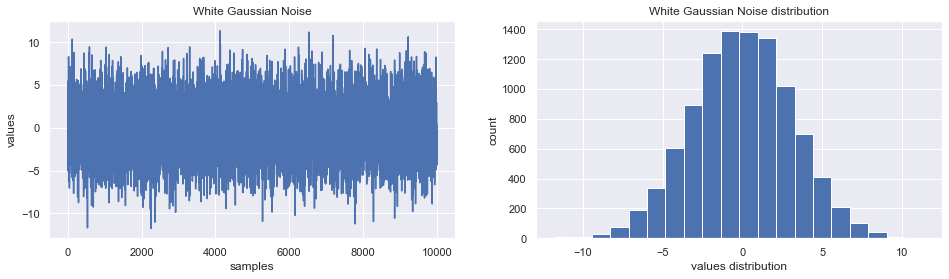
Белый Гауссов Шум

Помимо детерминированной части временного ряда, важно смоделировать его стохастическое поведение. Стохастическое поведение временного ряда в первую очередь связано с влиянием шума. Наиболее простой и наиболее распространенной моделью шума является Белый Гауссов шум (White Gaussian Noise, WGN) (идентичен понятию независимой и одинаково распределенной величины (independent and identically distributed, i.i.d) модели шума). WGN имеет нормальное распределение с нулевым средним значением и дисперсией . Шум обычно обозначается как

а его распределение задается функцией плотности вероятности:

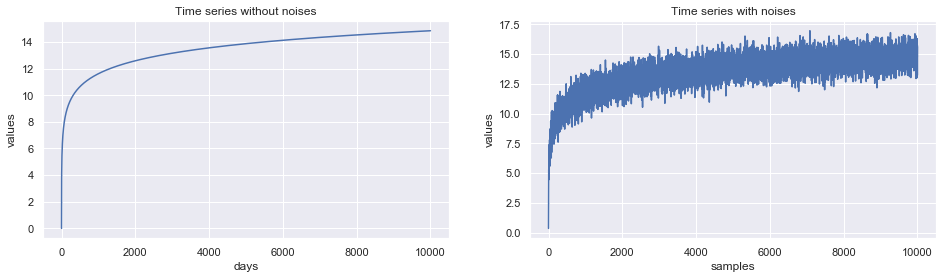
где мощность шума - стандартное отклонение шумов (квадратный корень из дисперсии). Давайте смоделируем белый гауссовский шум.

N\_OF\_SAMPLES = 10000  
  
noise\_power = 10   
  
wgn = np.sqrt(noise\_power)\*np.random.normal(size = N\_OF\_SAMPLES)  
  
ts = wgn   
  
fig, ax = plt.subplots(1,2)  
  
ax[0].plot(ts)  
ax[0].set(xlabel='samples',   
 ylabel='values',  
 title='White Gaussian Noise')  
  
ax[1].hist(ts, bins = 20)  
ax[1].set(xlabel='values distribution',   
 ylabel='count',  
 title='White Gaussian Noise distribution')  
plt.show()



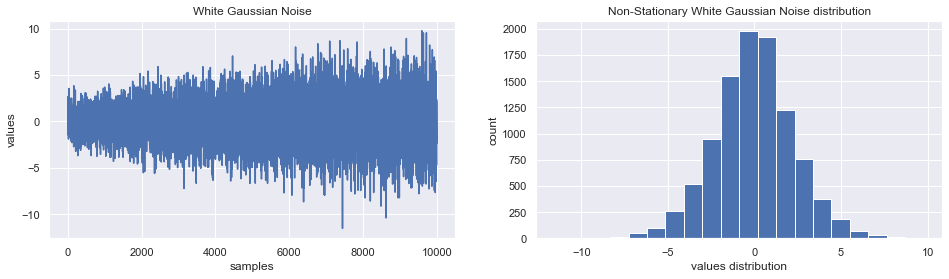
Таким образом, мы можем видеть, как шум влияет на временной ряд.

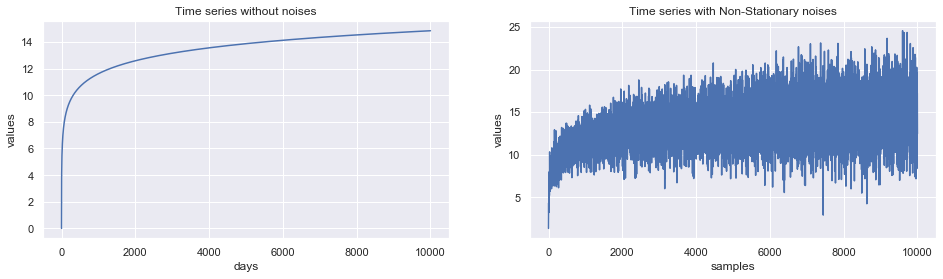
N\_OF\_SAMPLES = 10000  
  
noise\_power = 0.5   
  
wgn = (np.sqrt(noise\_power))\*(np.random.normal(size = N\_OF\_SAMPLES))  
  
a = 4#const  
c = 1.4   
n = np.arange(N\_OF\_SAMPLES)  
ts = c\*np.log(1+a\*(n))  
  
ts\_wn = ts + wgn  
  
fig, ax = plt.subplots(1,2)  
  
ax[0].plot(ts)  
ax[0].set(xlabel='days',   
 ylabel='values',  
 title='Time series without noises')  
  
ax[1].plot(ts\_wn)  
ax[1].set(xlabel='samples',   
 ylabel='values',  
 title='Time series with noises')  
plt.show()



Помимо одинаково распределенного шума, соответствующего стационарной модели шума, важно моделировать нестационарные случаи. Самый простой случай — линейно возрастающая вариация,

N\_OF\_SAMPLES = 10000  
a = 4#const  
c = 1.4   
noise\_power = np.linspace(1,10,N\_OF\_SAMPLES) #linearly growing noise power   
wgn = np.sqrt(noise\_power)\*np.random.normal(size = N\_OF\_SAMPLES)  
ts = c\*np.log(1+a\*np.arange(N\_OF\_SAMPLES))  
ts\_wn = ts + wgn  
fig, ax = plt.subplots(1,2)  
ax[0].plot(wgn)  
ax[0].set(xlabel='samples',   
 ylabel='values',  
 title='White Gaussian Noise')  
ax[1].hist(wgn, bins = 20)  
ax[1].set(xlabel='values distribution',   
 ylabel='count',  
 title='Non-Stationary White Gaussian Noise distribution')  
fig, ax = plt.subplots(1,2)  
ax[0].plot(ts)  
ax[0].set(xlabel='days',   
 ylabel='values',  
 title='Time series without noises')  
ax[1].plot(ts\_wn)  
ax[1].set(xlabel='samples',   
 ylabel='values',  
 title='Time series with Non-Stationary noises')  
plt.show()



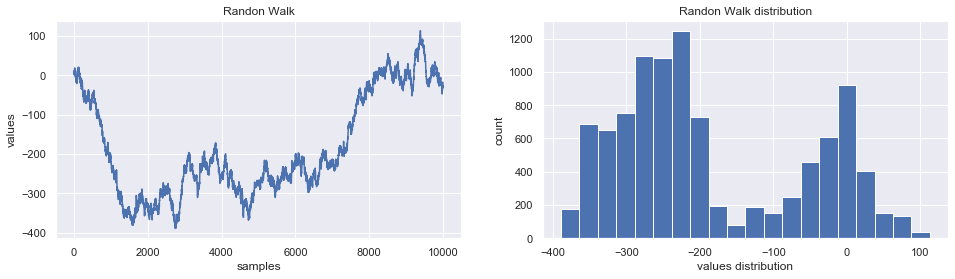


Случайное блуждание

Помимо аддитивного шума, важной моделью шума является случайное блуждание, которое в простейшем случае можно смоделировать как

где Модель широко распространена при исследовании ряда бизнес-процессов.

N\_OF\_SAMPLES = 10000  
noise\_power = 10  
wgn = np.sqrt(noise\_power)\*np.random.normal(size = N\_OF\_SAMPLES)  
ts = np.cumsum(wgn )  
fig, ax = plt.subplots(1,2)  
ax[0].plot(ts)  
ax[0].set(xlabel='samples',   
 ylabel='values',  
 title='Randon Walk')  
ax[1].hist(ts, bins = 20)  
ax[1].set(xlabel='values distribution',   
 ylabel='count',  
 title='Randon Walk distribution')  
plt.show()



Упражнения для самоконтроля (необязательные) 1

1. Реализуйте модель логистического тренда Prophet
2. Для модели тренда добавьте аддитивную квартальную сезонность.
3. Для модели тренда добавьте мультипликативную ежемесячную сезонность.
4. Реализуйте мультипликативную детерминированную модель временных рядов с сезонной, циклической и трендовой частями.
5. К модели добавьте падение спроса в праздничные дни в начале года.

Упражнения для самоконтроля (необязательные) 2

1. Исследовать влияние аддитивного стационарного и нестационарного белого шума на временные ряды с сезонными частями и частями тренда в следующей форме
2. Смоделировать модель временного ряда в следующем виде
3. Исследуйте 3 модели случайного блуждания:
   * Модель с дрейфом
   * Модель с трендом
   * Модель с изменением интенсивности

**Юнит 12. Прогноз временного ряда**

Для того, что выстроить предсказание будущих событий, основываясь на известных событиях прошлого, используется прогнозирование временных рядов.

Одним из типичных примеров прогнозирования временных рядов является предсказание погодных условий, цен на акции, продажи товаров и услуг, а также спроса на продукцию.

Решение задачи прогноза временного ряда означает создание модели, которая может предсказывать будущие значения величины на основе прошлых данных.

В этом юните мы рассмотрим типы решения задачи прогноза временных рядов.

Существует несколько методов решения таких задач прогноза временных рядов:

подход на основе статистических моделей (model based;)

подход на основе данных (data driven).

**Подход на основе статистических моделей** предполагает, что рассматривается некоторое формальное описание временного ряда, например, как сумма линейного тренда с белым гауссовым шумом. Для этой модели решается уравнения нахождения параметров тренда (наклон и смещение). Причем решение находится подходящими статистическими методами, например, методом наименьших квадратов. Подход будет работать до тех пор, пока предположения о тренде и шуме справедливы.

**Подход на основе данных** рассматривает данные без излишних гипотез об их характере. Вместо этого заявляется что при достаточно разнообразном наборе данных, охватывающем все возможные случи поведения ВР мы можем взять достаточно универсальный оценщик с адаптируемыми параметрами. Например, персептрон. Параметры мы будем адаптировать так, чтобы оценщик давал как можно точнее результаты для нашего набора данных. То есть мы решаем задачу оптимизации оценщика. Мы не меняем структуру оценщика в отличии от первого подхода. Мы ожидаем что правильная оптимизация (по регулярным признакам) позволит работать оценщику за пределами использованных для его оптимизации данных. То есть для похожих на использованные данных. Мы называем этот обобщающей способностью.

Отметим, что 1-й подход более точный, но в тех случаях, когда модель подходит для данных. При отклонении от гипотез модели (например, от гипотезы о линейном тренде) точность модели резко начинает падать. Обобщающая способность 2-го подхода сложная функция параметров данных, но чем больше разных данных и чем больше случаев, которые мы можем встретить описано, тем обобщающая способность лучше.

Сейчас мы рассмотрим первый подход из описанных. **Стохастические модели (или подход, основанный на моделях)**.

Методы на основе моделей можно разделить на непараметрические и параметрические.

**Непараметрические методы,** такие как скользящее среднее, модель Холта-Винтера, простая регрессия, подразумевают лишь оценку своих коэффициентов внутри алгоритма. Тогда как параметрические подразумевают что будут также использованы и внешние настраиваемые параметры. Позже, в следующем модуле, мы обсудим эту разницу еще раз.

Пример типичного непараметрического метода предсказания, который мы уже рассматривали, можно решать следующим образом. Линейный тренд на фоне белого гауссового шумов можно описать как

(1),

где – модель временного ряда от параметра (дискретный шаг ВР); – искомые коэффициенты; - белый гауссов шум. Допустим, что , где – объем выборки (длительность нашей тренировочной выборки).

Тогда опишем модель аппроксимирующую такой ВР как , где – результат аппроксимации для коэффициентов аппроксимации . Заметьте, результат не содержит случайностей, вместо них он содержит оценки коэффициентов ( ). То есть когда мы найдем значения этих оценок то сможем аппроксимировать ВР и предсказывать его будущие значения . По этим предсказаниям мы сможем проверить, что наши оценки действительно валидны, то есть дают результат с приемлемой ошибкой. Тогда из этого будет следовать что и гипотеза о линейной модели (1) валидна.

Теперь осталось получить оценки коэффициентов модели (1). Мы сделаем это статистическим методом так, чтобы учесть всю выборку длины . Так мы как большей степени устраним влияние шумов, потому что шумы случайны и их мат. ожидание 0. То есть чем больше мы их учтем, тем ближе мы должны оказаться к их мат. ожиданию. Это можно сделать, например, методом наименьших квадратов. По сути метод как раз говорит о том, что мы складываем случайные величины как корень из суммы их квадратов. Для решения задачи указанным методом запишем функционал наименьших квадратов . Этот функционал мы будем минимизировать

На самом деле функционал с учетом (1) выглядит как .

