Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет имени первого Президента России Б. Н. Ельцина»

УТВЕРЖДАЮ

Директор по образовательной деятельности

_____ С.Т. Князев

«24» декабря 2021 г.

Анализ временных рядов

Учебно-методические материалы по направлению подготовки
09.04.01 Информатика и вычислительная техника
Образовательная программа «Инженерия искусственного интеллекта»

Екатеринбург

2021

РАЗРАБОТЧИКИ УЧЕБНО-МЕТОДИЧЕСКИХ МАТЕРИАЛОВ

Доцент, канд.техн.наук  Ронкин Михаил Владимирович

СОДЕРЖАНИЕ

Лекция 1

Лекция сопровождается презентацией.

Тема: Введение в анализ временных рядов.

Вопросы, рассмотренные в лекции:

- Особенности предмета анализ временных рядов;
- Обзор некоторых задач анализа временных рядов;
- Типы временных рядов;
- Особенности моделей временных рядов;
- Примеры задач анализа временных рядов;

В данной лекции следует уделить внимание введению в терминологию и приложения анализа временных рядов. Также следует разъяснить наиболее важные термины, которые будут в дальнейшем использоваться в курсе: тренд, сезонность, остаточная часть, цикличность, стационарность, шум. Важно показать, что временной ряд стохастический. Даже если ряд имеет детерминированную структуру, мы как правило вводим для него стохастическую модель, чтобы иметь возможность использовать соответствующий инструментарий для анализа. Также важно разъяснить чем отличается шум и остальные части ряда.

В ходе лекции следует разъяснить какие бывают типы тренда, как отличить тренд от сезонности (последняя более высокочастотная и регулярная). Важно объяснить, что сезонность именно регулярная, даже, если это редкие события. Также надо показать место цикличности – ее можно включить как сложный тренд, цикличность, как правило не регулярная – это очень низкочастотная часть. Также следует отметить такое понятие как случайный тренд (случайное блуждание) и привести примеры случайного блуждания.

Среди моделей временных рядов надо пояснить такие понятия, как лаг, одномерный и многомерный временные ряды, скользящее окно и сегмент ряда.

При анализе моделей временных рядов одним из самых важных понятий является стационарность. Это понятие нужно будет использоваться во всем курсе, его нужно тщательно объяснить «на пальцах», на следующей лекции мы к нему еще вернемся. Я обычно говорю о том, что можно представить поезд, если поставить микрофон к одному колесу и записывать его стук продолжительное время или если поставить по микрофону для каждого колеса без дефектов и записать их стук одни раз, то результат не изменится – это пример стационарного процесса.

Также нужно сказать, что стационарность мы рассматриваем по-разному, например, мы можем ввести модель детерминированного процесса со стационарным шумом или рассматривать стационарность всей модели в целом, тогда, например, тренд приведет к потере стационарности.

В отношении задач анализа временных рядов важно объяснить примеры таких задач, как прогнозирование, классификация и кластеризация. Также нужно сказать о вспомогательных задачах, особенно фильтрация и поиск аномалий. К этим задачам мы еще вернемся в курсе.

Лекция 2

Лекция сопровождается презентацией.

Тема: Статистический анализ временных рядов.

Вопросы в лекции:

- Основные статистические характеристики временных рядов.
- Понятие автокорреляционной функции.
- Анализ стационарности,
- Особенности анализа временных рядов как статистической задачи.

В данной лекции следует уделить внимание статистическим характеристикам временных рядов. Следует показать, что значат такие термины, как среднее и дисперсия. Важно сказать о том, что такое распределение и как его оценить по экспериментальным данным. Также важно показать, что значат для распределения такие понятия, как среднее, стандартное отклонение, мода, медиана, коэффициент асимметрии и коэффициент эксцесса. Важно отметить роль нормального распределения. Нужно сказать о том, что для стационарного процесса с неограниченным диапазоном величин мы должны иметь нормальное распределение, а для ограниченного диапазона равномерное. Это справедливо, если нет оснований полагать иное. Также можно отметить, что для многомерной величины кроме дисперсии следует рассматривать и ковариации как моменты второго порядка.

Одним из самых важных понятий данной лекции является автокорреляционная функция. Нужно пояснить ее на примерах. Также следует отметить разницу между корреляцией и ковариацией – эти понятия часто путают, но в анализе временных рядов принято, что корреляция нормирована на дисперсию. Также следует пояснить что есть взаимная корреляция и где она используется (например, для многомерных рядов). В связи с данными терминами следует также повторить понятие лага. Нужно также сказать, что правильно считать взаимную корреляцию для лагов в положительном и в отрицательном направлениях (иначе максиму можно не найти). Также полезно пояснить, что бывают корреляции по полной шкале лагов и по половинной (чтобы вход был одного размера с входом). Можно упомянуть, что иногда рассматривают отдельно смещенную и несмещенную корреляцию. Также можно пояснить, понятие свертки на базе понятия корреляции. Важно также сказать о том, что такое корреляционный коэффициент, как он связан с расстоянием косинуса, что значит коэффициент +1; -1 и 0.

Вторым важным понятием данной лекции является вопрос стационарности. Следует еще раз (см. лекция 1) отметить, что значит данное понятие, привести примеры разных видов нестационарности. Особенно важно сказать о том, как можно привести нестационарный ряд к стационарному виду. Но нужно сказать, что это может не всегда работать.

В свете стационарности следует разъяснить что такое стационарный и нестационарный шум. Также нужно показать, что такое белый гауссов шум; нужно объяснить почему это самый важный шум (например, по теореме больших чисел); можно также показать, какие еще бывают шумы.

В конце лекции можно освятить особенности анализа временных рядов, например, сказать о том, какие бывают методы анализа временных рядов, как мы оцениваем точность при анализе, что такое остаток (невязка) при анализе и почему для нее важен анализ стационарности. Данная часть лекция опциональная, поэтому на нее время можно отвести по остаточному принципу.

Лекция 4

Лекция сопровождается презентацией.

Тема: Модели авторегрессии скользящего среднего.

Вопросы в лекции:

- Специфика использования авторегрессии (AR) и скользящего среднего (MA) и объединенной модели АРСС (ARMA);
- Интегрированная модель ARMA (ARIMA) и ее использование в анализе временных рядов;
- Сезонная интегрированная модель ARMA (SARIMA) и ее использование в анализе временных рядов;
- Особенности выбора порядка моделей ARMA и других;
- Обзор других моделей на базе ARMA;
- Примеры решения задач анализа временных рядов с использованием ARMA.

Лекцию следует начать с того, что модели авторегрессии-скользящего среднего (АРСС) – это один из основных инструментов анализа временных рядов по настоящее время. Модели АРСС хорошо работают для сравнительно не больших выборок в условиях стационарности или сравнительно не сложной не стационарности. Модели лучше работают для одномерных временных рядов, однако, есть варианты и для многомерных рядов. При этом в условиях своей применимости модели АРСС как правило показывают результаты лучше, чем методы машинного обучения. Однако, основной недостаток модели – это сложность выбора порядка модели.

Касательно простой модели АРСС важно пояснить понятие порядков модели (авто регрессионного и скользящего среднего). Также можно сказать откуда берутся два этих компонента. После нужно сказать, что простая модель АРСС работает только для стационарного ряда. Поэтому в анализе временных рядов чаще работают с моделями интегрированного АРСС (ARIMA).

Про ARIMA нужно сказать о том, как выбрать порядок дифференцирования и почему нужно его выбрать (убрать тренда, сделать ряд стационарным). Также можно пояснить что такое лаговая форма записи ARIMA и напомнить о понятие лаг. Также важно сказать о том, как проверять модель ARIMA на стационарность, какие бывают тесты (напомнить лекцию 2 и 3). После можно сказать о предварительном выборе порядка моделей по графикам ACF и PAFC (но можно и не говорить, так как далее мы рассмотрим этот вопрос подробней).

Важно сказать, что в ряде случаев недостаточно дифференцировать тренд. Такая ситуация может иметь место, если влияние сезонности слишком высоко или она нестационарная. В этом случае нужно ввести сезонную производную. Модель с такой производной называется SARIMA. По данной модели надо сказать, о порядках модели и правилах выбора порядков модели. Также полезно рассмотреть несколько примеров выбора порядков модели.

В конце лекции можно сказать о других моделях АРСС, в том числе о VAR и об экзогенных и эндогенных факторах.

Тема: Статистический анализ временных рядов.

Вопросы в лекции:

- Анализ невязок (остатков).
- Фильтрация скользящим средним и другие методы сглаживания.
- Линейный регрессионный анализ временных рядов;
- Обзор возможностей робастной статистики;
- Особенности моделей нелинейной адаптивной регрессии.

Одним из самых важны вопросов данной лекции является анализ остатков (невязок) модели. Следует еще раз объяснить (см. лекцию 2) почему этот вопрос так важен. Следует освятить ряд методов анализа остатка и пояснить примеры для графического метода, метода анализа автокорреляционной функции, Q-Q диаграмма и численные тесты. Также важно пояснить понятие частичной автокорреляции и ее использование для анализа стационарности.

Вторым важным вопросом данной лекции являются методы сглаживания. Нужно сказать, что данные методы –это наиболее простые методы анализа временных рядов. Данные методы относятся к классу не параметрической статистики, они хорошо работают в случае белого гауссова шума. Для каждого типа данных методов можно дать примеры использования. Также нужно сказать, что слишком сильное сглаживание может испортить ряд. То есть сглаживание – это низкочастотная фильтрация, можно потерять в.ч. информацию, или, например, сделать из белого шума окрашенный. Также важно отметить, зачем нужно использовать двойное и тройное экспоненциальное сглаживание (модели с трендом и с трендом + сезонностью). Также следует отметить какие еще бывают модели в данном классе (общий класс Error-Trend-Seasonality). Нужно сказать, что модель нужно подбирать.

В конце лекции можно отметить, регрессионный подход к анализу временных рядов. Рассказать о линейной регрессии, нелинейной регрессии, сказать, что нейронные сети — это тип нелинейной регрессии. Можно сказать, об адаптивных моделях как типе регрессионной задачи. Также можно сказать об обобщенной адаптивной модели, и одном из ее примеров модели fb prohet – про нее нужно сказать, что она также тип нелинейной регрессии в конкретном приложении (бизнес-процессы).

Лекция 5

Лекция сопровождается презентацией.

Тема: Анализ и обработка признаков во временных рядах.

Вопросы в лекции:

- Признаки в анализе временных рядов.
- Особенности разведочного анализа данных;
- Некоторые методы для представления признаков в временных рядах;
- Рассмотрение методов извлечения признаков из временных рядов;
- Некоторые методы обработки и выбора признаков в временных рядах.

Лекция посвящена введению в тему признаков временных рядов. В ходе лекции следует разъяснить слушателям термин признак, типы признаков, которые бывают во временных рядах и методы их обработки. В том числе, лекция показывает следующие приемы. Приемы разведочного анализа временных рядов с целью получения информации о структуре ряда. Примеры представления данных во временных рядах и методы преобразования данных во временных рядах. Методы извлечения признаков из временных рядов и методы их отбора. В том числе в лекции освещены некоторые методы оценки важности признаков при их отборе.

Среди методов, освященных в лекции следует уделить больше внимания вопросам предварительного анализа временного ряда, техникам выделения признаков и техникам их отбора. Если остается время, то можно более подробно разъяснить вопросы частотного и время-частотного представления временного ряда, в том числе вопросы частотной фильтрации и декомпозиции. Эти вопросы, хотя и не являются базовыми, позволяют провести глубокий анализ временного ряда.

Среди вопросов отбора признаков предпочтительно разъяснить методы оценки их важности и встраиваемые методы.

Среди вопросов представления временного ряда предпочтительно разъяснить как перейти к проблеме одношагового прогноза. Такой подход в ряде случаев позволяет получить результаты лучше, чем предсказание выборок последовательности.

Лекция 6

Лекция сопровождается презентацией.

Тема: Специфика методов машинного обучения при анализе временных рядов.

Вопросы в лекции:

- Специфика анализа временных рядов с использованием методов машинного обучения;
- Обзор некоторых проблем анализа временных рядов и их решения с использованием методов машинного обучения;
- Метрики временных рядов;
- Обзор задач кластеризации временных рядов;
- Обзор функций расстояний для временных рядов.

Лекция представляет первую из двух частей, посвященных данной теме. В данной лекции рассмотрены в целом примеры использования методом машинного обучения в анализе временных рядов. В начале лекции возможно следует напомнить слушателям некоторые термина из области машинного обучения. В том числе, термины: обучение с учителем и без учителя, полу-контролируемое обучение; тренировочная, валиадционная и тестовая выборки. Возможно, что также следует указать на основные достоинства и недостатки машинного обучения. В том числе указать проблему переобучения, проблему недостаточности данных и проблему недостаточности интерпретируемести. Также нужно сказать о достоинствах машинного обучения, в том числе работа со сложными данными без модели, решение задач классификации и т.д., кластеризация и сжатие данных. В лекциях показаны примеры таких задач.

Во второй части лекции рассмотрены вопросы кластеризации с использованием методов машинного обучения и основные виды расстояний. Среди прочего следует подробно рассказать о расстояние DTW.

Если останется время можно повторить метрики точности и рассказать подробней о некоторых методах кластеризации. Также можно повторить, что такое эвклидово и косинус расстояние.

Лекция 7

Лекция сопровождается презентацией.

Тема: Специфика методов машинного обучения при анализе временных рядов.

Вопросы в лекции:

- Методы поиска аномалий во временных рядах;
- Специфика проблем классификации временных рядов и методов их решения;
- Специфика задач регрессии для временных рядов и методов их решения с помощью машинного обучения.

Лекция представляет вторую из двух частей, посвященных данной теме. В первой части данной лекции рассмотрены проблемы поиска аномалий во временных рядах. Тут надо рассказать о том, почему это важно и что нужно интерпретировать аномалии. Среди методов поиска аномалий можно уделить больше внимания методам BoxPlot, модельным методам и автоэнкодеру, DBSCAN и изоляционному лесу. В том числе пояснить как тут реализуется принцип полу-контролируемого и контролируемого обучения. В том, что касается обучения с учителем важно упомянуть о проблему дисбаланса классов.

Вторая часть лекции посвящена методам машинного обучения с учителем. В этой части важно сказать о специфики методов классификации в анализе временных рядов. Стоит еще раз объяснить, что классические методы машинного обучения в ряде случаев работают с временными рядами не очень хорошо. Это связано с тем, что в классике не учитываются временные зависимости в ряде. Также классика рассматривает каждую точку ряда отдельно. При этом ряд часто избыточен (следует из теоремы Кательникова). Поэтому для классификации нужно извлекать признаки.

Что касается методов классификации нужно сказать, что метод kNN-DTW можно рассмотреть, как base-line в этих приложениях. Также важно сказать о том, какие признаки используются и как извлекаются. Также нужно сказать, что в данной сфере часто используют ансамбли методов. Это связано с вопросами стабильности результатов. В ряде случаев ансамбль классики предпочтительней одной нейронной сети.

В том, что касается регрессии надо сказать о применимости методов машинного обучения. Они используются для больших выборок с сложными зависимостями. В других случаях классическая статистика будет работать лучше.

Лекция 8

Лекция сопровождается презентацией.

Тема: Использование методов глубокого обучения в анализе временных рядов.

Вопросы в лекции:

- Специфика методов глубокого обучения среди других методов машинного обучения.
- обзор особенностей обучения глубоких нейронных сетей в приложениях для анализа временных рядов.
- Рассмотрение перспектив и текущего состояния некоторых архитектур полносвязных нейронных сетей;
- Рассмотрение перспектив и текущего состояния некоторых архитектур рекуррентных нейронных сетей и их использования в анализе временных рядов;
- Одномерный сверточные нейронные сети и их использование в анализе временных рядов;
- Механизм внимания и его использование в архитекторах нейронных сетей для анализа временных рядов.

Лекция представляет собой продолжение второй части лекции 7 в некотором смысле. В данной лекции надо объяснить применимость нейронных сетей глубокого обучения в анализе временных рядов. Нужно сказать о том, что это не надо рассматривать как панацею. Однако, в случаях очень больших выборок со сложной структурой сети могут дать хороший выигрыш в точности. Следует повторить слушателям, что тут тоже можно использовать ансамбли сетей.

Среди подходов к глубокому обучению нужно объяснить, что наиболее распространённым следует считать одномерные сверточные сети. Однако, также важно разъяснить механизм внимания и трансформеры. Последние могут дать лучший результат при очень больших выборках и необходимости учета долговременного контекста (то есть при учете очень низкочастотных процессов). Также следует упомнить о рекуррентных сетях, их тоже часто используют. Отметим, что в лекции дано подробное разъяснение механизма внимания, это сделано на случай, если эта тема не затрагивалась в других курсах. В ином случае подробности можно опустить.

Лабораторная работа 1

Работа сопровождается методическими указаниями.

Разведывательный анализ временных рядов.

План работ

Разведывательный анализ временных рядов. Знакомство с библиотекой Pandas и методами работы с временными рядами в ней. Знакомство с библиотекой seaborn и методами визуализации временных рядов.

Импорт Pandas и вспомогательных библиотек

```
try:
import pandas
except:
!pip install pandas
finally:
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates#Date Parser
try:
import seaborn
except:
!pip install seaborn
finally:
import seaborn as sns
# Use seaborn style defaults and set the default figure size
sns.set(rc={'figure.figsize':(11, 4)})
```

Начальный анализ библиотеки Pandas

Набор данных

Рассмотрим набор данных Open Power Systems Data. В наборе данных производство и потребление электричества декларируется как общее ежедневное потребление в Гига Ваттах в час (GWh).

В файле данные разбит на колонки как:

• Date — дата в формате (гггг-мм-дд)
• Consumption — Потребление в ГВт (GWh)
• Wind — Производство веторэнергии в ГВт (GWh)
• Solar — Производство солнечной энергии в ГВт (GWh)
• Wind+Solar — Суммарное производства по двум предыдущим столбцам GWh

```
path_ts = 'https://github.com/jenfly/opsd/raw/master/opsd_germany_daily.csv'
df = pd.read_csv(path_ts)
df.sample(15, random_state=0)
Date Consumption Wind Solar Wind+Solar
print(df.shape)
(4383, 5)
```

Для получения информации по данным можно использовать метод `info()`:

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4383 entries, 0 to 4382
Data columns (total 5 columns):
# Column Non-Null Count Dtype
--- ------ -------------- -----
0 Date 4383 non-null object
```

Для начала заменим столбец индексов на данные

```
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
```

```
df.info()
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 4383 entries, 2006-01-01 to 2017-12-31
Data columns (total 4 columns):
# Column Non-Null Count Dtype
--- ------ -------------- -----
0 Consumption 4383 non-null float64
1 Wind 2920 non-null float64
2 Solar 2188 non-null float64
3 Wind+Solar 2187 non-null float64
dtypes: float64(4)
memory usage: 171.2 KB
```

Посмотрим некоторые итоги анализа при помощи метода `describe()`

```
df.describe()
Consumption Wind Solar Wind+Solar
count 4383.000000 2920.000000 2188.000000 2187.000000
mean 1338.675836 164.814173 89.258695 272.663481
std 165.775710 143.692732 58.550099 146.319884
min 842.395000 5.757000 1.968000 21.478000
25% 1217.859000 62.353250 35.179250 172.185500
50% 1367.123000 119.098000 86.407000 240.991000
75% 1457.761000 217.900250 135.071500 338.988000
max 1709.568000 826.278000 241.580000 851.556000
```

Набор данных имеет 4383 строк, в период с 1 января 2006 до 31 декабря 2017.

Для просмотра некоторых данных можно использовать методы `head()` и `tail()` для первых и последних нескольких строк.

```
df.head(3)
Consumption Wind Solar Wind+Solar
Date
df.tail(3)
Consumption Wind Solar Wind+Solar
Date
```

можно проверить тип данных для каждой колонки

```
df.dtypes
Consumption float64
Wind float64
Solar float64
Wind+Solar float64
dtype: object
```

Методы обращения к данным по временным меткам

Теперь можно загрузить данные сразу с использованием специального метода интерпретации данных. Назначим колонки данных как индексы при чтении. Также можно отметить, что временные метки можно сразу поделить на части: `year`, `month` и `day`.

```
df = pd.read_csv(path_ts, parse_dates=['Date'], index_col="Date")
df.head()
Consumption Wind Solar Wind+Solar
Date
```

Теперь можно обращаться к данным по их индексам.

```
df.loc['2017-08-10']
Consumption 1351.491
Wind 100.274
Solar 71.160
Wind+Solar 171.434
Name: 2017-08-10 00:00:00, dtype: float64
```

или по диапазонам индексов

```
df.loc['2014-01-20':'2014-01-22']
Consumption Wind Solar Wind+Solar
Date
```
также можно обращаться к отдельным столбцам в заданном диапазоне индексов с помощью метода loc:
```
df.loc['2014-01-20':'2014-01-25', 'Wind']
Date
Name: Wind, dtype: float64
```
или обращаться к столбцам как методам.
```
df.Wind.loc['2014-01-20':'2014-01-25']
Date
2014-01-20 78.647
2014-01-21 15.643
2014-01-22 60.259
2014-01-23 125.177
2014-01-24 106.527
2014-01-25 145.786
Name: Wind, dtype: float64
```
Также можно обращаться к столбцам как к ключевым словам:
```
df[['Wind']].loc['2014-01-20':'2014-01-25']
Wind
Date
2014-01-20 78.647
2014-01-21 15.643
2014-01-22 60.259
2014-01-23 125.177
2014-01-24 106.527
2014-01-25 145.786
df['Wind'].loc['2014-01-20':'2014-01-25']
Date
2014-01-20 78.647
2014-01-21 15.643
2014-01-22 60.259
2014-01-23 125.177
2014-01-24 106.527
2014-01-25 145.786
Name: Wind, dtype: float64
```
Для удобства в фреймворке Pandas также предусмотрен метод обращения по элементам массива:
```
df.iloc[0:2,0:3]
Consumption Wind Solar
Date
2006-01-01 1069.184 NaN NaN
2006-01-02 1380.521 NaN NaN
```
Если индексы имеют формат DateTimeIndex можно использовать следующие способы получения временным меток:
```
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=4383)
Int64Index([6, 0, 1, 2, 3, 4, 5, 6, 0, 1,
...
4, 5, 6, 0, 1, 2, 3, 4, 5, 6],
```

```
dtype='int64', name='Date', length=4383)
Int64Index([2006, 2006, 2006, 2006, 2006, 2006, 2006, 2006, 2006, 2006,
...
2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017],
dtype='int64', name='Date', length=4383)
```

Также можно представлять данные с разной частотой, например с частотой `D` -для дней, `W`,`M`,`Y` для недель, месяцев и лет соответственно.

```
df[['Wind']].loc['2014-01-20':'2014-02-25'].asfreq('W')
Wind
Date
```

Например, можно взять месяц из временных меток и посмотреть, как он будет выглядеть в недельном представлении:

```
df.loc['2012-02'].asfreq('W')
Consumption Wind Solar Wind+Solar
Date
```

Можно также использовать диапазонный тип обращений, как для типа лист в питоне.

```
df.loc['2012':].asfreq('Y')
Consumption Wind Solar Wind+Solar
Date
```

Вот пример для изученных обращений:

```
df.loc['2012':].asfreq('Y').set_index(df.loc['2012':].asfreq('Y').index.year)
Consumption Wind Solar Wind+Solar
Date
```

Также если индексы имеют формат дат, то можно их сгруппировать по заданным частотам, например `W`-неделя, 'Y' и `A` - год, и т.д. Также можно провести группировку по, например, `'2y'` - по 2м годам. Методы `groupby`, `resample` и `asfreq` могут быть использованы для группировки. Как правило после группировки могут быть подведены некоторые итоги, например, `sum`, `mean`, `median` или `std`. Например, так, как это показано ниже.

```
df.groupby(pd.Grouper(freq='1y')).sum()
Consumption Wind Solar Wind+Solar
Date
df.resample('1w').median().head(3)
Consumption Wind Solar Wind+Solar
Date
df.asfreq('1w').head(3)
Consumption Wind Solar Wind+Solar
Date
```

Чистка данных

При анализе данных необходимо как минимум исключить из них отсутствующие значения, которые обозначаются как `NaN`. На самом деле `NaN` могу быть исключены при помощи методов `ffill` или `bfill`. Например, можно сделать так`.asfreq('D', method='ffill')`. Также можно использовать методы `dropna` и `filna`. Но для начала давайте посчитаем число `NaN` в наборе данных. Отметим, что также можно использовать метод `isnull` вместо `isna` с тем же эффектом.

```
df.isna().sum()
Consumption 0
Wind 1463
Solar 2195
Wind+Solar 2196
dtype: int64
```

Теперь можно заменить отсутствующие значения

Отметим, что значения также можно выкинуть `NaN` при помощи метода `dropna`.