Распишем решение системы для желающих проследить процесс **(не обязательная часть!)**  Поскольку функционал мы ожидаем гладким в виде параболы в области выше нуля, минимум функционала можно найти как нуль его производную по каждому из параметров:

C учетом того, что*а* также перенеся в правую часть получим

Также напомним, что *,* а  
 *,* тогда  
 . Отметим, что если бы рассматривали более простую модель

решением бы было выражение

**Перечислим особенности непараметрических методов**;

* модели легко обучаются и дообучаются;
* низкая вероятность переобучения;
* обеспечивают наилучшую точность для сравнительно простых данных (стационарных с гауссовыми шумами или некоторыми простыми шумами, такими как симметрично распределенные);
* производительность резко снижается, если поведение данных отличается от предполагаемого (формируют предполагаемую статистическую гипотезу);
* хорошо работают только в одномерном случае;
* легко интерпретируются;
* могут иметь аналитическое решение (например, для линейной регрессии);
* для работы нам нужно знать тип распределения или делать серьезные предположения о нем.

В отличии от указанных выше непараметрических методов существует ряд методов, в которых помимо решения задачи оценки параметров модели мы должны выбрать какие-то гиперпараметры при помощи внешних алгоритмов. Допустим, чтомодель ВР была быгде -набор сезонных компонент, например, год, квартал, неделя и т.д., а  **–** число сезонных компоненты. Тогда помимо решения задачи нахождения коэффициентов модели мы должны определить и число компонент. Это не всегда простая задача. Более того на практике оказывается, что это внешняя задача, то есть она решается не в рамках процедуры поиска коэффициентов. Другими словами это гиперпараметр. Поэтому алгоритм параметрический. Примеры параметрическихх методов**,** ARIMA, GARCH, и т. д.

**Особенности параметрических методов следующие.**

* строятся на основе аналитико-параметрической модели поведения ряда;
* просты для понимания и реализации;
* модели легко обучаются и дообучаются;
* обеспечивают достаточно высокую точность прогноза при правильном выборе параметров;
* лучше работать с одномерными данными;
* лучше работают при относительно простых нестационарностях в данных;
* точность сильно зависит от выбора значений гиперпараметров (здесь гиперпараметры являются аналогом статистической гипотезы в непараметрическом подходе);
* не может справиться со сложной нелинейной зависимостью между данными;
* низкая производительность для огромных многомерных нестационарных рядов (особенно в случае данных с различным поведением и т. д.);
* легко интерпретируется;
* более устойчивый к выводам в случае хорошо выбранной модели (из-за сверхвысокого разрешения с выводами для небольших наборов данных).

Методы на основе данных можно разделить на классическое машинное обучение и глубокую нейронную сеть. К классическим методам относятся, например, такие подходы как векторная регрессия (SVR), случайный лес регрессии, XGBoost и тд. Такие подходы как правило не специализированы для анализа временных рядов. Однако подходы могут быть использованы в тех случаях, когда построить достаточно валидную модель не удается. То есть методы имеют следующие особенности.

* Позволяют работать с сильно нелинейными данными.
* Нет необходимости в статистической гипотезе для модели.
* Хорошо справляется с нестационарными отношениями между данными.
* Точность на тестовой выборке (обобщающая способность) сложная функция особенностей настройки модели машинного обучения и особенностей тренировочных данных.
* Трудно достичь сопоставимой точности с подходом на основе модели для относительно простых данных.
* Часто требуется ансамбль сетей для получения высокой точности.

Среди методов на основе данных также отдельно следует выделить глубокие нейронные сети.Эти методы позволяют работать с наиболее сложными временными рядами, позволяя автоматически извлекать из них признаки высоких уровней абстрактности. Однако, как показывает практика, методы применимы только тогда, когда нельзя использовать другие подходы. Таким образом, методы имеют следующие сособенности:

* Можно аппроксимировать любой многомерный временной ряд со сложной взаимосвязью поведения между данными, в том числе с пропущенными данными, аномалиями и другими нерегулярными шаблонами.
* Требуется длительная настройка гиперпараметров.
* Тяжело перетренировать.
* Трудно достичь сопоставимой точности с подходом на основе модели для относительно простых рядов.

Подведем итоги:

* Каждый подход имеет свои преимущества и ограничения, и выбор метода зависит от типа данных, стационарности ряда и и других факторов.
* Мы рекомендуем начинать с самых простых подходов и двигаться к сложным, только при необходимости. Всегда надо понимать, что для ВР чем сложнее подход, тем меньшую точность потенциально он может дать. Однако, также стоит понимать, что достичь пределов точности для простых можно только для простых задач.
* Важно учитывать, что точность любого предсказания сильно зависит от качества данных, объема данных для обучения и выбора гиперпараметров моделей.

# Юнит 13. Методы прогнозирования временного ряда

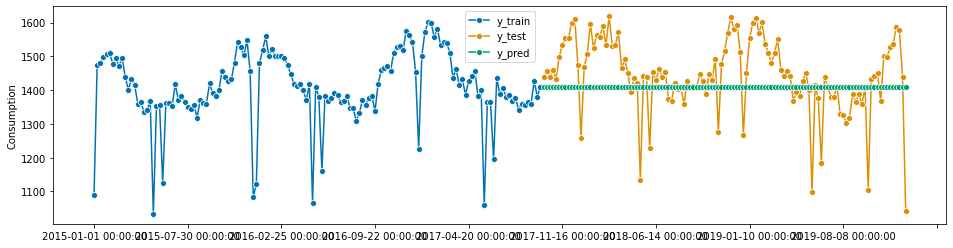
Вы уже познакомились с проблемой прогнозирования временного ряда и классификацией методов решения проблемы временных рядов. В данном юните мы детально рассмотрим несколько простых и широко используемых подходов, а именно:

* наивное предсказание,
* скользящее среднее,
* взвешенное скользящее среднее,
* экспоненциальное сглаживание,
* двойное экспоненциальное сглаживание,
* тройное экспоненциальное сглаживание.

Узнав эти методы, вы получите базовые инструменты для прогнозирования временных рядов и сможете применять их для решения различных задач в профессиональной сфере.

**Наивное предсказание** — это некоторый базовый метод, точностью меньше которого любой другой алгоритм не должен обладать. В самом простом случае Простое **наивное предсказание** можно записать как

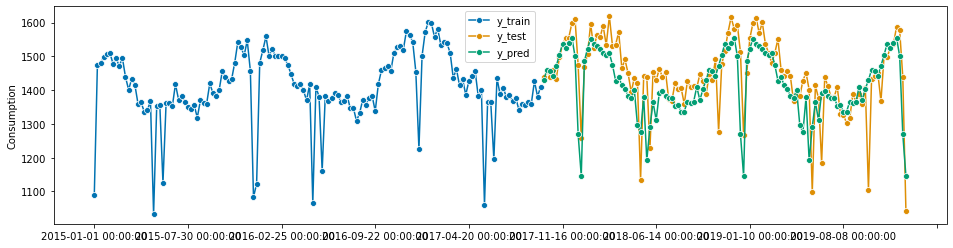
Где h - горизонт предсказания (число шагов), – текущий шаг предсказания; – результат предсказания; – значение временного ряда в момент .



Пример простого предсказания. То есть это предсказание, которое повторяет последнее значение ВР как все его будущие значения.

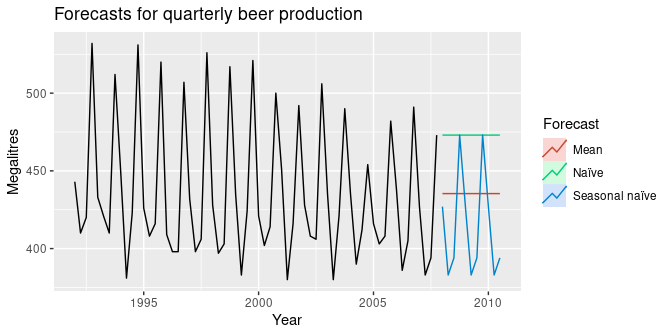
**Сезонное простое предсказание.** Допустим, чтовместо повторения последнего значения имеющегося ВР мы будем повторять последний период сезона таких значений.

Где m — период сезонности, величина, которая говорит о том, что мы повторяем последний сезон.



Пример простого сезонного предсказания. Мы можем повторять последний сезон со всеми его внутренними искажениями. Однако различие таких искажений в сезонах мы не учтем. Также тут важно правильно выбрать период сезонности, иначе ошибка будет большой.

Помимо указанных может быть предложен и ряд других простых предсказаний, например, вместо простого предсказания по последнему значению мы могли быть взять среднее значение за какой то промежуток времени или задаться скоростью изменения значений (приростом) как константой для расчета будущих значений. В целом такие методы представляют собой наиболее вычислительно простых, однако и наименее точных методов.

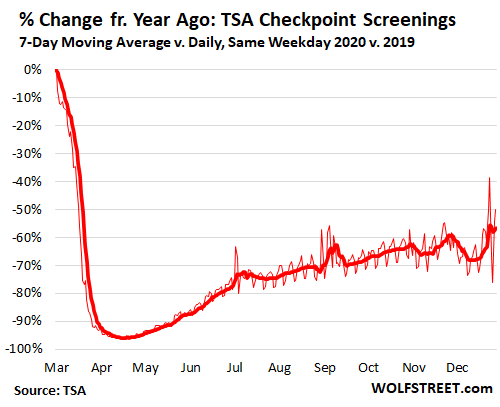


Пример простого предсказания, предсказания по среднему значению и простого сезонного предсказания.

**Скользящее среднее.** Более сложной группой методов, чем рассмотренные ранее. является группа методов, основанных на идеи скользящего окна. То есть по ВР скользит окно, чье будущее значение заменяет имеющиеся во ВР по заданной формуле. Предсказание будущий значений будет тогда строится не на самих значениях ВР, а на обработанных таким окном значениях.

В базовые идеи скользящее среднее используется для снижения шума во ВР. Однако, существует множество методов на этом принципе, позволяющих достичь разных результатов. Например, к этому классу относятся методы фильтрации – выделения отдельных составляющих ВР. Например, отдельных сезонных компонент или тренда. Простейший пример скользящего среднего это указанное сглаживания. То есть подавление шума и других быстроменяющихся процессов во ВР. В некотором смысле можно сказать что скользящее среднее аналогично эффекту размытия изображения в анализе изображений. И так, простое скользящее среднее можно записать как

где m — период сглаживания (размер окна). Чем шире окно, тем выше сглаживание. То есть увеличение размера окна позволяет оставлять снижать величину самого быстрого процесса, который не будет затронут. Другими словами, это низкочастотный фильтр – мы оставляем только самые медленные процессы. Чем шире окно, тем медленнее процессы остаются.



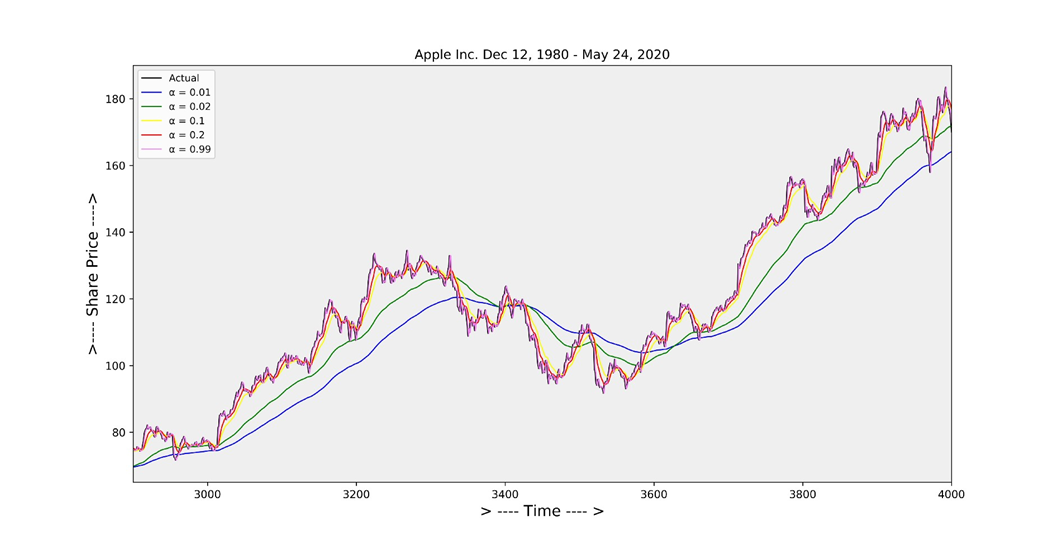
Пример скользящего среднего с заданным периодом. Скользящее среднее работает из-за предположения, что компоненты временного ряда изменяются со скоростью гораздо медленнее, чем m выборок (что обычно верно для тренда). В этом случае из-за независимости скользящее среднее уменьшает шумы WGN в m раз (СКО шумов уменьшается в раз).

Отметим также, что в ряде случаев целесообразно ввести взвешенное скользящее среднее вида , где т. Варьируя весовые коэффициент можно выделять отдельные компоненты ВР или более детально устранять шумы. В известном смысле взвешенное скользящее среднее это свертка. Другими словами, результаты сверток (результаты фильтрации, или представим, например, сверточных слоев), позволяют осуществлять более точные предсказания. Например, так можно получить разложение ВР на составляющие и получить предсказание по каждой из них отдельно. Потом результаты можно сложить. Такого рода прием мы попробуем позже в нашем курсе.