```
df.fillna(0, inplace=True)
df.head(3)
Consumption Wind Solar Wind+Solar
```

```
Date
2006-01-01 1069.184 0.0 0.0 0.0
2006-01-02 1380.521 0.0 0.0 0.0
2006-01-03 1442.533 0.0 0.0 0.0
```

Упражнение 1.

1. Откройте набор данных из примеров с индексами - значениями временных меток.
2. Добавьте к набору данных колонки `Year`, `Weekday` и `Month`.
3. Добавьте колонку `'other sources'` представляющую разность `total` и `wind+solar`.
4. Добавьте колонку `'ratio of Wind+Solar'` с отношением значений колонки `Wind+Solar` и `total`.
5. Создайте еще один набор данных без значений `NaN`.

Визуализация данных в Pandas и Seabron

Библиотека Pandas включает ряд методов из библиотеки matplotlib

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

Теперь давайте посмотрим более развернутый пример использования данного метода

```
cols_plot = ['Consumption', 'Solar', 'Wind']
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)')
```

Также может быть полезно визуализировать общее потребление и потребление по каждому типу (ветер, солнце) вместе. *Отметим, что* параметр min_count заменяет значения `NaN` на 0.

```
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)');
```

Теперь можно провести анализ полученных графиков: Графики Consumption, Solar, и Wind временных рядов осциллируют между локальными минимумами и максимумами в течение каждого года, что соответствует сезонным изменениям в погоде в году. Потребление электричества наибольшее зимой и наименьшее летом, что вполне логично. Можно разделить потребление на два кластера — медленные и интенсивные осцилляции, менее интенсивный и более быстрый разброс. Предположим, что такие осцилляции связаны с днями недели и месяцами. Выбросы на графике 2 связаны с праздничными и другими особыми днями
Солнечное электропроизводство максимально летом и минимально зимой. Ветренное потребление наоборот, что также логично. можно также увидеть растущий тренд в потреблении ветренной и солнечной энергий.
Описанная выше 2-х кластерное поведение общего энергопотребления может быть дополнительно выявлено при построении гистограммы. Это показано ниже, видно два кластера с пиками порядка 1100 и 1400 ГВт Отметим, что иногда полезно аппроксимировать гистограмму, например, это можно сделать следующим методом `ax = df.Consumption.plot(kind='kde' )`

```
ax = df.Consumption.hist(bins=30)
ax.set_ylabel('Count of samples')
ax.set_xlabel('GWh consumption')
plt.show()
```

Упражнение 2

1. Проведите визуальный анализ созданных в предыдущем упражнении столбцы 'other' и 'ratio'.
Опишите итоги анализа полученных визуализаций.
2. Проведите сравнение результатов для столбцов 'ratio' и 'consumption'.

Анализ сезонности

Сезонность и тренд являются двумя наиболее важными составляющими временных рядов. Однако, сезонность как правило бывает не однородной. Пример ниже показывает такую неоднородность:

```
ax = df.loc['2016-05':, 'Consumption'].plot()
ax.set_ylabel('Daily Consumption (GWh)');
```

Следующий график показывает недельные осцилляции в течение нескольких лет. На этом графике видно, что потребление электричества ниже всего в период январских праздников.

```
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()
```

Иногда полезно проверить как эти провалы выглядят

```
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()
```

На графике выше видно, что потребление имеет также провалы каждые выходные.

Анализ тренда

Предыдущий анализ показал наличие явного тренда в потреблении ветра и других параметров. Для выявления тренда может быть использована несколько способов. Один из наиболее простых и популярных методов это скользящее среднее (rolling average).

Отметим наиболее точные результаты будут достигнуты, если период скользящего будет совпадать с периодом сезонности. df.Wind.rolling(365) Но лучше использовать специальные значения скользящего, например **data frequency**, например, `'365d'`

```
df[['Wind']].rolling('365d').mean().plot( linewidth=1.5, );
```

Также мы можем проверить тренд при помощи использования типа графика boxplot - для группировки данных по различным временным периодам и визуализации результатов по группам.

```
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()
```

Альтернативное представление для boxplot - это violinplot, который однако, имеет туже интерпретацию.

```
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()
```

также можно построить т.н. barplot.

```
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');
```

В дополнение к предыдущему можно использовать специальные методы

скользящего среднего, например, экспоненциального скользящего среднего (EW).

```
df.Wind.ewm(halflife=365, min_periods=0,adjust=True).mean().plot()
df.Wind.rolling(365).mean().plot();
```

Еще о сезонности

Также приведенные выше методы позволяют выделить сезонность, например, путем скользящего среднего по 30 дней.

```
df.loc['2010':, 'Wind'].rolling('30d').mean().plot( linewidth=1.5, );
```

также можно использовать метод resample для анализа графиков.

```
start, end = '2010-01', '2017-06'
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();
```

График выше показывает некоторую нестабильность сезонности. Такое поведение можно также проверить с использованием barplot.

```
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()
```

График выше показывает, что ветренная энергетика имеет много выбросов, их можно объяснить например, экстремальными выбросами, штормами или другими эффектами.

```
df.loc[str(year):str(year)+'-12', 'Wind'].resample('30d').mean()
Date
2014-01-01 203.955533
2014-01-31 209.698400
2014-03-02 155.243172
2014-04-01 118.409033
2014-05-01 119.957933
2014-05-31 83.773967
2014-06-30 76.262933
2014-07-30 108.607600
2014-08-29 84.810200
2014-09-28 120.336133
2014-10-28 110.850200
2014-11-27 290.058600
2014-12-27 160.800800
Freq: 30D, Name: Wind, dtype: float64
```

Также ниже показан пример анализа сезонности с использованием сегментов данных (например, по годам).

```
for year in list(set(df.index.year))[-4:]:
plt.plot(df.loc[str(year):str(year)+'-12',
'Wind'].rolling('30d').mean().values, label=year)
plt.legend()
<matplotlib.legend.Legend at 0x1c8b0988d68>
```

Тут видно, что сезонность не стационарна и нестабильна с растущей интенсивностью от года к году. Как было упомянуто выше в анализируемых данных можно выделить два типа сезонности: дневную и месячную. Пример ниже показывает

анализ сезонности по дням недели.

```
ax = sns.boxplot(data=df, x=df.index.weekday, y='Consumption');
ax.set_ylabel('GWh')
ax.set_xlabel('week_day')
ax.set_title('Wind')
plt.show()
```

На графике ниже не видно, что потребление энергии наименьшее в выходные.
О*тметим* нулевой день тут - понедельник.

Упражнения 3

1. Проверьте гипотезу о снижении потребления солнечной и ветряной энергии в выходные.
2. Исследуйте графики consumption и solar временных рядов о их тренде, месячной и недельной сезонности.
3. Сделайте предположение о сезонности и тренде для колонки other.
4. Проведите анализ колонки Wind+solar и колонки ratio.__

Лабораторная работа 2

Работа сопровождается методическими указанями.

Моделирование временных рядов

Моделирование временных рядов. Детерминистические модели. Основные типы трендов. Модели сезонности. Регулярные и нерегулярные события.
Стохастические модели временных рядов. Понятие белый гауссов шум. Нестационарные шумы. Модель временного ряда со случайным блужданием.

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

Модель временного ряда

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

$y(t) = a0 + trend(t) + cyclic(t) + seasonal(t)$

где

· y(t) - это временной ряд - набор выборок, проиндексированных некоторой переменной t, обычно t – это временные отметки, если временной шаг дискретный, он также может быть обозначен как n (номер выборки), в этом случае в реальном времени - значение шага будет соответствовать $t = n \cdot Ts$, где Ts - период шага n (период дискретизации, с которым берутся отсчеты).

· $a0$ - некоторый начальный постоянный уровень,

· $trend$ - это наличие некоторого тренда, который является частью зависимости с медленным изменением.

· $seasonal$ - это сезонность или некоторые «относительно быстро изменяющиеся» периодические составляющие - это относительно быстро меняющаяся часть взаимосвязи.

· $cyclic$ - 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

$y(t) = trend(t) + seasonal(t)$.

Trend investigation

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 $y(t) = a \cdot t + b$

· параболический тренд $y(t) = a \cdot t2 + b \cdot x + c$

· полиномиальный тренд y_____(t) = $a \cdot tb + c$

· гиперболический тренд $y(t) = a$

$tb+c$

$+ d$

· экспоненциальный тренд $y(t) = \exp(a \cdot t + b)$
· насыщение (логистический) тренд $y(t) = c$
$1+exp(-k(t-m))$
· логарифмический тренд $y(t) = c\log b\ (a \cdot t)$
· многие другие функции, которые, как правило, сглажены, очень медленно меняются или даже монотонны.
Теперь мы можем попробовать логарифмический тренд с основанием e (Число Эйлера, натуральный логарифм) и $a = 4$.

```
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 Facebook Prophet,
$y(t) = (k + a(t)T\ \delta)t + m + a(t)T\ \gamma$,
где
· $a(t)$ матрица изменения роста, описывающая точки перегиба tj (матрица с единицами),
· k постоянная скорости роста,
· m смещение,
· δ вектор изменения скорости роста,
· γ коэффициент изменения роста $\gamma j = tj\delta j$,
· sj точки перегиба.
Notes:
В простейшем случае модель сводится к $y(t) = kt + m$ для временного ряда без точек перегиба tj .
В модели Facebook Prophet предложили рассматривать логистическую модель как альтернативу линейной, в этом случае тренд можно представить в виде
$y(t) = c(t)$
$1+exp(-(k+a(t)T\delta)(t-m-aT\gamma))$

.
```
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 matr
ix 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 adjust
ments
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()
```

Упражнение 1

· Реализуйте модель логистического тренда Facebook Prophet

$y(t) = c(t)$
$1+exp(-(k+a(t)T\delta)(t-m-aT\gamma))$

Простейшую сезонную часть временного ряда можно представить в виде

$s(t) = a \cdot sin($
$2\pi t$
T
$+ \theta 0) = a \cdot sin($
$2\pi nTs$
T
$+ \theta 0) = a \cdot sin($
$2\pi fn$
fs
$+ \theta 0),$

где:

· a интенсивность сезонной компоненты;

· T период сезонности (месяц, день, неделя и т.д.);

· $\theta 0$ начальный сдвиг (начальная фаза) сезонности;

· Ts период дискретизации;

· f и fs частота сезонности ($f = 1/T$) и частота дискретизации $fs = 1/Ts$.

Отметим, что в соответствии с теоремой Шеннона-Найквиста-Котельникова минимальное значение fs должно быть $fs \geq 2f \rightarrow T \geq 2Ts$. Для оценки приведенного количества периодов используйте $Nperiods = N \cdot T\_s/T$

```
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
```

Теперь давайте смоделируем более сложную сезонность в году, например, месяц и неделю, которые мы сделаем аддитивно, так что:

$seasonality = \Sigma ai$

M

$i{=}0$

$\cdot\, sin(2\pi nTs/Ti + \theta i\,)$

```
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()
```

Упражнение 2

· Для предыдущей модели добавьте квартальную сезонность.

Теперь мы можем промоделировать аддитивные и мультипликативные временные ряды.

$y(t) = seasonality(t) + trend(t)$

В нашем случае смоделируем их так:

$y(t) = biastrend + atrend\ nTs + am \cdot sin(2\pi nTs/Tm) + aw \cdot sin(2\pi nTs/Tw)$

```
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()
```

Упражнение 3

· Реализовать мультипликативную детерминированную модель временных рядов с сезонной частью и частью тренда.

· Реализовать модель логистического тренда с аддитивной сезонностью.