**Экспоненциальное сглаживание**

Следующая идея, чтобы уменьшить проблему , заключается в использовании экспоненциального сглаживания. По существу, это метод на основе взвешенного экспоненциального среднего, для случая, когда весовые коэффициенты выбраны как значения затухающей экспонаты. Можно показать, что в этом случае простое экспоненциальное сглаживание, SES (или Single Exponential smoothing, Single Exponential Moving Average, SEMA)может быть представлено как:

где – это параметр сглаживания; – предсказанное значение, начальное значение для предсказание задается как

****

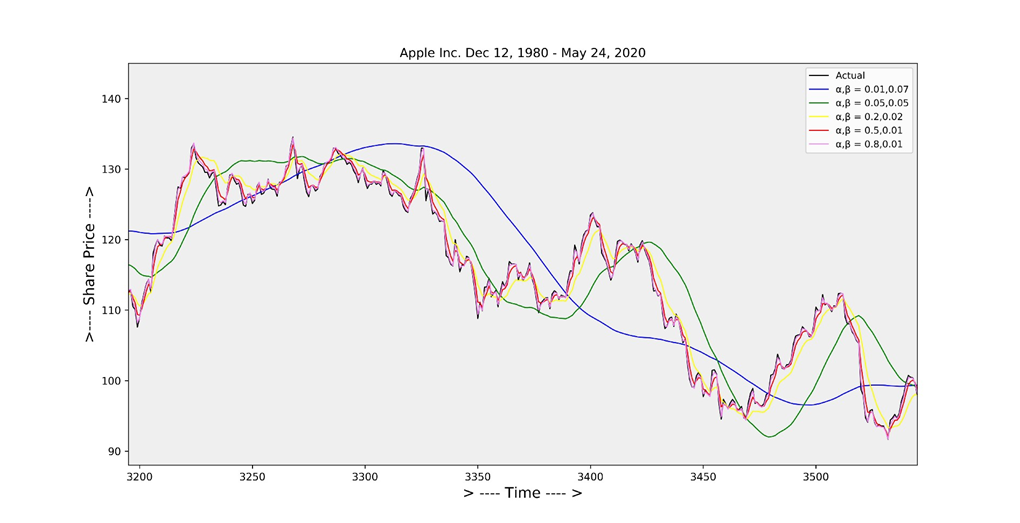
Пример экспоненциального сглаживания для разных значений параметра. Обратите внимание, что чем меньше значение сглаживания, тем более гладкой (низкочастотной, без быстрых изменений) становится ВР. То есть этот метод выделяет тренд из ВР. Но излишнее сглаживание вносит большую ошибку. Также важно отметить, что SES хорошо работает для данных с нулевым уровнем, однако для сложных рядов лучше выбрать другие методы. Следует использовать модель SES только при небольшом горизонте прогнозирования.

В некоторых случаях мы можем представить, что только выделять тренд нам недостаточно. Допустим мы хотели бы отдельно выделить тренд, а отдельно оценить начальный уровень. В некотором смысле мы хотели бы представить данные кусочно-линейно, в виде

где – тренд, – смещение тренда.

Но в отличии от непосредственной аппроксимации кусочно-линейно, тут мы бы хотели, чтобы линия была непрерывной, то есть без точек разрыва. Для такого подхода нужно сгладить два параметра. Поэтому мы получаем задачу **Двойное экспоненциальное сглаживание** DES (или двойное экспоненциальное скользящее среднее, DEMA, модель Холта). Такая модель может быть решена как

где — это дополнительный параметр сглаживания, а - горизонт предсказания; начальные значения задаются как , .

****

Пример работы методы Холта. Параметры методы определяют чувствительность модели. Чувствительная модель быстро реагирует на реальные изменения, а нечувствительная не реагирует на шум и случайные отклонения.

Если при прогнозе важны не только долговременные зависимости, но и учет быстрых изменений (сезонные колебания), то модель может быть расширена до модели Тройное **экспоненциальное сглаживание** (TES или тройное экспоненциальное скользящее среднее, TEMA, экспоненциальное сглаживание Холта-Виентерса, HW). Модель отдельно учитывает тренд, уровень и сезонную составляющую. То есть описывается как бы выражением

где – сезонная составляющая, – сезонное смешение, 𝐿 - оценка продолжительности сезона (период). Индекс 𝑛 − 𝐿 + 1 + (h − 1) 𝑚𝑜𝑑𝐿 в уравнении прогноза - это смещение сезонных компонентов от последнего полного сезона из данных наблюдений (т.е. если мы прогнозируем 3-ю точку в 45 сезоне, то мы не можем использовать сезонные компоненты 44-го сезона в будущем, поскольку этот сезон также является прогнозируемым - мы можем использовать только точки из наблюдаемых данных).

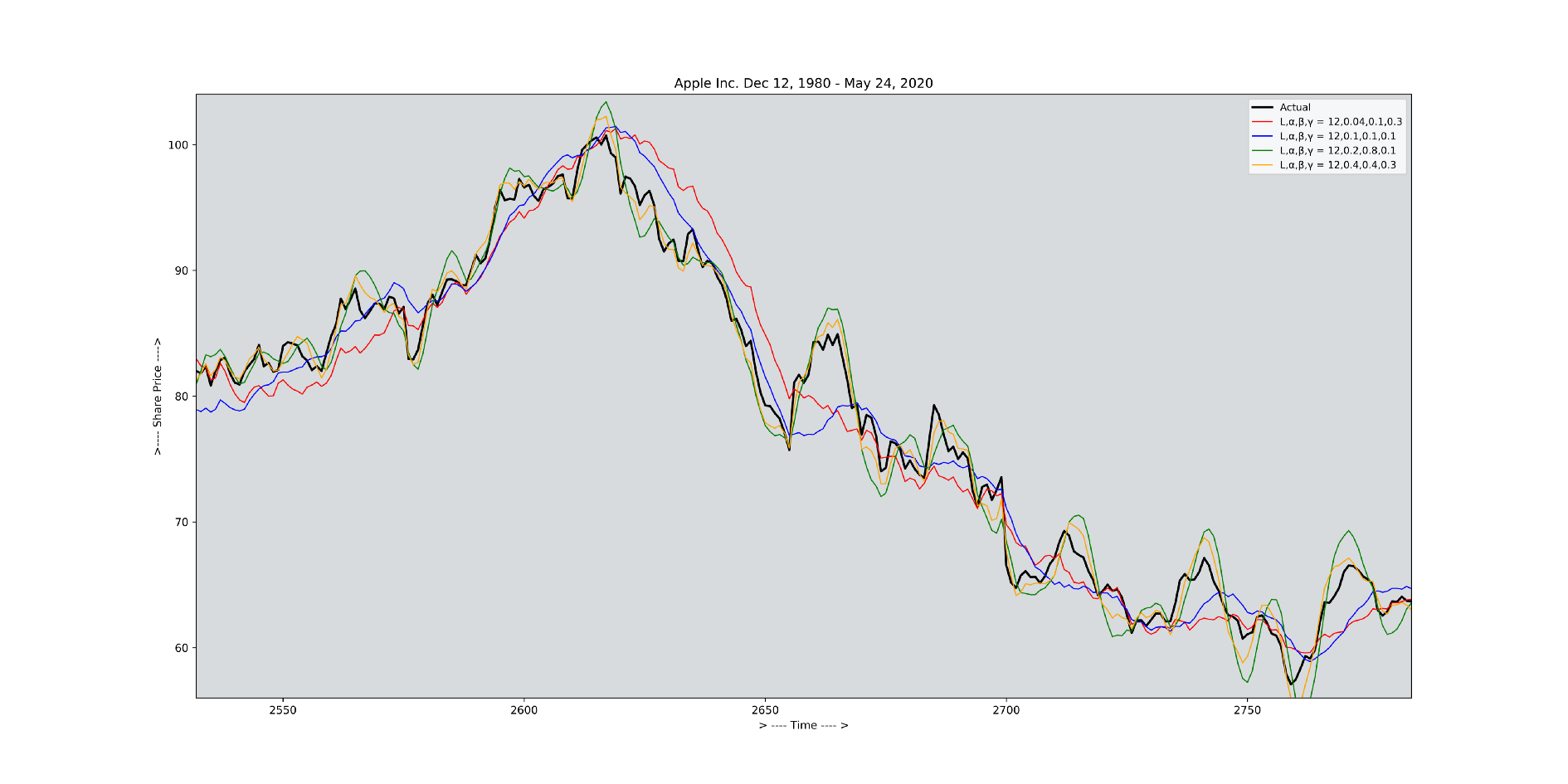
Выражение для компонент TES может быть задано как

;

;

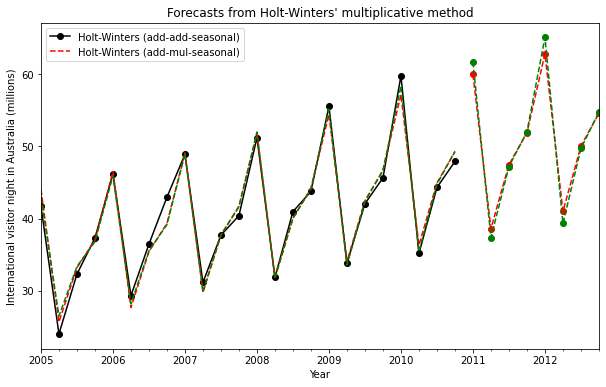
.

Где: — параметр тройного сглаживания; начальные значения задаются как , ,   
 . Отметим, что в рассмотренных моделях предполагается, что находятся в диапазоне от 0 до 1. Именно в этом случае модели будут сглаживающими.

****

Пример тройного сглаживания. Модель хорошо работает тогда, когда есть одна значимая сезонность. То есть сезонное колебание влияние которого на прогноз важно учесть. Однако тут важно указать сезонность правильно. Если сезонность нестационарная, луче указать некоторую среднюю величину на конец имеющегося набора данных.

Отметим, что помимо показанного аддитивного тройного сглаживания (экспоненциальное сглаживание Холта-Винтерса) существует несколько моделей Холта-Винтерса, например, для случайного мультипликативного ряда и для случая затухающего ряда. Выбор конкретной модели – не тривиальный вопрос. В каком случае какую модель выбрать – это не тривиальный вопрос. Можно сказать, что конфигурация модели – это гиперпараметр.

****

Пример мультипликативной и аддитивной моделей. На глаз обе модели работают схоже. Однако, небольшие расхождения есть. Поэтому нужно еще посмотреть на метрики. Какую модель выбрать лучше.

Более общие семейство Моделей экспоненциального сглаживания это модель **Error-Trend-Seasonality** (Ошибка, Тренд, Сезонность)**.** Этот подход предполагает общую постановку, включающую как описанные выше случаи, так соотнесение ошибки разложения с ними. Модель может быть записна как ETS (X, X, X) s, где X может быть N-None, A-аддитивным, M-мультипликативным, Ad-аддитивным сбросом, s-сезонный период, если S не равно None. С такими обозначениями изученные модели:

Простое экспоненциальное сглаживание соответствует ETS (A, N, N).

Тройное экспоненциальное сглаживание соответствует ETS (A, A, A).

Модели данного семейства предполагают также автоматическую процедуру подбора лучшей конфигурации. Это будет показано на практике.

Отметим также увидим, что семейство моделей экспоненциального сглаживания может быть и модифицировано и на более сложные случаи. Например, добавлен параметр иной природы по отношению к ВР (т.н. экзогенный параметр) или построена модель для нескольких сезонностей.

# **Юнит 14. Регрессионный анализ**

В этом юните мы рассмотрим различные методы регрессии и проблемы, с которыми можно столкнуться при их использовании.

Регрессионный анализ часто используется для решения задач прогнозирования. При изучении юнита вы познакомитесь с основными методами регрессии, начиная с линейной и полиномиальной регрессии. Вы узнаете, как эти методы могут быть применены к временным рядам и какие допущения имеются при использовании различных моделей.

Самый простой метод регрессии — линейная регрессия. Этот метод основан на предположении, что тренд имеет следующую модель:

где: э; и - коэффициент наклона и смещение; — переменная шума (случайного фактора) для выборки n.

Эта модель справедлива для любых линейных временных рядов, например, для линейного тренда. Используя эту модель, можно найти предыдущие и будущие значения данных. Но значения будут давать хорошую точность только если данные действительно хорошо соответствуют модели линейного тренда.

В более общем случае, ВР может быть выражен как некоторая линейная комбинация членов, например, полиномиальная регрессия

где – порядок полинома, а – набор коэффициентов.

Отметим, хотя это не важно! И ЧИТАТЬ НЕ НАДО

Для полиномиальной регрессии известно, что составляющие и т.д. не коррелируют. То есть составляющие не имеют линейной связи. Тогда можно их заменить на некоторые независимые переменные. Поэтому можно сказать, что общий вид регрессии указанного типа следующий

Или можно записать это в векторном виде как

Где отображение i-го члена, а нулевой член - смещение;  
 – вектор весов размера ; ; матрица размера , где – число точек в модели, – выходные данные; выходной вектор размера .

Таким образом, наша задача — найти коэффициенты , которые наилучшим образом аппроксимируют ВР по некоторым критериям.

Последнее означает, что мы должны ввести некоторую метрику для оценки отношения точности при выборе параметров нашей модели. Интуитивно мы можем предположить, что в качестве такой метрики можно выбрать минимум среднего по расстояниям между каждой выборкой и аппроксимационной кривой.

Аппроксимация в указанной выше форме посредством RSS-минимизации (минимизации суммы квадратов разностей/остатков). Метод такой оптимизации называется **методом наименьших квадратов (МНК, LSM)**. Может быть несколько вариантов метода. В самом простом случае метод имеет аналитическое решение , где это псведообратная матрица. Однако такой подход хорошо работает только для небольших ВР. Для ВР большего размера следует проводить поиск коэффициентов итеративно, например, методом градиентного спуска.

Также отметим, что решение МНК для белого гауссова шума является наилучшей несмещенной оценкой с наименьшей дисперсией, которая может быть оценена как , где – дисперсия шума.

Также важно понимать, что одним из основных препятствий для использования обычной регрессии является т.н. проблема плохой обсуловленности обрабатываемых данных. Другими словами, плохая обусловленность — это относительно высокая изменчивость результатов оценки, вызванная небольшим возмущением (изменением) входных данных.

Обратите внимание: чем больше отношение детерминированной части к шуму (отношение сигнал-шум), тем лучше обсуловленность ВР (т.е. меньше влияние возмущения данных результата оценки).

Проблема плохих условий или сильного влияния шума приводит к возможности переобучения

**Решения проблемы низкой обусловленности могут быть следующие.**

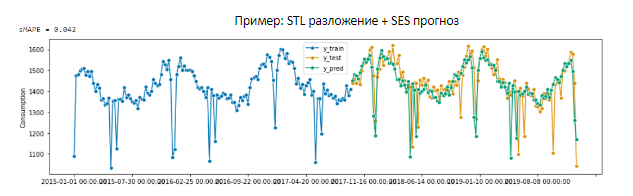
* Регуляризация. Среди таких методов, например, L1 лассо, L2 гребень (Тихонов) и др.
* Предобработка с целью подавления шумов, например, предварительно пройти по ряду скользящим средним.
* **Робастная регрессия** — методы могут быть основаны на изменении критериев оптимизации, например, вместо квадрата ошибок можно использовать медиану ошибок. Критерии обеспечивают статистически неэффективные (смещенные), но более устойчивые к шуму результаты.

**Про разложение ВР**

* 1. На использовании подходов регрессии и скользящего среднего могут быть построены высокоточные методы предсказания ВР. Также методы могут быть использованы в качестве вспомогательных. Например, можно аппроксимировать тренд ВР при помощи регрессии, затем вычесть его и аппроксимировать остаток другим способом. В более общем случае к таким подходам относят и разложения ВР на тренд и сезонность. В классическом случае такое разложение может быть произведено следующим образом.
  2. вычисление тренда методом скользящего среднего с большим окном.
  3. вычисление сезонной составляющей и шума как вычитания тренда
  4. оценка сезонной составляющей. Для каждого значения сезона m, усредняются значения без тренда (по п.2). То есть, если шаг 1 день, и сезон 1 неделя, то берется среднее всех значений за все понедельники как среднее значение понедельника, аналогично по вторникам и другим дням недели. Затем полученный сезон воспроизводится для всего набора данных.
  5. Вычисляется остаток разложения как разность тренда и полученных сезонностей.

Отметим, что данный тип разложения не самый продвинутый, есть и более сложные. Недостатком классического подхода является сложность учета нестационарной сезонности. Эта проблема решена в некоторых других алгоритмах. Например, STL разложение, использующее для вычисления сезонности локальную взвешенную регрессию по каждому периоду. Тренд в этом подходе вычисляется итерационно. Также на каждой итерации контролируется качество остатка разложения.

Результаты разложений на тренд и сезонности или и регрессионных оценок компонент ВР часто позволяют улучшить точность прогноза ВР.



Пример, когда ВР разложен и для компонент использована простое экспоненциальное сглаживание.

**Про нелинейную регрессию**

В ряде случаев указанной выше постановки задачи регрессии может быть недостаточно. То есть линейное представление ВР не позволяет достичь нужных точностей. Тогда можно поставить более общую постановку задачи как задачу обобщенной регрессии. ЕЕ можно записать как

где — это некоторый критерий (или функция потерь — это функция связи весовых параметров 𝑊 и входных данных 𝑋, в т. ч. .

Отметим, что задача обобщенной регрессии (в частности, нелинейной регрессии) может возникнуть, если данные не могут быть сведены формально к форме линейной регрессии, например, если ответы дискретные (задача классификации).

Нелинейная регрессия может быть решена с использованием метода градиентного спуска, в котором итеративно весовой вектор обновляется как

где 𝜇 — скорость обучения.

Если, например то

.

Также напомним, что процедура градиентного спуска требует инициализации вектора весов — первого приближения.

Как правило, можно использовать модифицированный градиентный спуск (например, стохастический, адаптивный, второго порядка и т. д.).

Отметим, что к классу нелинейных регрессий относят, например, нейронную сеть. Такой оценщик способен решить любую задачу аппроксимации, в том числе предсказания ВР. Это утверждается теоремой Цибенко-Хроника (1989). Эта теорема утверждает, что каждая ограниченная функция может быть аппроксимирована следующим решением: . Теорма также называется универсальной теоремой об аппроксимации. Могут быть другие формы функционалов, соответствующих теорме.

Среди других методовобощенной регрессии важно отметить стохастическую модель **Prophet.** Этомодель нелинейной регрессии, разработанная как адаптивная модель бизнес-прогноза. Мы уже рассматривали общий вид модели такого рода:

где может быть непериодический кусочно-линейный тренд или логистический тренд (уменьшение и увеличение) с насыщение видагде 𝐶, 𝑘, 𝑏 - обучаемые параметры: 𝐶 – коэффициент подъема; 𝑘 - скорость роста; 𝑏 - параметр смещения.Сезонность может быть любой начиная с недели. Сезонности меньше кодируются как регулярные но редкие события типа Сезонность может быть нестационарной, для этого она записывается в виде ряда Фурье с оцениваемыми коэффициентами.

Отметим, что классические методы прогнозирование, такие как ARIMA (рассмотрим далее) или ETS, генерируют стабильные прогнозы. Однако они могут давать большие ошибки при прогнозированиии редких событий и других нерегулрярных процессов во временных рядах. С другой стороны, если реальный ВР не описывается моделью выше, то и предсказания модели будут не точными. Поэтому тестировать модели нужно всегда. Также отметим, что если известна некоторая обобщенная (параметрическая) модель процесса, то модель прогноза можно ввести как адаптивную по образу указанной выше .

# **Юнит 15. Анализ остатков**

В этом юните вам предстоит познакомиться с анализом остатка. Анализ невязок (остатков) является важным этапом в процессе моделирования временных рядов или прогнозирования. Он позволяет проверить согласованность модели и реальных данных, а также определить необходимость внесения корректировок или изменения модели.

Для корректного решения задачи, важно не только осуществить декомпозицию временного ряда или прогноз, но важно проверить поведение ошибки.

Как обычно, мы будем вычислять ошибку предсказания как

где - значение ошибки для выборки; - прогнозируемое значениереальное значение.

В идеале мы ожидаем, что остаточные ошибки будут случайным белым гауссовским шумом — это означает, что модель (то есть прогнозы модели) охватила всю структуру, и единственная оставшаяся ошибка — это случайные флуктуации во временном ряду, которые не могут быть смоделированы (или объяснены). Другими словами мы не можем сказать, если в остатках какая то информация. В других случаях (если остатки содержат некоторую структуру или закономерности) это означает, что модель не включает всю возможную информацию.

Существует несколько **методов остаточного анализа**:

**На основе визуальной оценки.**

Визуальный анализ остатков (во временной или частотной области, а также анлиз АКФ остатков.

Анализ распределений остатков может быть произветне по гистограмме (или аппроксимации гистограммы) или, например, методом Q-Q (график вероятностей, графическое сравнение распределения вероятностей).

**На основе статистических тестов**.

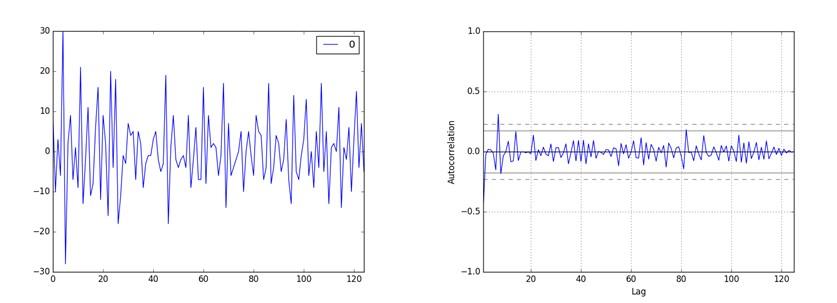
Тесты могут быть непараметрическими (наиболее популярными являются критерий Хи-квадрат, F-критерий, t-критерий, ADF-критерий, критерий Юнга – Бокса) или параметрическими. Последние могут быть более чувствительными, но требуют оптимизации гиперпараметров.

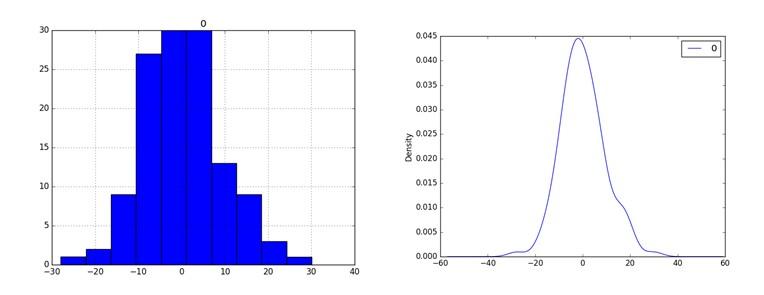
Также может быть произведен мводный статистический анализ (среднее значение, стандартное значение, поведение, разброс значений).

Отметим, что если остатки модели не удовлетворяют указанным критериям, это скорее всего означает, что в остатке есть неучтенная информация. При этом важно в каждом случае принять решение, следует ли проводить оптимизацию модели с целью извлечения этой информации или нет. Информация влияет на итоговую точность прогнозов. При этом в ряде случаев получается так, что с ростом горизонта прогнозирования ошибка в таком случае может расти.

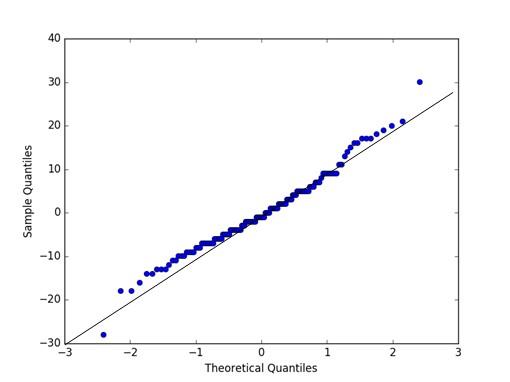
Если отклонения от нормального распределения у остатка небольшие часто это означает, что модель недообучена (или переобучена). То есть модель может остаться той же, но ее параметры нужно оптимизировать. Если отклонения существенно вероятно выбранная модель не может быть использована для ВР. Также в некоторых случаях можно использовать отдельную модель поверх существующей для извлечения данных из ВР.

Рассмотрим анализ остатков на примере.