Циклическая часть

Помимо тренда и сезонности, мы можем добавить некоторую цикличность (в качестве альтернативы можно рассматривать как медленное изменение тренда). Давайте смоделируем цикличность 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 можем добавить это к 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()
```

Упражнение 4

· Реализуйте мультипликативную детерминированную модель временных рядов с

сезонной, циклической и трендовой частями.

Моделирование редких и регулярных явлений

Помимо тренда и регулярной сезонности, в модель временных рядов могут быть внесены конкретные события. Например, если мы моделируем временные ряды продаж, будет интересно добавить изменение спроса в будние и выходные дни. В простейшем случае это можно сделать так:

week_days$(day) = \Sigma\, ai$
$7i$
$=1\ \delta(\lfloor(day-1)/7\rfloor + 1 - i)$,
где
· $\lfloor day/7\rfloor$ - остаток деления;
· δ дельта функция Кронокера,
$\delta(i,j) = \delta(i-j) = \{$
1 if$i = j$
0 otherwise
· i номер дня недели ($i = 1,2,3,4,5,6,7$).

Отметим, что если необходимо считать не с 1, то можно описать их влияние следующим образом, с использованием переменной shift:

week_days$(day) = \Sigma\, ai$
7
$i=1$
$\delta(\lfloor(day-1+shift-1)/7\rfloor + 1 - i)$,

Давайте проверим 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]
```

Теперь можно ввести коэффициенты от дня *aweek*

```
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()
```

Давайте проверим как дневные изменения выглядят на фоне линейного тренда
$y(day) = biastrend + atrend\ day + atrend\ week\_days$,
где *week_days* номер дня.

```
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()
```
Теперь добавим сезонную часть.
```
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()
```
Упражнение 5
· добавить падение спроса в праздничные дни в начале года к временным рядам, реа лизованным выше примере.
Симуляция случайного поведения
Белый гауссов шум
Помимо детерминированной части временного ряда, полезно смоделировать его стохастическое поведение. Стохастическое поведение временного ряда, в первую очередь связанное с влиянием шума. Самая простая и наиболее распространенная модель шума - это белый гауссовский шум (WGN) (почти что тоже самое, что и модели независимого и одинаково распределенного (i.i.d) шума). WGN имеет нормальное распределение с нулевым средним значением и дисперсией $\sigma 2$:
$noise(t) \sim N(0, \sigma)$;
Нормальное распределение имеет вид:
$P(t) = N(0, \sigma) =$
1
$\sigma\sqrt{2\pi exp}\ (-$
$t2$
$2\sigma 2)$
где σ средний разброс шумов или их стандартное отклонение (корень дисперсии).
Построим модель такого шума.
```
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()
```

Теперь посмотрим на влияние шумов 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 равномерно распределенного шума, соответствующего стационарной модели шума, важно моделировать нестационарные случаи. Самый простой случай - это линей но возрастающая дисперсия,

```
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()
```

Exercise 6

· Исследовать влияние аддитивного стационарного и нестационарного белого шума на временные ряды с сезонными частями и частями тренда в следующей форме
$y(t) = a0 + trend(t) + seasonal(t) + cyclic(t) +$ rare_events(t) $+ noise(t)$.
· _____Смоделировать модель временного ряда в следующем виде:
$y(t) = a0 + (trend(t) \cdot cyclic(t) + seasonal1(t)) \cdot seasonal2(t) + noise(t)$.

Модель случайного блуждания

Помимо аддитивного шума, важной моделью шума является случайное блуждание, которое в простейшем случае можно смоделировать как
$y(t) = y(t-1) + \eta(t)$,
где $\eta(t) \sim N(0, \sigma 2)$. Такая модель широко распространена во многих бизнес процессах и не только.

```
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()
```

Упражнение 7

1. Исследуйте 3 модели случайного блуждания:
· Модель с дрифтом:
$y(t) = \alpha + y(t-1) + \eta(t)$
· Модель с трендом:
$y(t) = \alpha + \beta \cdot t + y(t-1) + \eta(t)$
· Модель с изменением интенсивности:
$y(t) = y(t-1) + \eta(t), \eta(t) \sim N(0, \sigma 2(t)), \sigma 2(t) = \sigma 0 + \gamma \cdot t$__

Лабораторная работа 3

Работа сопровождается методическими указаниями.

Введение в statsmodels. Непараметрический анализ временных рядов.

Знакомство с библиотекой статистического анализа временных рядов statsmodels.tsa. Разложение временных рядов. Методы непараметрического предсказания временных рядов. Методы скользящего среднего.

Импорт данных.

Statsmodels фреймворка python - один из самых популярных инструментов исследователей для решения многих задач статистического анализа. Одним из наиболее интересных модулей этой библиотеки является statsmodels.tsa, описание которого вы можете найти здесь: https://www.statsmodels.org/stable/tsa.html

```
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={'font.size': 15})
%matplotlib inline
try:
import statsmodels.api as sm
except:
!pip install statsmodels
finally:
import statsmodels.api as sm
import statsmodels
```

В этой работе мы будем работать со следующим набором данных

```
airpass = sm.datasets.get_rdataset("AirPassengers", "datasets")
airpass = pd.DataFrame(airpass.data["value"])
airpass.index = pd.date_range(start = "1949-01", periods =
len(airpass.index), freq = "M").to_period()
airpass.index = airpass.index.to_timestamp()
airpass=airpass.rename(columns={"value": "data"}, inplace = False)
airpass.plot();
```

Введение в StatsModels

Для визуализации временных рядов с помощью StatsModels воспользуемся модулем sm.graphics. Для начала проанализируем функцию автокорреляции (АКФ, ACF). АКФ - это степень зависимости текущих значений временного ряда по отношению к его отстающей версии самого себя

$cor(k) =$

1

N

$\Sigma N-1($

$i=0\ yk - ev)(yi-k - ev)$

$var(y)$

Ниже взяты первые 20 лагов (начиная от 1-го ($k = 1$))

```
#Plot the ACF:
lags = 20
sm.graphics.tsa.plot_acf(airpass,
lags = lags,
zero = False)
plt.xticks(np.arange(1, lags + 1, 1.0));
plt.show()
```

Помимо полной корреляции временного ряда пакет StatsModels позволяет оценить его

частичную корреляционную функцию (PACF). «Частичная» корреляция между двумя переменными - это степень корреляции между ними, которая не объясняется их взаимной корреляцией с заданным набором других переменных. Например, если мы регрессируем переменную Y по другим переменным X1, X2 и X3, частичная корреляция между Y и X3 - это степень корреляции между Y и X3, которая не объясняется их общими корреляциями с X1 и X2. Здесь X1 X2 и X3 могут быть отстающими версиями Y.
$PACF(k,p) =$
$\Sigma\ (yk - \hat{y}k|k-p+1) \cdot (yi-k - \hat{y}i-k|i-k-p+1\ N-1\ )$
$i=0$
$std(yk - \hat{y}k|k-p+1) \cdot std(yi-k - \hat{y}i-k|i-k-p+1)$
$=$
$PACVF(k,p)$
$std(yk - \hat{y}k|k-p+1) \cdot std(yi-k - \hat{y}i-k|i-k-p+1)$
;
где
· $\hat{y}k|k-p+1$ - это $\hat{y}k$ предсказанное по $yk-1, \dots, yk-p+1$;
· $\hat{y}i-k|i-k-p+1$ - это $\hat{y}i-k$ предсказанное по $yi-k-1, \dots, yi-k-p+1$;
· $PACVF(k,p)$ - это частичная ковариация,
$PACVF(k,p) = \Sigma(yk - \hat{y}k|k-p+1) \cdot (yi-k - \hat{y}i-k|i-k-p+1)$
$N-1$
$i=0$
$/N$

```
lags = 20
sm.graphics.tsa.plot_pacf(airpass,
lags = lags,
zero = False)
plt.xticks(np.arange(1, lags + 1, 1.0))
plt.show()
```

Для графической проверки распределения в наборе данных также можно построить график Q – Q (квантиль-квантиль) и график гистограммы.

```
#Plot the QQ plot of the data:
sm.qqplot(airpass,
line='s')
plt.title("QQ Plot")
plt.show()
sns.displot(airpass,bins=20,kde=True);
```

Также проверим данные по тесту Люнга-Бокса (гипотеза нормального распределения).

```
lags = 20
#The Ljung-Box test results for the first k lags:
tmp_acor = sm.stats.diagnostic.acorr_ljungbox(airpass, lags = lags,
boxpierce = True, return_df = False)
# get the p-values
p_vals = pd.Series(tmp_acor[1])
#Start the index from 1 instead of 0 (because Ljung-Box test is for
lag values from 1 to k)
p_vals.index += 1
fig = plt.figure()
#Plot the p-values:
p_vals.plot(linestyle='',
marker='o',
title = "p-values for Ljung-Box statistic",
legend = False)
#Add the horizontal 0.05 critical value line
plt.axhline(y = 0.05, color = 'blue', linestyle='--')
plt.show()
```

Мы также можем проверить стационарность данных с помощью расширенного теста

Дики – Фуллера (ADF).

```
statmodels.tsa.stattools.adfuller.
dftest = sm.tsa.stattools.adfuller(airpass.data, autolag='AIC')
print("Test statistic = {:.3f}".format(dftest[0]))
print("P-value = {:.3f}".format(dftest[1]))
print("Critical values :")
for k, v in dftest[4].items():
print("\t{}: {} - The data is {} stationary with {}%
confidence".format(k, v, "not" if v<dftest[0] else "", 100-int(k[:-1])))
Test statistic = 0.815
P-value = 0.992
Critical values :
1%: -3.4816817173418295 - The data is not stationary with 99%
confidence
5%: -2.8840418343195267 - The data is not stationary with 95%
confidence
10%: -2.578770059171598 - The data is not stationary with 90%
confidence
```

Упражнение 1

1. Промоделируйте нормальное распределение проведите для него весь представленный выше анализ.

Разложение тренд-сезон-остаток с помощью StatsModels

Мы также можем попробовать убрать тренд из данных, для этого можно использовать встроенную процедуру statsmodels.tsa.stattools.detrend. На практике мы воспользуемся тремя разными методами и сравним их результаты.

```
airpass[['de_trend_1']] = (airpass[['data']] -
airpass[['data']].rolling(window=12).mean()) /
airpass[['data']].rolling(window=12).std()
airpass[['de_trend_2']] = airpass[['data']].diff(1)
airpass[['de_trend_3']] =sm.tsa.tsatools.detrend(airpass[['data']],
order=1)
plt.figure(figsize = (18,4))
plt.subplot(131)
airpass.de_trend_1.plot();
plt.subplot(132)
airpass.de_trend_2.plot();
plt.subplot(133)
airpass.de_trend_3.plot();
```

Обратите внимание, что для того, чтобы сделать данные стационарными, мы также можем удалить сезонную часть. Самый простой способ - сделать сезонную производную для рядов с исключенным трендом.

```
airpass[['de_season']] = airpass.de_trend_1.diff(12)
airpass.de_season.plot();
airpass.head(3)
data de_trend_1 de_trend_2 de_trend_3 de_season
1949-01-01 112 NaN NaN 21.690038 NaN
1949-02-01 118 NaN 6.0 25.032854 NaN
1949-03-01 132 NaN 14.0 36.375670 NaN
print('airpass.de_trend_1')
dftest = sm.tsa.stattools.adfuller(airpass.de_trend_1.dropna(),
autolag='AIC')
print("Test statistic = {:.3f}".format(dftest[0]))
print("P-value = {:.3f}".format(dftest[1]))
print("Critical values :")
for k, v in dftest[4].items():
print("\t{}: {} - The data is {} stationary with {}%
```

```
confidence".format(k, v, "not" if v<dftest[0] else "", 100-int(k[:-1])))
print('airpass.de_season')
dftest = sm.tsa.stattools.adfuller(airpass.de_season.dropna(),
autolag='AIC')
print("Test statistic = {:.3f}".format(dftest[0]))
print("P-value = {:.3f}".format(dftest[1]))
print("Critical values :")
for k, v in dftest[4].items():
print("\t{}: {} - The data is {} stationary with {}%
confidence".format(k, v, "not" if v<dftest[0] else "", 100-int(k[:-1])))
airpass.de_trend_1
Test statistic = -2.481
P-value = 0.120
Critical values :
1%: -3.4865346059036564 - The data is not stationary with 99%
confidence
5%: -2.8861509858476264 - The data is not stationary with 95%
confidence
10%: -2.579896092790057 - The data is not stationary with 90%
confidence
airpass.de_season
Test statistic = -3.181
P-value = 0.021
Critical values :
1%: -3.4924012594942333 - The data is not stationary with 99%
confidence
5%: -2.8886968193364835 - The data is stationary with 95%
confidence
10%: -2.5812552709190673 - The data is stationary with 90%
confidence
```

Упражнение 2

1. Проверьте изучаемый ряд на стационарность , а также ряды `de_trend_2` и `de_trend_3` с и без сезонной части.
2. Проведите графический анализ `de_season` части.
3. Визуализируйте части изучаемого временного ряда (тренд, сезонность, остаток) с использованием изученного метода (используйте $trend = y(t) - seasonal - residual$ и для сезонности также).