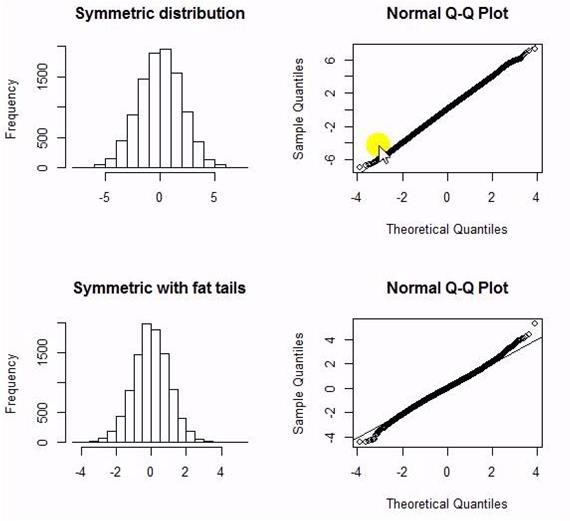
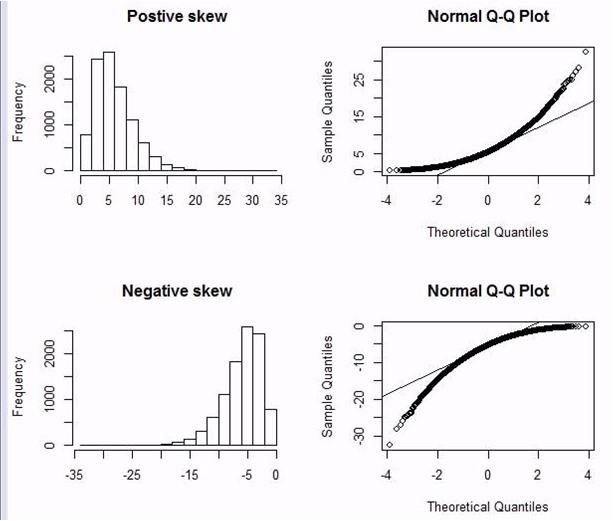
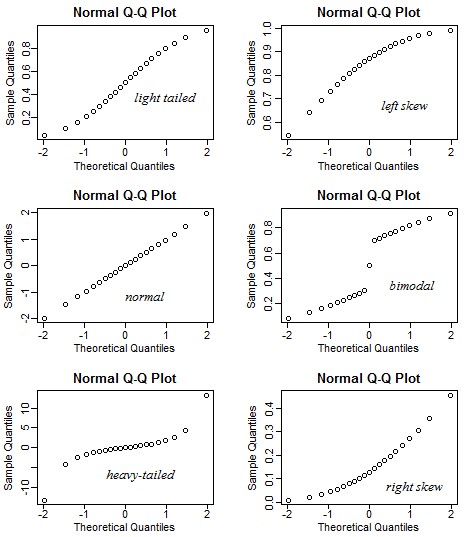
Пример остатков прогноза временных рядов. Для этого прогноза видим следующее. 1. Среднее значение ошибки близко к нулю, но, возможно, недостаточно близко (на самом деле мы не можем сказать, связано ли это с малым размером выборки, смещением ряда или наличием ценного компонента). 2. Несимметричные минимальные и максимальные значения также подтверждают некоторое ненормальное поведение. 3. Стандартное значение почти в 3 раза меньше минимального, это нормально. Чтобы подтвердить данные результаты оценим параметры остатка: mean=0,064; std=9.180; min -28,00, max = 30,00.



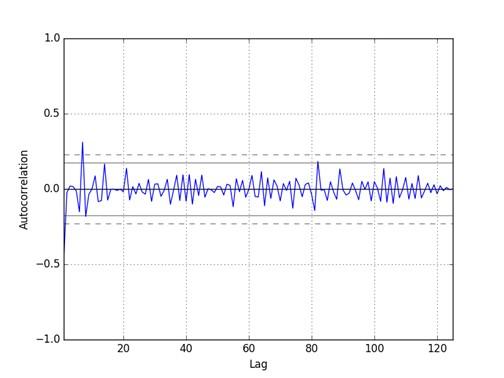
**Тут показана гистограмма и ее нормализованное** приближение для примера аналогичного анализируемому выше. Мы можем видеть некоторую погрешность среднего значения и наличие асимметрии (статистический момент 3-го порядка, некоторая асимметрия распределения). При этом отметим, что небольшая асимметрия позволяет предположить, что требуется некоторое улучшение модели. Если у модели бы была Большая асимметрия, это бы позволилопредположить, что необходимо проверить другую модель или что перед прогнозом (или разложением) необходима некоторая предварительная обработка (логарифм, квадратный корень из данных).



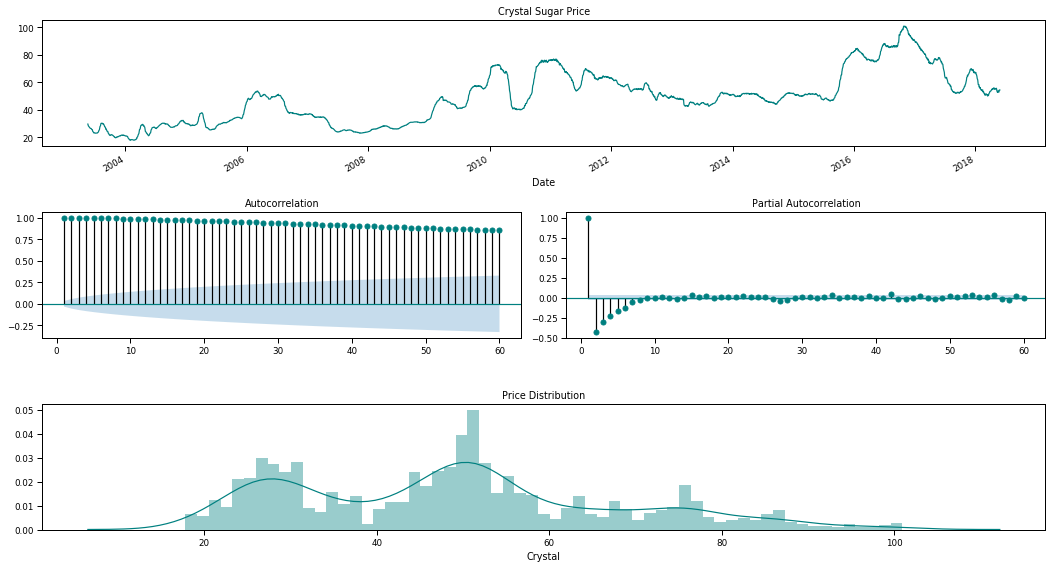
**Пример г**рафика Q-Q или график квантилей для аналогичного случая. Метод позволяет сделать визуальную оценку того, насколько похожим или отличным может быть распределение от нормального. Сравнение отображается как диаграмма рассеяния (теоретическая по оси x и наблюдаемая по оси y), где соответствие между двумя распределениями показано диагональной линией от нижнего левого угла до верхнего правого угла графика (с угол 45 градусов 𝑦 = 𝑥). График Q-Q может помочь быстро показать отклонения от этого ожидания.



Примеры разных QQ случаев



Пример выше. Тут линии на рисунке показывают уровни, ниже которых (или между + level и -level) с достоверностью 95% для непрерывных и 99% для штриховых линий образцы принадлежат белому шуму. Значения этих линий получены с помощью теста Льюнга-Бокса — теста, проверяющего, принадлежат ли m лаг ряда АКФ белому гауссовскому шуму с заданным уровнем достоверности. Расчетное значение, вычисленное с помощью теста Льюнга-Бокса, является так называемым p-значением (статистической значимостью) по тесту Льюнга-Бокса.  
Обратите внимание, что может случиться, что доверительный интервал 95% соответствует нормальному распределению, а 99% не соответствует, тогда нужны дополнительные меры анализа, но иногда это значит, что точки с такой ситуацией – это выбросы

****

На указанном графике указан пример крайне информативного остатка. Мы видим это визуально, а также по поведению АКФ и характерному распределению. В таком случае для остатка можно провести анализ (выделение информации) отдельным методом.

Выводы:

1. Для корректного решения задачи прогнозирования временного ряда, необходимо проверить поведение ошибки предсказания.

2. Анализ невязок (остатков) является важной частью процесса моделирования временных рядов или прогнозирования, так как позволяет проверить согласованность модели и реальных данных.

3. Для анализа невязок существуют различные методы, включая визуальную оценку и статистические тесты.

# Юнит 16. Практика. Библиотека SCIKIT-TIME (SKTIME)

В данном юните мы на практике познакомимся с библиотекой SCIKIT-TIME (SKTIME) — мощным инструментом для работы с временными рядами в Python.

SCIKIT-TIME предоставляет широкий спектр функций и алгоритмов для обработки, анализа и прогнозирования временных рядов. Основной упор библиотеки делается на использование методов машинного обучения для решения задач временных рядов.

В этом юните мы посмотрим на практике на методы прогнозирования временного ряда, регрессионный анализ и анализ остатков.

Текст практики и сопровождающие инструкции представлены в знакомом вам формате тетради Google Colab:

<https://colab.research.google.com/drive/10Tz7rpX1NLr1UMxNFs-2Y3ynIbbdfusS>

Изученные методы вы сможете применить для решения различных задач в области временных рядов!

Библиотека SCIKIT-TIME (SKTIME)

[Scikit-Time (Sktime)](https://www.sktime.org/en/stable/index.html) — это набор инструментов Python с [открытым исходным кодом](https://github.com/alan-turing-institute/sktime) для работы временными рядами. В основном sktime предполагает использование методов машинного обучения.

Библиотека [Sktime](https://www.sktime.org/en/stable/index.html) расширяет API scikit-learn для задач временных рядов. Он предоставляет необходимые алгоритмы и инструменты преобразования для эффективного решения задач регрессии, прогнозирования и классификации. Библиотека включает специальные алгоритмы обучения временных рядов и методы преобразования.

Если у вас не установленна данная библиотека, давайте установим ее при помощи следующего кода. Рекомендации по установке можно найти в официальной документации [тут](https://www.sktime.org/en/stable/installation.html).

Отметим, что хоть мы и указываем тут код для установки компонент sktime, в общем случае рекомендуем установить полную версию пакета с использованием командной строки. Например, для этого можно использовать следующий скрипт

pip install --upgrade sktime[all\_extras]

try:  
 import sktime  
except:  
# !pip install sktime --user  
# !pip install pmdarima  
# !pip install statsmodels  
# !pip install prophet  
 !pip install --upgrade sktime[all\_extras]  
import sktime

import warnings  
from statsmodels.tools.sm\_exceptions import ConvergenceWarning  
warnings.simplefilter('ignore', ConvergenceWarning)

import sktime  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt

from sktime.forecasting.model\_selection import temporal\_train\_test\_split  
from sktime.forecasting.base import ForecastingHorizon  
from sktime.forecasting.compose import (  
 EnsembleForecaster,  
 MultiplexForecaster,  
 TransformedTargetForecaster,  
 make\_reduction,  
)  
from sktime.forecasting.model\_evaluation import evaluate  
from sktime.forecasting.model\_selection import (  
 ExpandingWindowSplitter,  
 ForecastingGridSearchCV,  
 SlidingWindowSplitter,  
 temporal\_train\_test\_split,  
)  
from sktime.forecasting.exp\_smoothing import ExponentialSmoothing  
from sktime.forecasting.naive import NaiveForecaster  
from sktime.forecasting.theta import ThetaForecaster  
from sktime.forecasting.trend import PolynomialTrendForecaster  
from sktime.performance\_metrics.forecasting import MeanAbsolutePercentageError, MeanSquaredError  
from sktime.transformations.series.detrend import Deseasonalizer, Detrender  
from sktime.utils.plotting import plot\_series  
from sktime.forecasting.compose import TransformedTargetForecaster  
from sktime.forecasting.trend import PolynomialTrendForecaster  
from sktime.transformations.panel.tsfresh import TSFreshFeatureExtractor  
from sktime.forecasting.fbprophet import Prophet  
from sktime.forecasting.tbats import TBATS  
smape = MeanAbsolutePercentageError(symmetric = True)  
rmse = MeanSquaredError(square\_root=True)  
  
from sklearn.ensemble import RandomForestRegressor  
  
from sklearn.pipeline import make\_pipeline  
  
from sklearn.metrics import r2\_score  
from sklearn.ensemble import RandomForestClassifier  
  
  
  
r2\_score = lambda y\_pred, y\_test: 1-np.sum(np.square(y\_pred - y\_test))/np.sum(np.square(y\_test - np.mean(y\_test)))  
  
warnings.simplefilter("ignore", FutureWarning)  
%matplotlib inline

В качестве набора данных давайте загрузим рассмотренный ранее набор данных о потреблении электричества в Германии.

path\_ts = 'de\_data.csv'  
df = pd.read\_csv(path\_ts, parse\_dates=['Date'], index\_col="Date")  
df=df.fillna(df.mean())  
df.head()

Consumption Wind Solar Wind+Solar  
Date   
2015-01-01 1088.317 325.165 103.386051 371.25795  
2015-01-02 1246.588 603.554 7.757000 611.31100  
2015-01-03 1117.554 462.955 7.237000 470.19200  
2015-01-04 1081.980 385.023 19.982000 405.00500  
2015-01-05 1325.920 216.540 26.522000 243.06200

В первую очередь в данном уроке мы будем рассматривать однопеременные методы (univariate time series). Поэтому выберем один из столбцов данных в качестве отдельного временного ряда.

Кроме того, в ходе предыдущего анализа было установлено, что ряд имеет как минимум 2 составляющие (быструю, с периодом неделя и медленную с периодом 1 год). Для упрощения анализа и без значительных потерь значимости удалим быструю составляющую при помощи перегруппировки данных.

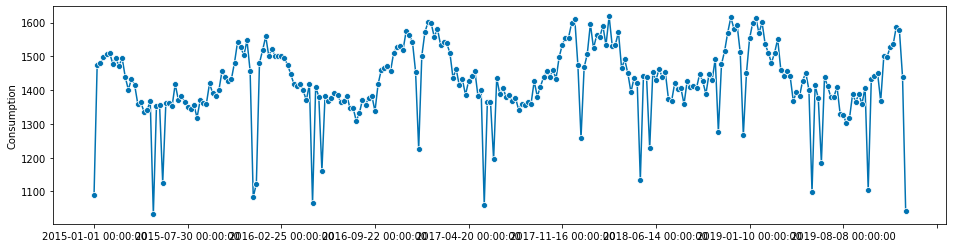
y = df.Consumption.asfreq('7d')  
# y.index = pd.PeriodIndex(y.index) # pd.to\_datetime(y.index)  
y.head()

Date  
2015-01-01 1088.317  
2015-01-08 1474.375  
2015-01-15 1479.245  
2015-01-22 1496.905  
2015-01-29 1508.443  
Freq: 7D, Name: Consumption, dtype: float64

# В качестве альтернативы можно бы было использовать resample('7d'),   
# но в этом случае пришлось бы дополнительно использовать следующую функцию.  
# def add\_freq(idx, freq=None):  
# """Add a frequency attribute to idx, through inference or directly.  
# Returns a copy. If `freq` is None, it is inferred.  
# """  
# idx = idx.copy()  
# if freq is None:  
# if idx.freq is None:  
# freq = pd.infer\_freq(idx)  
# else:  
# return idx  
# idx.freq = pd.tseries.frequencies.to\_offset(freq)  
# if idx.freq is None:  
# raise AttributeError('no discernible frequency found to `idx`. Specify'  
# ' a frequency string with `freq`.')  
# return idx  
# y = df.Consumption.resample('7d').sum()[1:-2]  
# y.index = add\_freq(y.index)

Проведем визуализацию полученных данных при помощи встроенных утилит.

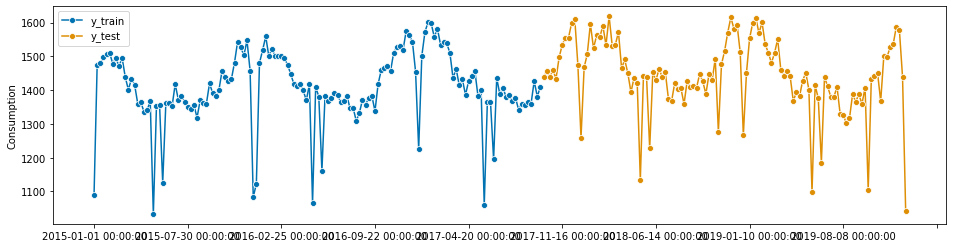
sktime.utils.plotting.plot\_series(y);



Для разделения данных воспользуемся функцией temporal\_train\_test\_split, которая позволяет адаптировать разделение данных не разрушая временные зависимости в данных. Отметим, что данная функция не единственный тип разделения данных. Более подробную информацию можно найти [тут](https://notebooks.githubusercontent.com/view/ipynb?browser=chrome&color_mode=auto&commit=24f6be86f95bfc1ec246dee7dcdd455e0a84a872&device=unknown&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f616c616e2d747572696e672d696e737469747574652f736b74696d652f323466366265383666393562666331656332343664656537646364643435356530613834613837322f6578616d706c65732f77696e646f775f73706c6974746572732e6970796e62&logged_in=false&nwo=alan-turing-institute%2Fsktime&path=examples%2Fwindow_splitters.ipynb&platform=android&repository_id=156401841&repository_type=Repository&version=98).

TEST\_SIZE = int(0.45\*y.size)  
  
y\_train, y\_test = temporal\_train\_test\_split(y, test\_size=TEST\_SIZE)  
  
print(f'Check splitted data size: Train: {y\_train.shape[0]}, Test: {y\_test.shape[0]}')  
  
sktime.utils.plotting.plot\_series(y\_train, y\_test, labels=["y\_train", "y\_test"]);

Check splitted data size: Train: 144, Test: 117



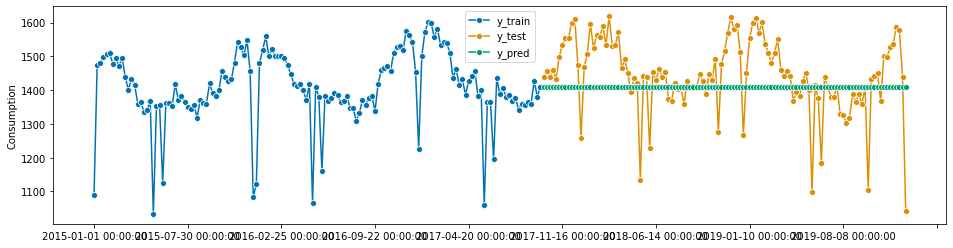
Помимо разделения данных на тренированные и тестовые предсказание в sktime может потребовать т.н. forecasting horizon - то есть непосредственного указания числа предсказываемых значений. Такой горизонт предсказания можно задавать по разному, при помощи массива или объекта класса ForecastingHorizon. Последний способ предоставляет более развитый инструментарий.

Для начала давайте попробуем выполнить наивное предсказание. То есть каждое следующее предсказанное значение будет лишь копией предыдущего, начиная с последнего значение тренировочной выборки.

Мы полагаем, что не точность данного предсказания очевидна, однако, давайте все же проверим ошибку. Для этого воспользуемся симметричной средней процентной ошибкой (symmetry Mean Average Percentage Error, sMAPE)

# ГОРИЗОНТ  
#fh = np.arange(y\_test.size) + 1  
fh = ForecastingHorizon(y\_test.index, is\_relative=False)  
  
# ПРЕДСКАЗАТЕЛЬ  
forecaster = NaiveForecaster(strategy="last")  
forecaster.fit(y\_train)  
  
# ПРЕДСКАЗАНИЕ  
y\_pred = forecaster.predict(fh)  
plot\_series(y\_train, y\_test, y\_pred, labels=["y\_train", "y\_test", "y\_pred"])  
  
# ОШИБКА  
print(f'sMAPE = {smape(y\_pred.values, y\_test.values):.3f}')

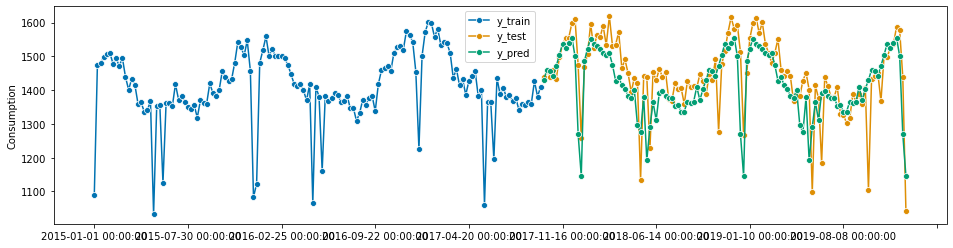
sMAPE = 0.061



Также давайте попробуем оценить точность наивного сезонного предсказания.

SEASON = 52  
  
forecaster = NaiveForecaster(strategy="mean", sp=SEASON)  
forecaster.fit(y\_train)  
  
y\_pred = forecaster.predict(fh)  
  
plot\_series(y\_train, y\_test, y\_pred, labels=["y\_train", "y\_test", "y\_pred"])  
  
print(f'sMAPE = {smape(y\_pred.values, y\_test.values):.3f}')

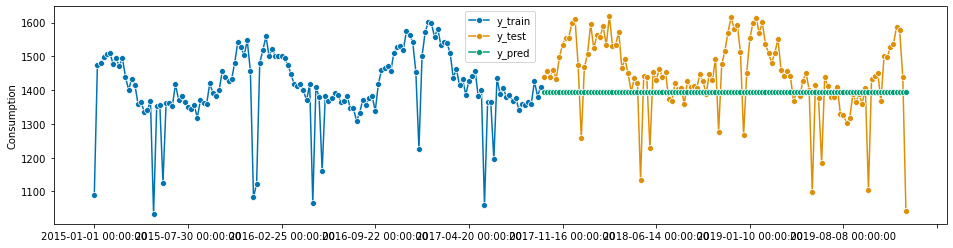
sMAPE = 0.043



Более продвинутым вариантом предсказателей является Семейство методов на основе экспоненциального сглаживания (Exponential Smoothing, Holt-Winter). Давайте посмотрим на их работу. Для начала оценим точность наиболее простого варианта Simple Exponential Smoothing.

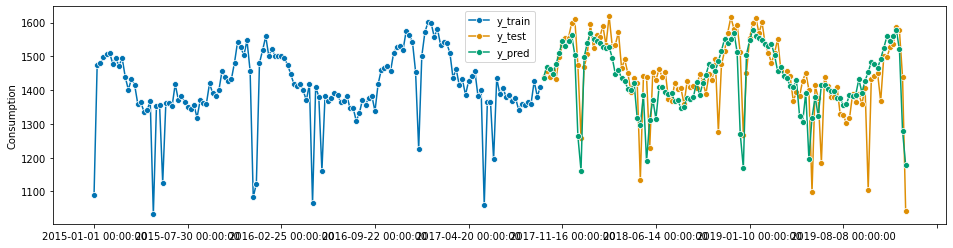
forecaster = ExponentialSmoothing()  
forecaster.fit(y\_train)  
y\_pred = forecaster.predict(fh)  
plot\_series(y\_train, y\_test, y\_pred, labels=["y\_train", "y\_test", "y\_pred"])  
  
print(f'sMAPE = {smape(y\_pred.values, y\_test.values):.3f}')

sMAPE = 0.066



SEASON = 52  
  
# МЕТОДЫ  
ses = ExponentialSmoothing(sp=SEASON)  
holt = ExponentialSmoothing(trend="add", damped\_trend=False, sp=SEASON)  
damped\_holt = ExponentialSmoothing(trend="add", damped\_trend=True, sp=SEASON)  
holt\_winter = ExponentialSmoothing(trend="add", seasonal="additive", sp=SEASON)  
holt\_winter\_add\_boxcox = ExponentialSmoothing(trend="add", seasonal="additive", use\_boxcox =True, sp=SEASON)  
holt\_winter\_mul\_boxcox = ExponentialSmoothing(trend="mul", seasonal="additive", use\_boxcox =True, sp=SEASON)  
holt\_winter\_sadd\_boxcox = ExponentialSmoothing(trend="add", seasonal="mul", use\_boxcox =True, sp=SEASON)  
holt\_winter\_smul\_boxcox = ExponentialSmoothing(trend="mul", seasonal="mul", use\_boxcox =True, sp=SEASON)  
  
forecaster = holt\_winter  
  
forecaster.fit(y\_train)  
y\_pred = forecaster.predict(fh)  
plot\_series(y\_train, y\_test, y\_pred, labels=["y\_train", "y\_test", "y\_pred"])  
  
print(f'sMAPE = {smape(y\_pred.values, y\_test.values):.3f}')

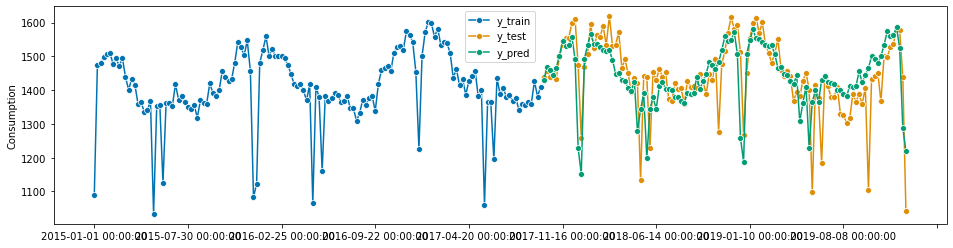
sMAPE = 0.041



Точность в предыдущих случаях оставляла желать лучшего. Однако, теперь давайте посмотрим на работу более сложных методов экспоненциального сглаживания.

SEASON = 52  
  
# МЕТОДЫ  
ses = ExponentialSmoothing(sp=SEASON)  
holt = ExponentialSmoothing(trend="add", damped\_trend=False, sp=SEASON)  
damped\_holt = ExponentialSmoothing(trend="add", damped\_trend=True, sp=SEASON)  
holt\_winter = ExponentialSmoothing(trend="add", seasonal="additive", sp=SEASON)  
holt\_winter\_add\_boxcox = ExponentialSmoothing(trend="add", seasonal="additive", use\_boxcox =True, sp=SEASON)  
holt\_winter\_mul\_boxcox = ExponentialSmoothing(trend="mul", seasonal="additive", use\_boxcox =True, sp=SEASON)  
holt\_winter\_sadd\_boxcox = ExponentialSmoothing(trend="add", seasonal="mul", use\_boxcox =True, sp=SEASON)  
holt\_winter\_smul\_boxcox = ExponentialSmoothing(trend="mul", seasonal="mul", use\_boxcox =True, sp=SEASON)  
  
# ПРЕДСКАЗАТЕЛЬ  
forecaster = EnsembleForecaster(  
 [  
# ("ses", ses),  
# ("holt", holt),  
# ("damped", damped\_holt),  
 ("holt-winter",holt\_winter),  
# ("holt-winter, additive trend, box-cox", holt\_winter\_add\_boxcox),  
 ("holt-winter, multiplicative trend, box-cox", holt\_winter\_mul\_boxcox),  
# ("holt-winter, multiplicative season, box-cox", holt\_winter\_sadd\_boxcox),  
# ("holt-winter, multiplicative both, box-cox", holt\_winter\_smul\_boxcox)   
 ]  
)  
forecaster.fit(y\_train)  
  
# ПРЕДСКАЗАНИЕ  
y\_pred = forecaster.predict(fh)  
plot\_series(y\_train, y\_test, y\_pred, labels=["y\_train", "y\_test", "y\_pred"])  
  
# РЕЗУЛЬТАТ  
print(f'sMAPE = {smape(y\_pred.values, y\_test.values):.3f}')

sMAPE = 0.043



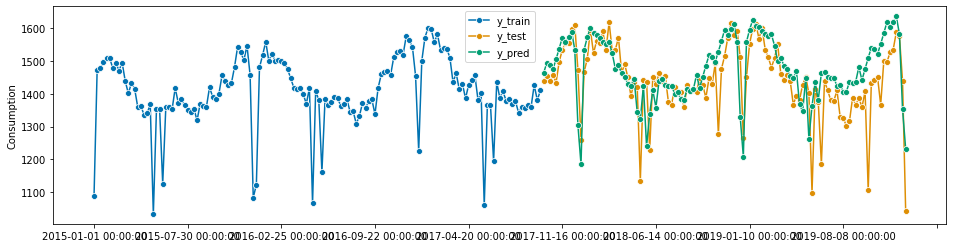
В общем случае пакет sktime позволяет использовать автоматизированный подбор параметров модели экспоненциального сглаживания. Для этого может быть использован пакет AutoETS. Модель, реализуемая данным пакетом также называется Error-Trend-Season (ETS). В случае необходимости исследователь может вручную задать параметры модели, в формате ETS(X,X,X)s, где X может быть N-None, A-additive, M-multiplicative, Ad-additive dumped, s-период сезона или None.