Существует множество методов декомпозиции временных рядов. Мы начнем со statsmodels.tsa.seasonal_decompose, чтобы автоматически разложить ряд. Метод реализует одностороннюю или двухстороннюю декомпозицию тренда простым скользящим средним, а затем пытается найти такую составляющую, что $S\tilde{}\ t = S\tilde{}\ t+period$, где $S\tilde{}$ – это временной ряд без тренда.

```
result = sm.tsa.seasonal_decompose(airpass.data, model='additive', period
= 12, two_sided = True)
result.plot()
plt.show()
```

Мы также можем построить результаты разложения вместе.

```
plt.figure(figsize=(18,6))
plt.plot(airpass.data, label="airpass")
result.trend.plot(label="trend")
result.seasonal.plot(label="seasonal")
result.resid.plot(label="resid")
plt.legend(fontsize ='x-large')
plt.show()
```

Давайте проанализируем остатки нашей декомпозиции

```
residuals =result.resid
```

```
rmse = np.sqrt(np.sum(np.power(residuals,2)))
print('Test RMSE: %.3f' % rmse)
residuals = pd.DataFrame(residuals)
print(residuals.describe())
plt.plot(residuals)
plt.show()
Test RMSE: 221.531
resid
count 132.000000
mean -0.751263
std 19.340535
min -43.967172
25% -11.248422
50% -0.452020
75% 9.527146
max 61.051768
```

Другой метод разложения тренд-сезонной составляющих основан на локальной полиномиальной регрессии, которая также известна как LOESS (сглаживание локально оцененной диаграммы разброса). Этот метод называется разложением по сезонным трендам с использованием LOESS (**Seasonal-Trend decomposition using LOESS, STL**).

```
result_stl =sm.tsa.STL(airpass.data, period = 12,).fit()
fig = result_stl.plot()
```

Упражнение 3

1. Проанализируйте с помощью ранее показанного графического анализа и ADF остатки разложения `statsmodels.tsa.seasonal_decompose`.
2. Проанализируйте с помощью ранее показанного графического анализа и ADF остатки разложения `statsmodels.tsa.STL`.

Предсказание временных рядов

Наивное предсказание

Мы можем сделать простой (наивный) одношаговый прогноз. Давайте превратим наш набор данных в задачу обучения с учителем. Мы можем добиться этого, создав ряд с задержкой. Теперь, в преобразованном наборе данных значения в (t) являются предикторами (X), а значения в (t + 1) являются целевой переменной (Y).

```
airpass[['label']] = airpass.data.shift(1)
airpass.head()
data de_trend_1 de_trend_2 de_trend_3 de_season label
1949-01-01 112 NaN NaN 21.690038 NaN NaN
1949-02-01 118 NaN 6.0 25.032854 NaN 112.0
1949-03-01 132 NaN 14.0 36.375670 NaN 118.0
1949-04-01 129 NaN -3.0 30.718487 NaN 132.0
1949-05-01 121 NaN -8.0 20.061303 NaN 129.0
```

Затем мы можем разделить набор данных на обучающую и тестовую подмножества, как показано ниже: 70% серии - обучающие, а 30% - тестовые данные.

```
x = airpass.data.values
y = airpass.label.values
train_size = int(len(x) * 0.7)
x_train, x_test = x[:train_size], x[train_size:]
y_train, y_test = y[:train_size], y[train_size:]
```

Наивное предсказание эквивалентно $y(t+1) = y(t)$

```
def naive_forecast(x_ts, n_predict):
forecast = [x_ts[-1]]*n_predict
return forecast
def forecast_residual(x_pred, x_ground_truth):
residual = x_ground_truth-x_pred
return residual
```

Мы можем оценить нашу базовую модель на тестовом наборе данных. Мы шаг за шагом

будем просматривать набор тестовых данных (второй столбец) и получать прогнозы.

```
n_preidctions = x_test.size
predicted = naive_forecast(x_train,n_preidctions)
residuals = forecast_residual(x_test,y_test)
rmse = np.sqrt(np.sum(np.power(residuals,2)))
print('Test RMSE: %.3f' % rmse)
residuals = pd.DataFrame(residuals)
print(residuals.describe())
plt.plot(residuals)
plt.show()
Test RMSE: 327.744
0
count 44.000000
mean -1.909091
std 49.943139
min -87.000000
25% -42.250000
50% -11.000000
75% 42.750000
max 101.000000
```

Также будет полезно визуализировать данные.

```
plt.plot(predicted)
plt.plot(y_test)
```

Как мы видим выше, остатки нашего прогноза достаточно отличаются от нормального распределения. Также сводная статистика позволяет предположить смещение в модели.

Помимо простого наивного прогноза, его можно сделать сезонно-наивным,

$y(t) = y(t - ts)$,

где ts s лаг (задержанное значение) временного ряда, где s – это период.

```
def snaive_forecast(x_ts, season_period, n_predict):
forecast = np.zeros(n_predict)
for i in range (min(n_predict,season_period)):
forecast[i] = x_ts[-season_period+i]
if n_predict>season_period:
for i in range (n_predict-season_period):
forecast[i+season_period] = forecast[i]
return forecast
```

Давайте протестируем нашу функцию

```
x =np.arange(36)* np.sin(2*np.pi*np.arange(36)/12)
plt.plot(x)
x_also = snaive_forecast(x, season_period=12, n_predict=36)
plt.plot(x_also)
n_preidctions = x_test.size
season_period = 12
predicted = snaive_forecast(x_train,season_period,n_preidctions)
residuals = forecast_residual(predicted,y_test)
rmse = np.sqrt(np.sum(np.power(residuals,2)))
print('Test RMSE: %.3f' % rmse)
residuals = pd.DataFrame(residuals)
print(residuals.describe())
plt.plot(residuals)
plt.show()
Test RMSE: 692.528
0
count 44.000000
mean 79.340909
std 68.643990
```

```
min -38.000000
25% 28.000000
50% 73.500000
75% 117.500000
max 251.000000
plt.plot(predicted)
plt.plot(y_test);
```
Как мы видим предсказание несколько лучше.

Упражнение 4

1. Постройте графики результатов прогноза и проведите их вместе.
2. Проверьте остатки наивного прогноза с ранее показанным графическим анализом (ACF, PACF, Q-Q, Hist, p-значения Ljung-Box).
3. Реализуйте прогноз на основескользящего среднего, используя
$y(t) = 1$
M
$\Sigma M-1\ y$
$i=0\ (t-i)$ и сделай сравнение с наивным прогнозом.

Exercise 5

1. Сделайте прогноз для одно из полученных выше результатов разложения на сезон и тренд (можно наивный прогноз).
2. Проверьте остатки прогноза при помощи изученных тестов (ACF,PACF, Q-Q, Hist, Ljung-Box p-values, ADF).

Сглаживающие предсказания

Помимо простого скользящего среднего, может выполняться экспоненциальное сглаживание.

```
from statsmodels.tsa.holtwinters import (SimpleExpSmoothing, # SEMA
Holt,# DEMA
ExponentialSmoothing) # TEMA
```
Single Exponential Smoothing (экспоненциальное скользящее среднее, **SEMA**):
$\hat{y}0 = y0;$
$\hat{y}n = \alpha yn + (1-\alpha)\hat{y}n-1,$
где α параметр сглаживания; $\hat{y}$ предсказанное значение.

```
n_predict = x_test.size//2
x_train = pd.DataFrame(x_train)
plt.figure(figsize=(18,8))
plt.plot(airpass.data.values, label='groud')
# Simple Exponential Smoothing
fit1 =
SimpleExpSmoothing(x_train).fit(smoothing_level=0.2,optimized=False)
fcast1 = fit1.forecast(n_predict).rename(r'$\alpha=0.2$')
# plot
fcast1.plot(marker='o', color='blue', legend=True)
fit1.fittedvalues.plot(marker='o', color='blue')
fit2 =
SimpleExpSmoothing(x_train).fit(smoothing_level=0.6,optimized=False)
fcast2 = fit2.forecast(n_predict).rename(r'$\alpha=0.6$')
# plot
fcast2.plot(marker='o', color='red', legend=True)
fit2.fittedvalues.plot(marker='o', color='red')
fit3 =
SimpleExpSmoothing(x_train).fit(smoothing_level=0.8,optimized=False)
fcast3 =
fit3.forecast(n_predict).rename(r'$\alpha=%s$'%fit3.model.params['smoothin
g_level'])
# plot
fcast3.plot(marker='o', color='green', legend=True)
```

```
fit3.fittedvalues.plot(marker='o', color='green')
plt.show()
```
Double Exponential Smoothing (двойное экспоненциальное сглаживание, **DEMA, Holt smoothing, Сглаживание Холта**):
$b0 = y1 - y0; \rightarrow trend$
$l0 = y0; \rightarrow level$
$ln = \alpha\, yn + (1-\alpha)(ln{-}1 + bn{-}1);$
$bn = \beta(ln - ln{-}1) + (1-\beta)bn{-}1;$
$\hat{y}n{+}1 = ln + bn$
где β дополнительный параметр сглаживания.
```
n_predict = x_test.size//2
x_train = pd.DataFrame(x_train)
plt.figure(figsize=(18,8))
plt.plot(airpass.data.values, label='groud')
fit1 = Holt(x_train).fit(smoothing_level=0.8, smoothing_trend=0.2,
optimized=False)
fcast1 = fit1.forecast(n_predict).rename("DEMA with linear trend")
fit2 = Holt(x_train, exponential=True).fit(smoothing_level=0.8,
smoothing_trend=0.2, optimized=False)
fcast2 = fit2.forecast(n_predict).rename("DEMA with Exponential trend")
fit3 = Holt(x_train, damped_trend=True).fit(smoothing_level=0.8,
smoothing_trend=0.2)
fcast3 = fit3.forecast(n_predict).rename("DEMA with Additive damped
trend")
fit1.fittedvalues.plot(marker="o", color='blue')
fcast1.plot(color='blue', marker="o", legend=True)
fit2.fittedvalues.plot(marker="o", color='red')
fcast2.plot(color='red', marker="o", legend=True)
fit3.fittedvalues.plot(marker="o", color='green')
fcast3.plot(color='green', marker="o", legend=True)
plt.show()
```
Triple Exponential Smoothing (or тройное экспоненциальное сглаживание, **TEMA**, **Holt-Winters, Холт-Винтер сглаживание, HW**):
$b0 = y1 - y0; \rightarrow trend$
$l0 = y0; \rightarrow level$
$s0 = \Sigma\,(yL{+}i - yi\,)/L2$
$L{-}1$
$i{=}0$
$; \rightarrow seasonality$
$l\_n = \alpha(y\_n - sn{-}L + (1-\alpha)(ln{-}1 + bn{-}1);$
$bn = \beta\,(ln - ln{-}1) + (1-\beta)bn{-}1;$
$sn = \gamma(yn - ln\,) + (1-\gamma)sn{-}L\,;$
$\hat{y}n{+}m = ln + mbn + sn{-}L{+}1{+}(m{-}1)modL$
где
· γ параметр тройного сглаживания;
· L период сезонности;
· m число точек для предсказания.
Индекс $n - L + 1 + (m-1)modL$ в уравнении прогноза для TEMA - это смещение сезонных компонентов от последнего полного сезона из наблюдаемых данных (т.е. если мы прогнозируем 3-ю точку в сезоне 45 в будущем, мы не можем использовать сезонные компоненты из 44-го сезона поскольку этот сезон также является прогнозируемым - мы можем использовать только точки из наблюдаемых данных).
Для TEMA можно добавить дополнительные уравнения для оценки значений отклонения.
$\hat{y}maxx = ln{-}1 + bn{-}1 + sn{-}L + mdk{-}L\,,$
$\hat{y}minx = ln{-}1 + bn{-}1 + sn{-}L - mdk{-}L\,,$

$dk = \gamma \mid yk - yk\hat{} \mid +(1-\gamma)dk-L$,
где d ожидаемое отклонение.

```
n_predict = x_test.size
x_train = pd.DataFrame(x_train)
fit1 = ExponentialSmoothing(x_train, seasonal_periods=12, trend='add',
seasonal='add').fit(use_boxcox=True)
fit2 = ExponentialSmoothing(x_train, seasonal_periods=12, trend='add',
seasonal='mul').fit(use_boxcox=True)
fit3 = ExponentialSmoothing(x_train, seasonal_periods=12, trend='add',
seasonal='add', damped_trend=True).fit(use_boxcox=True)
fit4 = ExponentialSmoothing(x_train, seasonal_periods=12, trend='add',
seasonal='mul', damped_trend=True).fit(use_boxcox=True)
plt.figure(figsize=(18,8))
plt.plot(airpass.data.values, label='groud')
fit1.fittedvalues.plot(style='--', color='red' )
fit2.fittedvalues.plot(style='--', color='green', label='trend add, season
mul')
fit1.forecast(n_predict).rename("TEMA trend add, season
add").plot(style='--', marker='o', color='red', legend=True)
fit2.forecast(n_predict).rename("TEMA trend add, season
mul").plot(style='--', marker='o', color='green', legend=True)
plt.show()
plt.figure(figsize=(18,8))
plt.plot(airpass.data.values, label='groud')
fit3.fittedvalues.plot(style='--', color='red')
fit4.fittedvalues.plot(style='--', color='green')
fit3.forecast(n_predict).rename("TEMA trend add, season add,
dumped").plot(style='--', marker='o', color='red', legend=True)
fit4.forecast(n_predict).rename("TEMA trend add, season mul,
dumped").plot(style='--', marker='o', color='green', legend=True)
plt.show()
```

Упражнение 6

1. Проанализируйте остатки прогнозов SEMA, DEMA и TEMA, как в предыдущем упражнении._

Лабораторная работа 4

Работа сопровождается методическими материалами.

Знакомство с SCIKIT-TIME (SKTIME)

Знакомство с библиотекой машинного обучения для анализа временных рядов sktime. Представления временных рядов с точки зрения задач машинного обучения. Преобразования временных раядов. Предсказание временных рядов.

Imports

Scikit-Time (sktime) - это набор инструментов Python с открытым исходным кодом для машинного обучения и работы с временными рядами. Это проект, реализуемый сообществом и финансируемый Советом экономических и социальных исследований Великобритании, Центром исследования потребительских данных и Институтом Алана Тьюринга.

Sktime расширяет API scikit-learn для задач временных рядов. Он предоставляет необходимые алгоритмы и инструменты преобразования для эффективного решения задач регрессии, прогнозирования и классификации временных рядов. Библиотека включает специальные алгоритмы обучения для временных рядов и методы преобразования, не представленные во многих других распространенных библиотеках.

Установим sktime и его зависимости

```
!pip install sktime --user
!pip install pmdarima
!pip install tbats
import sktime
import pandas as pd
import numpy as np
from warnings import simplefilter
from sktime.datasets import load_airline
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 sMAPE, smape_loss
from sktime.transformations.series.detrend import Deseasonalizer,
Detrender
from sktime.utils.plotting import plot_series
simplefilter("ignore", FutureWarning)
%matplotlib inline
```

Набор данных

Для использования встроенных наборов данных мы можем импортировать их из соответствующего модуля. Для начала импортируем уже знакомый вам набор данных с пассажирами авиакомпаний.

```
y = sktime.datasets.load_airline()
print('output data type: ',type(y))
y.plot();
print(y.describe())
output data type: <class 'pandas.core.series.Series'>
count 144.000000
mean 280.298611
std 119.966317
min 104.000000
25% 180.000000
50% 265.500000
75% 360.500000
max 622.000000
Name: Number of airline passengers, dtype: float64
```
Мы также можем использовать встроенный метод plot_series для визуализации данных.
```
sktime.utils.plotting.plot_series(y);
```
Для разделения массивов или матриц на последовательные обучающие и тестовые подмножества мы будем использовать метод temporal_train_test_split. Мы будем использовать прогнозирование с заранее определенным горизонтом, используя переменную TEST_SIZE.
```
TEST_SIZE = 36
y_train, y_test =
sktime.forecasting.model_selection.temporal_train_test_split(y,
test_size=TEST_SIZE)
print('check splitted data size: ', y_train.shape[0], y_test.shape[0])
sktime.utils.plotting.plot_series(y_train, y_test, labels=["y_train",
"y_test"]);
check splitted data size: 108 36
```
После разделения мы можем указать горизонт прогнозирования, используя абсолютные значениях отсчетов времени.
```
fh = ForecastingHorizon(y_test.index, is_relative=False)
print(fh)
ForecastingHorizon(['1958-01', '1958-02', '1958-03', '1958-04', '1958-05',
'1958-06', '1960-11', '1960-12'],
dtype='period[M]', name='Period', freq='M', is_relative=False)
```
В качестве альтернативы мы можем использовать относительные отсчеты, испльзуя np.array
```
fh = np.arange(len(y_test)) + 1
fh = np.array([2, 5]) # 2nd and 5th step ahead
```
Задача предсказания

Традиционная постановка

Сделаем наивный прогноз, как на предыдущем занятии (по statsmodels.tsa). SKTime позволяет сделать это в общем стиле, совместимом с другими scikit библиотеками. Для оценки здесь будет использоваться так называемая мера Symmetry MAPE:

$$sMAPE = \frac{1}{H} \Sigma \frac{|y(hi) - \hat{y}(hi)|}{|y(hi)| + |\hat{y}(hi)|}$$
$$i=1$$

Для наивного прогноза:
```
fh = ForecastingHorizon(y_test.index, is_relative=False)
forecaster = NaiveForecaster(strategy="last")
```

```
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('score = ',smape_loss(y_pred, y_test))
score = 0.2319577038795143
```
Для сезонного-наивного прогноза:
```
forecaster = NaiveForecaster(strategy="last", sp=12)
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('score = ',smape_loss(y_pred, y_test))
score = 0.145427686270316
```
Для Holt-Winter сглаживающего прогнзоа:
```
forecaster = ExponentialSmoothing(trend="add", seasonal="additive", sp=12)
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('score = ',smape_loss(y_pred, y_test))
score = 0.05027652903776341
```
Примечание для получения простого экспоненциального сглаживания (SES) и двойного экспоненциального сглаживания (HOLT, DEA) используйте
```
ses = ExponentialSmoothing(sp=12)
holt = ExponentialSmoothing(trend="add", damped_trend=False, sp=12)
```
Для построения ансамбля методов используйте
```
ses = ExponentialSmoothing(sp=12)
holt = ExponentialSmoothing(trend="add", damped_trend=False, sp=12)
damped_holt = ExponentialSmoothing(trend="add", damped_trend=True, sp=12)
holt_winter = ExponentialSmoothing(trend="add", seasonal="additive",
sp=12)
forecaster = EnsembleForecaster(
[
("ses", ses),
("holt", holt),
("damped", damped_holt),
("holt-winter",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('score = ',smape_loss(y_pred, y_test))
score = 0.1393047282084183
```
Для случая автоматизированного подбора гиперпараметров экспоненциального сглаживания (Холта-Винтера):
```
from sktime.forecasting.ets import AutoETS
forecaster = AutoETS(auto=True, sp=12, 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('score = ',smape_loss(y_pred, y_test))
score = 0.06318722677034444
```

Упражнение 1

1. Сравнить точность предсказания различных вариантов модели ETS, таких как:

```
· trend : str{"add", "mul", "additive", "multiplicative", None}, optional
(default=None)
Type of trend component.
· damped_trend : bool, optional (default=None)
Should the trend component be damped.
· seasonal : {"add", "mul", "additive", "multiplicative", None}, optional
(default=None)
Type of seasonal component.
```

Примечание. В обобщенном виде методы сглаживания могут быть объединены в так называемую модель Error-Trend_Seasonality (ETS). Модель может быть описана как Ошибка, Тренд, Сезонность (ETS): s = ETS (X, X, X) s, где X может быть N-None, A-аддитивным, M-мультипликативным, Ad-аддитивным затухающим, s-период сезонности, если S не равно None.

Известные вам случае могут соответствовать следующим вариантам модели ETS: Простое экспоненциальное сглаживание соответствует ETS (A, N, N). Тройное экспоненциальное сглаживание соответствует ETS (A, A, A).

2. Поробуйте провести прогноз моделью ETS с учетом части `error`.

```
error : str, optional
The error model. "add" (default) or "mul".
```

3. Сравните результаты прогноза ETS с использованием (или без) трансформации временного ряда методом boxcox.

```
use_boxcox : {True, False, 'log', float}, optional
Should the Box-Cox transform be applied to the data first? If 'log'
then apply the log. If float then use lambda equal to float.
```

Использование предсказаний с использованием библиотеки `scikit-learn`

Помимо встроенных методов, SKTime позволяет применять подходы, основанные на scikit-learn. Например, задачу прогнозирования можно сравнить с задачей регрессии в sklearn. Однако, прямое использование стандартной регрессии sklearn, требуют наличия данных и меток, которые не нужны во временных рядах. Как будет показано ниже эта проблема может быть сравнительно просто решена.

Лучший способ составить прогноз с использованием обычной регрессии - это использовать так называемую технику “сокращения прогнозирования”, которая представляет собой преобразование неявно имеющейся задачи долгосрочной регрессии в задачу на основе скользящего окна. Такое преобразование можно сделать с помощью метода make_reduction, как показано ниже.

Идея редукции состоит в том, чтобы свести задачу регрессии в отношении временных рядов к проблеме табличной регрессии, как показано ниже. Обратите внимание, что в задаче табличной регрессии или регрессии скользящего окна на этапе обучения вы можете использовать предсказанные значения вместо известных в соответствующих позициях. Это показано серыми стрелками.

Посмотрим, как работает метод make_reduction. Существуют «прямые», «рекурсивные» и «многовыводные» стратегии прогнозирования. В прямой стратегии мы используем разные прогнозы для каждого результата (цели) (без серых стрелок). В рекурсивной стратегии мы используем предыдущие результаты в прогнозах для каждого следующего результата (с серыми стрелками). В стратегии с несколькими выходами мы напрямую прогнозируем несколько шагов.

```
from sklearn.neighbors import KNeighborsRegressor
REGRESSION_WINDOW = 5
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('score = ',smape_loss(y_pred, y_test))
score = 0.2329490896108744
```
В make_reduction аргументы window_length и стратегии являются гиперпараметрами, которые мы, можем оптимизировать. В приведенном ниже примере мы используем поиск по сетке для проверки наилучшей длины окна. Для этого мы используется следующие шаги:
· создаем сетку длин окна
· создание начального предсказателя с помощью регрессора KNeighborsRegressor.
· Разделение выборки на тренировочную и для проверки (мы сделали это, сдвинув начало окна до 80% размера выборки и продвинувшись с длиной 25 точек до конца),
· производим поиск по сетке прогнозов с оценкой на каждой итерации и функцией SMAPE в качестве меры.
```
grid = {"window_length": [5, 7, 10, 12, 15,17,20]}
#initial forecaster
regressor = KNeighborsRegressor(n_neighbors=1)
forecaster = make_reduction(regressor,
window_length=grid["window_length"][-1],
strategy="recursive")
# use temporal cross-validation to find the optimal parameter.
cros_val = SlidingWindowSplitter( initial_window=int(len(y_train) * 0.7),
window_length=25)
#grid search
gscv = ForecastingGridSearchCV(forecaster, strategy="refit", cv=cros_val,
param_grid=grid, scoring=sMAPE())
gscv.fit(y_train)
print('best window size = ',gscv.best_params_)
best window size = {'window_length': 12}
```
to see the full protocol use the following code
```
pd.DataFrame(gscv.cv_results_)
mean_test_sMAPE mean_fit_time mean_pred_time params
\
0 0.133554 0.001182 0.002694 {'window_length': 5}
1 0.117308 0.001151 0.002576 {'window_length': 7}
2 0.095433 0.001213 0.002512 {'window_length': 10}
3 0.090602 0.001333 0.002454 {'window_length': 12}
4 0.095508 0.001121 0.002542 {'window_length': 15}
5 0.095508 0.001181 0.002361 {'window_length': 17}
6 0.095508 0.001150 0.002240 {'window_length': 20}
rank_test_sMAPE
0 7.0
1 6.0
2 2.0
3 1.0
4 4.0
5 4.0
6 4.0
```
после поиска вы можете сделать прогноз с наилучшими результатами
```
y_pred = gscv.predict(fh)
plot_series(y_train, y_test, y_pred, labels=["y_train", "y_test",
"y_pred"])
print('score = ',smape_loss(y_pred, y_test))
score = 0.14008272913734346
```
Также возможен поиск по сетке другими методами.
```
ses = ExponentialSmoothing(sp=12)
```

```
holt = ExponentialSmoothing(trend="add", damped_trend=False, sp=12)
holt_winter = ExponentialSmoothing(trend="add", seasonal="additive",
sp=12)
forecaster = MultiplexForecaster(
forecasters=[
("ses", ses),
("holt", holt),
("holt_winter", holt_winter),
]
)
cv = SlidingWindowSplitter(initial_window=int(len(y_train) * 0.5),
window_length=30)
forecaster_grid = {"selected_forecaster": ["ses", "holt", "holt_winter"]}
gscv = ForecastingGridSearchCV(forecaster, cv=cv,
param_grid=forecaster_grid)
gscv.fit(y_train)
print(gscv.best_params_, "\n\n\n", gscv.best_forecaster_)
{'selected_forecaster': 'holt_winter'}
MultiplexForecaster(forecasters=[('ses', ExponentialSmoothing(sp=12)),
('holt',
ExponentialSmoothing(sp=12,
trend='add')),
('holt_winter',
ExponentialSmoothing(seasonal='additive',
sp=12,
trend='add'))],
selected_forecaster='holt_winter')
y_pred = gscv.predict(fh)
plot_series(y_train, y_test, y_pred, labels=["y_train", "y_test",
"y_pred"])
print('score = ',smape_loss(y_pred, y_test))
0.05027652903776341
```
Помимо простого деления train-test-split мы можем сделать ExpandingWindowSplitter.
```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor()
forecaster = make_reduction(regressor,
window_length=12,
strategy="recursive")
cv = ExpandingWindowSplitter(
step_length=12, fh=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12],
initial_window=72)
df = evaluate(forecaster=forecaster, y=y, cv=cv, strategy="refit",
return_data=True)
df.iloc[:, :5]
cutoff fit_time len_train_window pred_time test_sMAPE
0 1954-12 0.125927 72 0.071958 0.132538
1 1955-12 0.123929 84 0.074957 0.055439
2 1956-12 0.126938 96 0.078944 0.078507
3 1957-12 0.142903 108 0.084950 0.127128
4 1958-12 0.146924 120 0.083951 0.070665
5 1959-12 0.140903 132 0.072958 0.053559
```
теперь вы можете видеть результат этого ExpandingWindowSplitter.
```
# visualization of a forecaster evaluation
fig, ax = plot_series(
y,
df["y_pred"].iloc[0],
```

```
df["y_pred"].iloc[1],
df["y_pred"].iloc[2],
df["y_pred"].iloc[3],
df["y_pred"].iloc[4],
df["y_pred"].iloc[5],
markers=["o", "", "", "", "", "", ""],
labels=["y_true"] + ["y_pred (Backtest " + str(x) + ")" for x in
range(6)],
)
ax.legend();
```

Упражнение 2

1. Сделайте EnsembleForecaster как минимум двумя методами из sktime и sklearn.
2. Сделайте ForecastingGridSearchCV для поиска оптимального количества ближайших соседей для метода KNN.
3. Сделайте MultiplexForecaster для Naive, Holt-Winter, Random Forest Regressor, AdaBoostRegressor.