Частными примерами модели ETS являются:

* Simple Exponential smoothing ETS(A,N,N).
* Triple Exponential smoothing ETS(A,A,A).
* и многие другие.

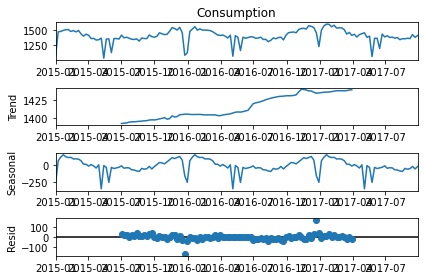
from sktime.forecasting.ets import AutoETS  
  
forecaster = AutoETS(auto=True, sp=SEASON, n\_jobs=-1)  
  
forecaster.fit(y\_train)  
  
y\_pred = forecaster.predict(fh)  
  
plot\_series(y\_train, y\_test, y\_pred, labels=["y\_train", "y\_test", "y\_pred"])  
  
print(f'sMAPE = {smape(y\_pred.values, y\_test.values):.3f}')

sMAPE = 0.045



Другим методом оценки данных с вилянием сезонности и тренда является использование предварительной декомпозиции временного ряда. Одним из наиболее простых методов декомпозиции является разделение на тренд, сезонность и остаток. Давайте посмотрим на то, как будут выглядеть результаты такого разложения.

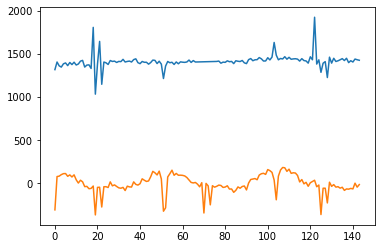
from statsmodels.tsa.seasonal import seasonal\_decompose  
result = seasonal\_decompose(y\_train, model='additive', period = 52)  
result.plot();



В рамках пакета SKTime разложение можно выполнить при помощи объектов класса Deseasonalizer и Detrender.

deseason = Deseasonalizer(model="multiplicative", sp=52)  
detrend = Detrender(forecaster=PolynomialTrendForecaster(degree=1))  
plt.plot(deseason.fit\_transform(y\_train.values))  
plt.plot(detrend.fit\_transform(y\_train.values))

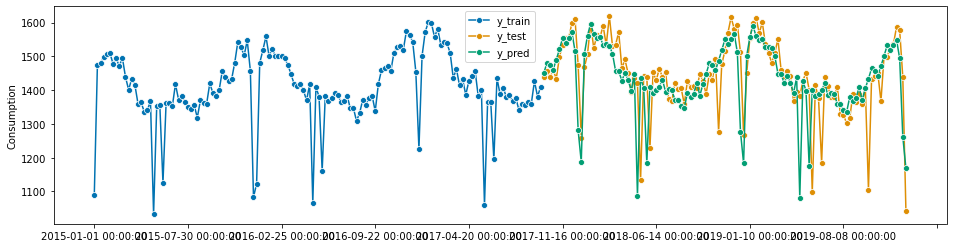
[<matplotlib.lines.Line2D at 0x7fb085895550>]



Посмотрим на работу наиболее простого экспоненциального сглаживания с использованием разложения. Для этого будем использовать контейнер TransformedTargetForecaster - который является некоторым аналогом пайплайна.

SEASON = 52  
  
fh = ForecastingHorizon(y\_test.index, is\_relative=False)  
  
# объек предсказатель  
ses = ExponentialSmoothing()  
  
# пайплайн  
forecaster = TransformedTargetForecaster(  
 steps=[  
 ("deseasonalize", Deseasonalizer(model="multiplicative", sp=SEASON)),  
 ("detrend", Detrender(forecaster=PolynomialTrendForecaster(degree=2))),  
 ("forecaster", ses)  
 ])  
forecaster.fit(y\_train)  
  
# Предсказание  
y\_pred = forecaster.predict(fh)  
  
# Результаты  
plot\_series(y\_train, y\_test, y\_pred, labels=["y\_train", "y\_test", "y\_pred"])  
  
print(f'sMAPE = {smape(y\_pred.values, y\_test.values):.3f}')

sMAPE = 0.042



# Использование пакета Scikit-learn для предсказаний

Изученные прежде методы предсказания из SKTime скорее следовало бы считать некоторыми предварительными - базовыми оценками. В последующем мы рассмотрим более развитый инструментарий оценок, специализированных для временных рядов. Однако, прежде давайте посмотрим на то, как тут могли быть быть использованы традиционные подходы из [Scikit-learn (sklearn)](https://scikit-learn.org/stable/).

Первая проблема, с которой можно столкнуться при использовании традиционных подходов для анализа временных рядов - это представление выборок данных. Выше в этом уроке мы уже рассматривали термин горизонт предсказания. Такой горизонт можно описать моделью seq2seq когда одна последовательность данных подается на вход модели и на выходе вычисляется новая - предсказанная последовательность. Однако, такой путь не совсем удачная идея для Scikit-learn. Отметим, что в пакете Scikit-learn принято задавать данные в "табличном" виде, то есть вектор входных данных - одно или несколько выходных. Для того, чтобы представить временной ряд в таком виде есть несколько техник. Мы воспользуемся одной из них recursive reduction.

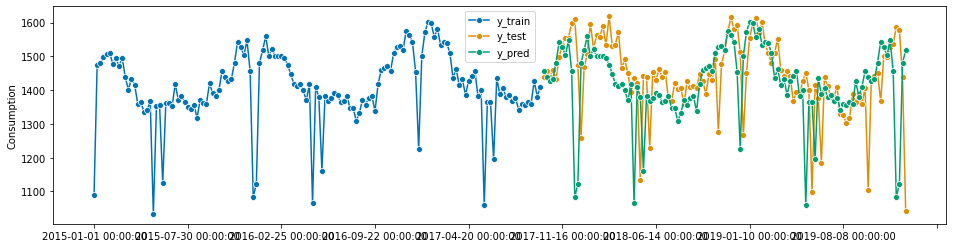
Идея техники recursive reduction достаточно проста, принцип работы показана ниже. Принцип заключается в движении по меткам ряда окна, размером с тренировочную выборку, таким образом, что каждый следующий семпл ответ связан с положением окна. Данная техника может быть реализована в рамках функции make\_reduction. Отметим, что в общем случае [данная функция имеет и другие режимы редукции](https://www.sktime.org/en/stable/api_reference/auto_generated/sktime.forecasting.compose.make_reduction.html).

Также важно заметить, что рекурсивная редукция может быть реализована только в тренировочном режиме. В тестовом режиме такая редукция сводится лишь к 1-step regression или к регрессии с горизонтом при помощи подстановки предсказанных значений.

В качестве примера рассмотрим регрессию ближайших соседей.

from sklearn.neighbors import KNeighborsRegressor  
  
REGRESSION\_WINDOW = 15  
  
regressor = KNeighborsRegressor(n\_neighbors=1)  
forecaster = make\_reduction(regressor, window\_length=REGRESSION\_WINDOW, strategy="recursive")  
  
forecaster.fit(y\_train)  
  
y\_pred = forecaster.predict(fh)  
  
plot\_series(y\_train, y\_test, y\_pred, labels=["y\_train", "y\_test", "y\_pred"])  
  
print(f'sMAPE = {smape(y\_pred.values, y\_test.values):.3f}')

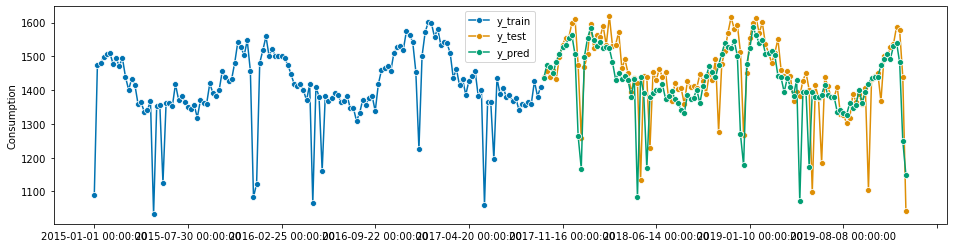
sMAPE = 0.065



Попробуем также провести регрессию с использованием пайплайна.

regressor = KNeighborsRegressor(n\_neighbors=3)  
  
forecaster = TransformedTargetForecaster([  
 ("deseasonalize", Deseasonalizer(model="multiplicative", sp=52)),  
 ("detrend", Detrender(forecaster=PolynomialTrendForecaster(degree=2))),  
 ( "forecast", make\_reduction(  
 regressor,  
 scitype="tabular-regressor",  
 window\_length=REGRESSION\_WINDOW,  
 strategy="recursive",)),  
 ])  
  
forecaster.fit(y\_train)  
  
y\_pred = forecaster.predict(fh)  
  
plot\_series(y\_train, y\_test, y\_pred, labels=["y\_train", "y\_test", "y\_pred"])  
print(f'sMAPE = {smape(y\_pred.values, y\_test.values):.3f}')

sMAPE = 0.044

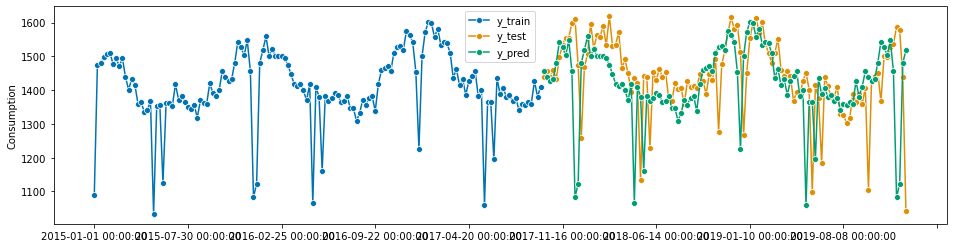


В некоторых случаях также можно использовать методы поиска лучших гиперпараметров. Пример ниже позволяет найти наилучшие гиперпараметры как для оценки из sklearn, так и для параметра из SKTime.

from sklearn.neighbors import KNeighborsRegressor  
  
from sktime.forecasting.compose import make\_reduction  
from sktime.forecasting.model\_selection import (  
 ForecastingGridSearchCV,  
 SlidingWindowSplitter,  
)  
  
param\_grid = {"window\_length": [10, 12, 15], "estimator\_\_n\_neighbors": [1,2,3,4]}  
  
regressor = KNeighborsRegressor()  
  
forecaster = make\_reduction(  
 regressor, scitype="tabular-regressor", strategy="recursive"  
)  
  
# Предварительные данные  
cv = SlidingWindowSplitter(initial\_window=int(len(y\_train) \* 0.7), window\_length=30)  
gscv = ForecastingGridSearchCV(forecaster, cv=cv, param\_grid=param\_grid)  
  
gscv.fit(pd.DataFrame(y\_train))  
  
y\_pred = gscv.predict(fh)  
  
plot\_series(y\_train, y\_test, y\_pred, labels=["y\_train", "y\_test", "y\_pred"])  
print(f'sMAPE = {smape(y\_pred.values, y\_test.values):.3f}')  
  
print('Best paramters',gscv.best\_params\_)

sMAPE = 0.065

Best paramters {'estimator\_\_n\_neighbors': 1, 'window\_length': 15}

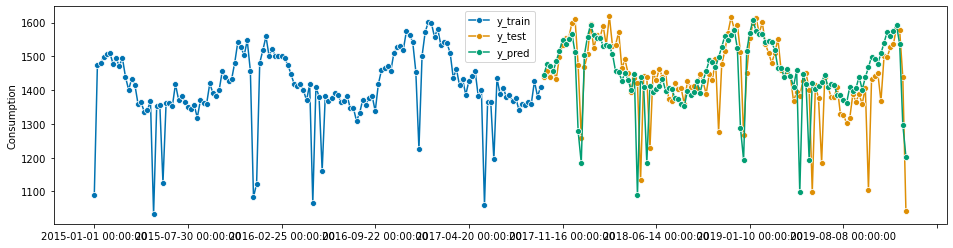


В конце также давайте посмотрим несколько современных методов регрессии, разработанных для временных рядов и представленных в рамках пакета SKTime.

ThetaForecaster - метод предсказания временных рядов на основе экспоненциального сглаживания для модели с дрейфом тренда (случайный тренд).

forecaster = ThetaForecaster(sp=SEASON)  
forecaster.fit(y\_train, fh=fh)  
y\_pred = forecaster.predict(fh)  
  
plot\_series(y\_train, y\_test, y\_pred, labels=["y\_train", "y\_test", "y\_pred"])  
print(f'sMAPE = {smape(y\_pred.values, y\_test.values):.3f}')

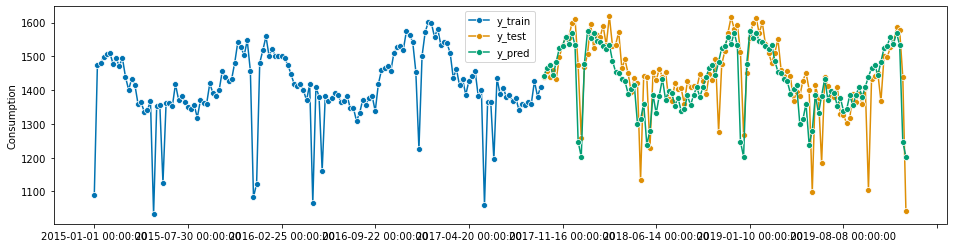
sMAPE = 0.044



TBATS - метод на основе простого преобразования временного ряда (BOX-COX) и затем использования отдельно предсказания для тренда, сезонной части и сложной модели ряда (ARMA). Метод подходит для временных рядов с несколькими сезонными составляющими.

forecaster = TBATS(  
 use\_box\_cox=True,  
 use\_trend=True,  
 use\_damped\_trend=True,  
 sp=SEASON,  
 use\_arma\_errors=True,  
 n\_jobs=-1)  
  
forecaster.fit(y\_train)  
  
fh = ForecastingHorizon(y\_test.index, is\_relative=False)  
y\_pred = forecaster.predict(fh)  
  
plot\_series(y\_train, y\_test, y\_pred, labels=["y\_train", "y\_test", "y\_pred"])  
print(f'sMAPE = {smape(y\_pred.values, y\_test.values):.3f}')

sMAPE = 0.040

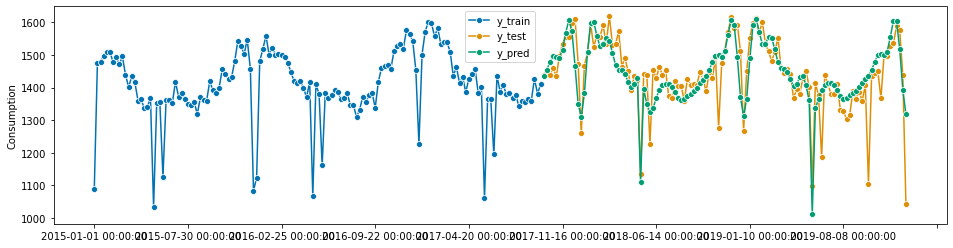


Prophet - метод предсказания временных рядов на основе т.н. обобщенной регрессии для модели бизнес процессов, которая включает тренд с точками перегиба и насыщением, несколько компонент сезонности и редкие, но регулярные события (типа выходных дней).

forecaster = Prophet(  
 seasonality\_mode='additive',  
 n\_changepoints=int(len(y\_train) / 4),  
 add\_country\_holidays={'country\_name': 'Germany'},  
 yearly\_seasonality=True)  
  
forecaster.fit(y\_train)  
fh = ForecastingHorizon(y\_test.index, is\_relative=False)  
y\_pred = forecaster.predict(fh)  
plot\_series(y\_train, y\_test, y\_pred, labels=["y\_train", "y\_test", "y\_pred"])  
print(f'sMAPE = {smape(y\_pred.values, y\_test.values):.3f}')

23:40:25 - cmdstanpy - INFO - Chain [1] start processing  
23:40:25 - cmdstanpy - INFO - Chain [1] done processing

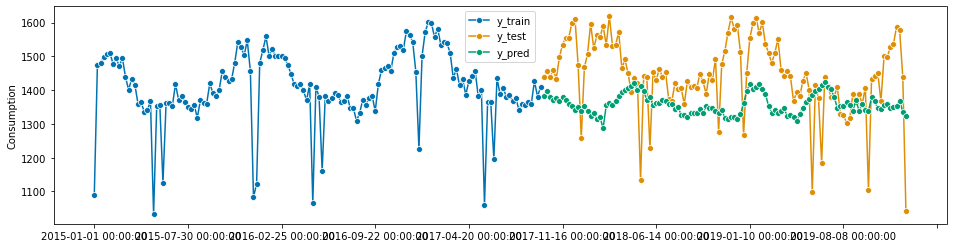
sMAPE = 0.033



Также в некоторых случаях могут быть рассмотрены предсказатели на основе выделения признаков.

regressor = make\_pipeline(  
 TSFreshFeatureExtractor(show\_warnings=False, disable\_progressbar=True),  
 RandomForestRegressor(),  
)  
  
forecaster = make\_reduction(  
 regressor, scitype="time-series-regressor", window\_length=15  
)  
forecaster.fit(y\_train)  
  
fh = ForecastingHorizon(y\_test.index, is\_relative=False)  
y\_pred = forecaster.predict(fh)  
  
plot\_series(y\_train, y\_test, y\_pred, labels=["y\_train", "y\_test", "y\_pred"])  
print(f'sMAPE = {smape(y\_pred.values, y\_test.values):.3f}')

sMAPE = 0.084



# **Юнит 17. Итоги модуля**

⭐ Поздравляем, вы завершили второй модуль по работе с временными рядами!

В этом модуле вы глубже погрузились в изучение временных рядов, однако на данном этапе мы показали лишь только основные – наиболее простые подходы к анализу этой модальности данных. Изученные материалы будут полезны при постановке и проверке базовых гипотез об анализируемом ВР.

В модуле вы изучили:

* прогноз ВР,
* регрессионный анализ,
* анализ остаточных частей ВР,
* методы предсказания ВР.

Все эти концепции, хотя и могут быть переосмыслены в рамках иных подходов, создают область знаний, дающую передовые методы решения релевантных задач.

В следующих модулях вы узнаете о более сложных подходах к анализу ВР, однако изученные в данном модуле технологии останутся базовыми (т.н. baseline)

В этом модуле вы успешно освоили несколько важных тем в области анализа временных рядов. Рассмотрели прогнозирование временных рядов, изучили различные методы прогнозирования, включая регрессионный анализ. Освоили анализ остатков, что поможет вам более точно интерпретировать результаты прогнозирования.

Как уже отмечалось ранее, временной ряд — это особый вид данных, поэтому перед работой очень важно познакомиться с рядом, анализ которого вам предстоит сделать, и проанализировать его свойства. Таким образом, знание и умение пользоваться изученными инструментами является неотъемлемой частью базового набора дата-сайентиста.

Однако для эффективного анализа данных этих методов далеко не всегда бывает достаточно — обычно они используются для первичного анализа и позволяют сделать предварительные выводы. После этого в ход идут более сложные методы, с которыми мы будем знакомиться в следующих модулях.

Список литературы:

1. Autocorrelation / www.itl.nist.gov — [Электронный ресурс]. URL: https://www.itl.nist.gov/div898/handbook/eda/section3/eda35c.htm (дата обращения: 20.08.2023).
2. Brownlee J., A Gentle Introduction to Exponential Smoothing for Time Series Forecasting in Python / machinelearningmastery.com — [Электронный ресурс]. URL: https://machinelearningmastery.com/exponential-smoothing-for-time-series-forecasting-in-python/ (дата обращения: 20.08.2023).
3. Computer Science Center (2016) [лекция] Лекция 9. Экспоненциальное сглаживание. Распознавание образов: метод к-го ближайшего соседа / YouTube. 28 декабря (<https://www.youtube.com/watch?v=M0Hz1u59Ysw>). Просмотрено: 10.06.2023.
4. Hyndman, R.J., & Athanasopoulos, G. (2018) Forecasting: principles and practice, 2nd edition, OTexts: Melbourne, Australia. OTexts.com/fpp2.
5. Installing statsmodels / www.statsmodels.org — [Электронный ресурс]. URL: https://www.statsmodels.org/devel/install.html (дата обращения: 20.08.2023).
6. Zach, How to Perform a Durbin-Watson Test in Python / www.statology.org — [Электронный ресурс]. URL: https://www.statology.org/durbin-watson-test-python/ (дата обращения: 20.08.2023).

# Юнит 18. Словарь модуля

**В ходе изучения модуля ключевыми являются следующие понятия и термины:**

Для снижения нестационарности могут быть выполнены преобразования ВР. Например, его дифференцирование для устранения тренда, или **преобразования Бокса-Кокса** для устранения т.н. волатильности — то есть неравномерной дисперсии, она также называется **гетероскедастичностью**.

**Непараметрические методы предсказания** — это методы, базирующиеся на основе аналитической модели ВР, не требующей излишне прецизионной настройки некоторых гиперпараметров. Как правило при наличии некоторых допущений о его свойствах (например, шумы – это БГШ). Как правило в условиях своей работы такие модели наиболее точные, легко интерпретируемые и наиболее быстрые.

**Параметрические методы предсказания** — это методы, базирующиеся на основе параметрической модели ВР. Такая модель является аналитической, но требует явного указания ряда своих параметров для точной работы. Такие параметры могу быть выбраны из некоторых эвристических предположений или настроены при помощи внешних критериев. Благодаря этому при корректности модель имеет более широкую область применения, нежели непараметрические модели.

**Модели, управляемые данными или подход машинного обучения** — это методы, базирующиеся на основе идеи о замене некоторой модели ВР на примеры входных данных и результатов на выходе модели. Такие модели предназначены для тех случаев, когда использовать предыдущие подходы не удается. Среди таких подходов в т.ч. подходы на основе глубокого обучения нейронных сетей. Различия классических подходов и подходов на основе глубокого обучения нейронных сетей в формализации признаков. Последние предназначены быть использованными тогда, когда признаки не могут быть выделены в ВР. Например, для продолжительных многомерных временных рядов со сложной структурой.

**Наивное предсказание** —это некоторая базовая оценка, точностью меньше которого любой другой алгоритм не должен обладать.

**Скользящее среднее** — это некоторая группа методов, основанных при предположении о самоподобие ВР, то есть предположении о том, что следующие его значения связаны с рядом предыдущих. Причем ВР должен изменяться со скоростью медленней, чем число усредняемых выборок.

**Экспоненциальное сглаживание** — это набор методов,предполагающий, что истинные значения следующих выборок временного ряда частично связаны со всеми предыдущими истинными и текущими зашумленными значениями ВР. Причем чем дальше предыдущие значения, тем экспоненциально меньше эта связь. Это позволяет модели с достаточной степенью гибко реагировать на новейшие изменения в данных, сохраняя при этом информацию об историческом поведении временного ряда. Среди таких методов могут быть выделены как простые методы, так и методы с раздельным учетом связанности для тренда, уровня и сезонности. Могут быть и другие, более сложные модели. Общая группа таких моделей часто называется Error-Trend-Seasonality (ETS).

**Экспоненциальное сглаживание** — это метод, в основе которого лежит расчет экспоненциальных скользящих средних сглаживаемого ряда. Этот метод хорошо работает для временных рядов без сезональности, но может иметь проблемы с прогнозированием сезонных колебаний.

**Тройное экспоненциальное сглаживание** — это метод, который добавляет к модели компонент сезонности. Он хорошо работает для временных рядов с явно выраженными сезонными колебаниями и позволяет учесть, как тренд, так и сезонность в прогнозах.

**Подход линейной регрессии** — это подход на основан на представлении ВР в виде некоторой линейной модели, это может быть, как просто линейная модель тренда, так и модель сводимая к линейной путем некоторых преобразований. Благодаря таким преобразованиям даже весьма сложные ВР могут быть описаны моделями такого класса. Решения таких моделей в условиях БГШ наиболее точные.

Проблемами, на которые указывает регрессионный подход являются т.н. проблемы **плохой обусловленности данных** — относительно высокая изменчивость результатов оценки, вызванная небольшим возмущением (изменением) данных. Решение такой проблемы может быть достигнуто путем **регуляризации**.

**Разложение ВР** — это набор методов, позволяющих представить (факторизовать) компоненты ВР по отдельности. Достаточно грубые методы разложения как правило предполагают выделения тренда и сезонности. Методы могут быть поострены на свойствах регрессии или скользящего среднего.

**Подход нелинейной регрессии** — это подход на основан на представлении ВР в виде некоторой линейной модели, решение которой, однако достигается общими с регрессией методами. В том числе используются приемы гладкой оптимизации, например, метод градиентного спуска, или решения с регуляризацией. К таким моделям относятся, например, нейронные сети, а также, к примеру, обобщенная адаптивная регрессия. Одной из практических реализацией последней является модель **Prophet.** Эта модель адаптирована для решения ряда бизнес задач и позволяет достигать на них высокой точности.

**Остаток модели (невязка)** — это некоторая составляющая ВР, необъясненная моделью. Такая часть может быть БГШ в идеале. Однако на практике это не всегда так. Поэтому остатки требуют анализа. Анализ может быть выполнен визуально или при помощи статистических тестов.