Обратите внимание, что sktime содержит множество заимствований из других пакетов. В некоторых случаях их необходимо будет установить дополнительно.

```
from sktime.forecasting.arima import ARIMA, AutoARIMA
forecaster = AutoARIMA(sp=12, suppress_warnings=True)
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('score = ',smape_loss(y_pred, y_test))
0.04117062367046532
from sktime.forecasting.tbats import TBATS
forecaster = TBATS(sp=12, use_trend=True, use_box_cox=False)
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('score = ',smape_loss(y_pred, y_test))
0.08493353477049963
```

Преобразование данных

Sktimes содержит ряд инструментов для декомпозиции временных рядов и прогнозирования компонентов. Один из универсальных инструментов - преобразование детрендинг. Мы можем использовать его с методом прогнозирования тренда PolynomialTrendForecaster. В примере ниже мы будем использовать линейный тренд (степень полинома равна 1).

```
# liner detrending
forecaster = PolynomialTrendForecaster(degree=1)
transformer = Detrender(forecaster=forecaster)
y_detrend = transformer.fit_transform(y_train)
fh_ins = -np.arange(len(y_train)) # in-sample forecasting horizon
y_trend = forecaster.fit(y_train).predict(fh=fh_ins)
reconstructed = y_trend + y_detrend
plot_series(y_train, y_trend, y_detrend, reconstructed,
labels=["y_train", "fitted linear trend", "de-trended data",
"reconstruction"]);
```

Мы можем сделать конвейер трансформаций для прогноза, как это показано ниже.

```
regressor = KNeighborsRegressor(n_neighbors=3)
forecaster = TransformedTargetForecaster(
[
("deseasonalize", Deseasonalizer(model="multiplicative", sp=12)),
("detrend",
```

```
Detrender(forecaster=PolynomialTrendForecaster(degree=1))),
( "forecast",
make_reduction(regressor,window_length=15,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('score = ',smape_loss(y_pred, y_test))
score = 0.03983950097748029
```

Упражнение 3

1. Создайте конвейер с Deseasonalizer и Naive Forecast для остаточной части, сравните и объясните разницу со случаем без преобразования.
2. Сделайте тренд-сезонность-остаток-декомпозицию, проанализируйте результаты.__

Лабораторная работа 5

Работа сопровождается методическими материалами.

Использование моделей АРСС

Использование моделей АРСС для предсказания и анализа временных рядов. Библиотеки sktime, statsmodels, pmdarima. Выбор параметров для модели ARIMA. Тесты на стационарность. Автоматические методы подбора параметров. Анализ остатков. Особенности выбора параметров для модели SARIMA. Использование экзогенных факторов – модель SARIMAX.

Импорт библиотек и данных.

Один из методов, доступных в Python для моделирования и прогнозирования временных рядов, известен как SARIMAX, что означает сезонное авторегрессионное интегрированное скользящие средние с экзогенными регрессорами.

Классический подход к адаптации модели ARIMA - следовать методологии Бокса-Дженкинса.

· Идентификация модели: используйте графический метод и метод сводной статистики для определения тренда и сезонности, чтобы получить представление о порядке производной (d) и порядках модели (p – порядок авторегрессии и q – порядок скользящего среднего).

· Оценка модели: оценка коэффициентов регрессионной модели.

· Диагностика модели максимального правдоподобия: используйте графический метод и статистические тесты остаточных ошибок, чтобы определить особенности данных, не охваченной моделью.

```
!pip install -U statsmodels
!pip install -U pmdarima
import warnings
import itertools
import pandas as pd
import numpy as np
import statsmodels.api as sm
from statsmodels.tsa.stattools import adfuller
import matplotlib.pyplot as plt
#
from sktime.forecasting.model_selection import temporal_train_test_split
from sktime.utils.plotting import plot_series
from sklearn.metrics import mean_squared_error
import pmdarima as pm
```

Мы будем работать с набором данных под названием «Атмосферные выбросы CO2 из непрерывных проб воздуха в обсерватории Мауна-Лоа, Гавайи, США», который собирал замеры выбросов CO2 с марта 1958 года по декабрь 2001 года..

```
dataset = sm.datasets.co2.load_pandas()
y_ = dataset.data
y_.head()
co2
1958-03-29 316.1
1958-04-05 317.3
1958-04-12 317.6
1958-04-19 317.5
1958-04-26 316.4
```

Для начала произведем небольшую предварительную обработку данных.

```
# The 'MS' string groups the by start of the month
y = y_['co2'].resample('MS').mean()
# The term bfill means that we use the value before filling in missing val
ues
y = y.fillna(y.bfill())
y.head()
1958-03-01 316.100000
```

```
1958-04-01 317.200000
1958-05-01 317.433333
1958-06-01 315.625000
1958-07-01 315.625000
Freq: MS, Name: co2, dtype: float64
y.plot();
```

Когда мы наносим данные на график, появляются некоторые различимые закономерности. Временной ряды имеют очевидную сезонность, а также общую тенденцию к росту.

```
y.describe()
count 526.000000
mean 339.624826
std 17.110954
min 313.400000
25% 324.025000
50% 337.912500
75% 354.537500
max 373.800000
Name: co2, dtype: float64
```

Исследование модели ARIMA

Один из наиболее распространенных методов, используемых при прогнозировании временных рядов, известен как модель ARIMA, что означает AutoregRessive Integrated Moving Average. Существует три различных параметра (порядка) с целыми значениями (p, d, q), которые используются для параметризации моделей ARIMA. По этой причине модели ARIMA обозначаются обозначением ARIMA (p, d, q):

· p - авторегрессивная часть модели. Этот параметр позволяет учесть влияние прошлых значений на текущее для модели. Прошлые значения здесь называются запаздывающими наблюдениями (также известными как «запаздывание» или «лаг»). Интуитивно это похоже на утверждение, что завтра, вероятно, будет тепло, если в последние 3 дня было тепло. Другими словами, здесь мы можем сказать, что наше текущее значение температуры зависит от последних трех значений.

· d – интегрирование модели. Этот параметр включает в себя степень различия лагов (то есть количество прошлых временных точек, которые нужно вычесть из текущего значения), чтобы сделать временной ряд стационарным (чтобы исключить часть тренда). Интуитивно это было бы похоже на утверждение о том, что, вероятно, будет одно и то же повышение температуры каждый день (или одно и то же ускорение для второй производной и т.д.).

· q - скользящая средняя часть модели. Этот параметр позволяет представить остаточную часть (шум, ошибку) модели как линейную комбинацию остаточных значений, наблюдаемых в предыдущие моменты времени.

Ручной выбор параметров модели ARIMA

Для начала рассмотрим порядок дифференцирования для тостижения стационарности. Как правило, это 1-3 порядок, реже - больше.

Для проверки стационарности здесь мы будем использовать два метода:

· Скользящая статистика: построение скользящего среднего и скользящего стандартного отклонения. Идея этого метода в том, что временные ряды являются стационарными, если они остаются неизменными во времени.

· Расширенный тест Дики-Фуллера: временной ряд считается стационарным, если значение p низкое (в соответствии с нулевой гипотезой), а критические значения с доверительными интервалами 1%, 5%, 10% максимально близки к статистике ADF.

Скользящая статистика визуально показывает нестационарность среднего значения. А также мы видим уменьшение дисперсии.

```
rolling_mean = y.rolling(window = 12).mean()
rolling_std = y.rolling(window = 12).std()
plt.figure(figsize=(12,4))
```

```
plt.plot(y-y[0], color = 'blue', label = 'Original')
plt.plot(rolling_mean-y[0], color = 'red', label = 'Rolling Mean')
plt.plot(rolling_std, color = 'black', label = 'Rolling Std')
plt.legend(loc = 'best')
plt.title('Rolling Mean & Rolling Standard Deviation')
plt.show()
```

Тест ADF также показывает, что статистика ADF далека от критических значений, а значение p превышает пороговое значение (0,05). Таким образом, можно сделать вывод, что временной ряд не является стационарным.

```
result = adfuller(y)
print('ADF Statistic: {}'.format(result[0]))
print('p-value: {}'.format(result[1]))
print('Critical Values:')
for key, value in result[4].items():
print('\t{}: {}'.format(key, value))
ADF Statistic: 2.359809953995333
p-value: 0.9989901230798025
Critical Values:
1%: -3.4432119442564324
5%: -2.8672126791646955
10%: -2.569791324979607
```

Теперь давайте посмотрим на 1ю производную

```
y_diff = y.diff(1)
# for fill obtained first NaN Value with next
y_diff = y_diff.dropna()
rolling_mean = y_diff.rolling(window = 12).mean()
rolling_std = y_diff.rolling(window = 12).std()
plt.figure(figsize=(12,4))
plt.plot(y_diff, color = 'blue', label = 'Original')
plt.plot(rolling_mean, color = 'red', label = 'Rolling Mean')
plt.plot(rolling_std, color = 'black', label = 'Rolling Std')
plt.legend(loc = 'best')
plt.title('Rolling Mean & Rolling Standard Deviation')
plt.show()
```

Здесь и ниже мы видим, что наши данные теперь удовлетворяют стационарным критериям.

```
y_diff.head()
1958-04-01 1.100000
1958-05-01 0.233333
1958-06-01 -1.808333
1958-07-01 0.000000
1958-08-01 -0.675000
Freq: MS, Name: co2, dtype: float64
result = adfuller(y_diff)
print('ADF Statistic: {}'.format(result[0]))
print('p-value: {}'.format(result[1]))
print('Critical Values:')
for key, value in result[4].items():
print('\t{}: {}'.format(key, value))
ADF Statistic: -5.063202630318491
p-value: 1.6614851317686715e-05
Critical Values:
1%: -3.4432119442564324
5%: -2.8672126791646955
10%: -2.569791324979607
y_diff.head()
```

```
1958-04-01 1.100000
1958-05-01 0.233333
1958-06-01 -1.808333
1958-07-01 0.000000
1958-08-01 -0.675000
Freq: MS, Name: co2, dtype: float64
```
Примечание. Помимо ADF существует множество тестов, среди которых также полезно проверить

· Анализ ACF (АКФ) , в котором для нестационарного процесса вы увидите медленное уменьшение значений АКФ, и резкий спад значений автокорреляции для стационарного случая.

· Тест Квятковского – Филлипса – Шмидта – Шина (KPSS), который дает значения, отличающиеся от ADF в случае детерминированного тренда с точками перегиба.

```
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
# Original Series
plot_acf(y[:], title='Autocorrelation of Original Series');plt.show()
# Usual Differencing
plot_acf(y_diff[:], title='Autocorrelation of Differenced Series');plt.sho
w()
plt.show();
from statsmodels.tsa.stattools import kpss
def kpss_test(series, **kw):
statistic, p_value, n_lags, critical_values = kpss(series, **kw)
# Format Output
print(f'KPSS Statistic: {statistic}')
print(f'p-value: {p_value}')
print(f'num lags: {n_lags}')
print('Critial Values:')
for key, value in critical_values.items():
print(f' {key} : {value}')
print(f'Result: The series is {"not " if p_value < 0.05 else ""}statio
nary')
kpss_test(y_diff)
KPSS Statistic: 0.07042168811681968
p-value: 0.1
num lags: 19
Critial Values:
10% : 0.347
5% : 0.463
2.5% : 0.574
1% : 0.739
Result: The series is stationary
```
Примечание. Если ваш ряд немного недодифференцирован, то необходимо будет добавить один или нескольких дополнительных слогаемых в авторегрессионную часть (повысить порядок) обычно это компенсирует. Аналогичным образом, если разница немного выше, попробуйте добавить дополнительный член к скользящиму средниму.

После выбора порядка интеграции необходимо выбрать порядки AR и MA частей. Для этого могут быть даны следующие рекомендации по этому поводу

· Чтобы оценить порядок авторегрессии (порядок AR), проанализируйте график частичной автокорреляции (PACF). Как правило, график состоит из доверительных интервалов, которые отображаются в виде конуса. По умолчанию установлен доверительный интервал 95%, что предполагает, что значения корреляции за пределами этого интервала, скорее всего, являются корреляцией, а не статистической случайностью.

· После оценки AR мы можем сделать первоначальное предположение о порядке

скользящего среднего (порядок MA). Для этого нужно будем использовать график автокорреляции (ACF). Число ненулевых членов ACF сообщает, сколько членов MA необходимо для устранения любой автокорреляции в стационарном ряду.

```
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
plot_pacf(y_diff);
plt.show()
C:\Users\Администратор\AppData\Roaming\Python\Python37\site-packages
\statsmodels\regression\linear_model.py:1434: RuntimeWarning: invali
d value encountered in sqrt
return rho, np.sqrt(sigmasq)
```

График PACF показывает, что у нас есть как минимум модель AR 1-го порядка с некоторыми дополнительными эффектами, такими как сезонность или не стационарность.

Примечание. График начинается с лага- 0, поэтому мы не можем его учитывать.

```
plot_acf(y_diff);
plt.show()
```

График АКФ показывает зависимость как минимум 2-го порядка, а также наличие некоторой сезонности.

Тестирование выбранной модели

Давайте протестируем выбранную модель ARMA(p=1,d=1,q=2).

```
from statsmodels.tsa.arima.model import ARIMA
# 1,1,2 ARIMA Model
model = ARIMA(y.values, order=(1,1,2))
model_fit = model.fit()
print(model_fit.summary())
SARIMAX Results
==========================================================================
=
Dep. Variable: y No. Observations: 526
Model: ARIMA(1, 1, 2) Log Likelihood -607.411
Date: Mon, 03 May 2021 AIC 1222.822
Time: 11:42:34 BIC 1239.876
Sample: 0 HQIC 1229.500
- 526
Covariance Type: opg
==========================================================================
=
coef std err z P>|z| [0.025 0.975]
--------------------------------------------------------------------------
-
ar.L1 0.4018 0.073 5.529 0.000 0.259 0.544
ma.L1 0.5221 0.072 7.241 0.000 0.381 0.663
ma.L2 0.3636 0.057 6.379 0.000 0.252 0.475
sigma2 0.5909 0.041 14.295 0.000 0.510 0.672
==========================================================================
===
Ljung-Box (L1) (Q): 0.74 Jarque-Bera (JB): 1
.15
Prob(Q): 0.39 Prob(JB): 0
.56
Heteroskedasticity (H): 0.99 Skew: 0
.09
Prob(H) (two-sided): 0.96 Kurtosis: 2
.85
```

```
==========================================================================
===
```
Выведенное описание модели раскрывает много информации. В первой таблице представлена общая информация, включая критерии качества (AIC, BIC и HQIC). Таблица посередине - это таблица коэффициентов, где значения под «coef» - это веса соответствующих слогаемых. Значение sigma2 – это RSS ошибка модели. В последней таблице представлены результаты различных статистических тестов для полученных остатков.
Помимо табличного представления, мы можем проводить диагностику остатков графическим способом.

```
model_fit.plot_diagnostics(figsize=(12,8));
```

На графиках выше мы видим: остаточные ошибки колеблются около нулевого среднего и имеют равномерную дисперсию (верхний левый график). Остаток имеет почти нормальное распределение (верхний правый график). График Q-Q также показывает почти нормальное распределение (внизу слева, в идеале все точки должны точно совпадать с красной линией). Однако на графике ACF (коррелограмме) мы можем заметить некоторые выбросы, превышающие уровень доверительного интервала (внизу справа, любая автокорреляция будет означать, что существует некоторая закономерность в остаточных ошибках, которые не объясняются в модели).
Проведенный анализ показывает, что мы можем улучшить нашу модель. В качестве первого предположения мы можем попытаться увеличить AR-порядок модели.

```
from statsmodels.tsa.arima.model import ARIMA
# 1,1,2 ARIMA Model
model = ARIMA(y.values, order=(2,1,2))
model_fit = model.fit()
print(model_fit.summary())
model_fit.plot_diagnostics(figsize=(12,8));
SARIMAX Results
==========================================================================
=
Dep. Variable: y No. Observations: 526
Model: ARIMA(2, 1, 2) Log Likelihood -533.373
Date: Mon, 03 May 2021 AIC 1076.745
Time: 11:42:37 BIC 1098.062
Sample: 0 HQIC 1085.092
- 526
Covariance Type: opg
==========================================================================
=
coef std err z P>|z| [0.025 0.975]
--------------------------------------------------------------------------
-
ar.L1 1.5539 0.034 45.970 0.000 1.488 1.620
ar.L2 -0.8466 0.038 -22.003 0.000 -0.922 -0.771
ma.L1 -0.8716 0.059 -14.654 0.000 -0.988 -0.755
ma.L2 0.0571 0.068 0.836 0.403 -0.077 0.191
sigma2 0.4447 0.027 16.593 0.000 0.392 0.497
==========================================================================
===
Ljung-Box (L1) (Q): 2.10 Jarque-Bera (JB): 14
.18
Prob(Q): 0.15 Prob(JB): 0
.00
Heteroskedasticity (H): 0.97 Skew: 0
.40
Prob(H) (two-sided): 0.82 Kurtosis: 3
```

```
.06
==============================================================================
===
```
Как мы видим здесь, мы уменьшаем значения как критериев AIC (и BIC), так и ошибку RSS (sigma2) - это означает, что мы движемся в правильном направлении. Однако мы немного ухудшили поведение остатков . Поиск лучших параметров - сложная задача. Здесь мы также можем заметить, что у нас есть небольшое значение компоненты ma.L2, и можем попробовать его устранить.

```
from statsmodels.tsa.arima.model import ARIMA
# 1,1,2 ARIMA Model
model = ARIMA(y.values, order=(2,1,1))
model_fit = model.fit()
print(model_fit.summary())
model_fit.plot_diagnostics(figsize=(12,8));
SARIMAX Results
==============================================================================
=
Dep. Variable: y No. Observations: 526
Model: ARIMA(2, 1, 1) Log Likelihood -533.989
Date: Mon, 03 May 2021 AIC 1075.978
Time: 11:42:40 BIC 1093.032
Sample: 0 HQIC 1082.656
- 526
Covariance Type: opg
==============================================================================
=
coef std err z P>|z| [0.025 0.975]
------------------------------------------------------------------------------
-
ar.L1 1.5353 0.028 55.566 0.000 1.481 1.589
ar.L2 -0.8285 0.030 -27.249 0.000 -0.888 -0.769
ma.L1 -0.8117 0.037 -21.901 0.000 -0.884 -0.739
sigma2 0.4458 0.026 17.040 0.000 0.395 0.497
==============================================================================
===
Ljung-Box (L1) (Q): 5.44 Jarque-Bera (JB): 13
.62
Prob(Q): 0.02 Prob(JB): 0
.00
Heteroskedasticity (H): 0.94 Skew: 0
.39
Prob(H) (two-sided): 0.68 Kurtosis: 3
.10
==============================================================================
===
```
Теперь мы видим, что действительно второй член не влияет на точность предсказания данных.

Теперь мы можем построить график для подобранной модели. В следующем примере, когда вы устанавливаете dynamic = False, для прогнозирования используются запаздывающие значения в выборке. То есть модель обучается до предыдущего значения, чтобы сделать следующий прогноз. Это может привести к тому, что подогнанный прогноз и фактические данные будут выглядеть искусственно хорошими.

```
# Actual vs Fitted
y_hat = model_fit.predict(dynamic=False)
plt.figure(figsize=(12,6))
plt.plot(y_hat[1:], label='predicted')
```

```
plt.plot(y[1:].values, label='original')
plt.legend()
plt.show()
```
Помимо построения модели по существующим данным, мы можем проверить модель тестовых данных. Для этого мы можем разделить наши данные на две выборки - тестовая и тренировочная.
```
# Create Training and Test
train = y[:int(y.size*0.9)]
test = y[int(y.size*0.9):]
# Build Model
model = ARIMA(train, order=(2, 1, 1))
fitted = model.fit()
# Forecast
forecast_res = fitted.get_forecast(test.size, alpha=0.05, dynamic=False)
# 95% conf
# forecast = fitted.forecast(test.size, alpha=0.05) # 95% conf
forecast = forecast_res.predicted_mean
# Make as pandas series
fc_series = pd.Series(forecast.values, index=test.index)
lower_series = pd.Series(forecast_res.conf_int()['lower co2'], index=test.
index)
upper_series = pd.Series(forecast_res.conf_int()['upper co2'], index=test.
index)
# Plot
plt.figure(figsize=(12,4), dpi=100)
plt.plot(train, label='training')
plt.plot(test, label='actual')
plt.plot(fc_series, label='forecast')
plt.fill_between(lower_series.index,
lower_series,
upper_series,
color='k',
alpha=0.15)
plt.title('Forecast vs Actuals')
plt.legend(loc='upper left', fontsize=12)
plt.show()
```
Теперь мы видим, что наша модель была переобучена. Для оценки точности нашего прогноза мы можем ввести следующие меры:
```
# Accuracy metrics
def forecast_accuracy(forecast, actual):
mape = np.mean(np.abs(forecast - actual)/np.abs(actual)) # MAPE
me = np.mean(forecast - actual) # ME
mae = np.mean(np.abs(forecast - actual)) # MAE
mpe = np.mean((forecast - actual)/actual) # MPE
rmse = np.mean((forecast - actual)**2)**.5 # RMSE
corr = np.corrcoef(forecast, actual)[0,1] # corr
mins = np.amin(np.hstack([forecast[:,None],
actual[:,None]]), axis=1)
maxs = np.amax(np.hstack([forecast[:,None],
actual[:,None]]), axis=1)
minmax = 1 - np.mean(mins/maxs) # minmax
return({'mean absoute percentage error':mape,
'mean absoute error ': mae,
'mean percentage error ': mpe,
'root mean square ':rmse,
'correlation coefficient ':corr,
```

```
'minmax error ':minmax})
forecast_accuracy(fc_series.values, test.values)
{'mean absoute percentage error': 0.0037188201927048957,
'mean absoute error ': 1.370467806185665,
'mean percentage error ': -0.0036203570222564747,
'root mean square ': 1.448243682381261,
'correlation coefficient ': 0.984801545546153,
'minmax error ': 0.003718716705316538}
```

Мы можем выбрать лучшие порядки модели, как показано ниже.

```
# Create Training and Test
train = y[:int(y.size*0.9)]
test = y[int(y.size*0.9):]
# Build Model
model = ARIMA(train, order=(3, 2, 2))
fitted = model.fit()
# Forecast
forecast_res = fitted.get_forecast(test.size, alpha=0.05, dynamic=False)
# 95% conf
forecast = forecast_res.predicted_mean
# forecast = fitted.forecast(test.size, alpha=0.05) # Alternative method
# Make as pandas series
fc_series = pd.Series(forecast.values, index=test.index)
lower_series = pd.Series(forecast_res.conf_int()['lower co2'], index=test.
index)
upper_series = pd.Series(forecast_res.conf_int()['upper co2'], index=test.
index)
# Plot
plt.figure(figsize=(12,4), dpi=100)
plt.plot(train, label='training')
plt.plot(test, label='actual')
plt.plot(fc_series, label='forecast')
plt.fill_between(lower_series.index,
lower_series,
upper_series,
color='k',
alpha=0.15)
plt.title('Forecast vs Actuals')
plt.legend(loc='upper left', fontsize=12)
plt.show()
forecast_accuracy(fc_series.values, test.values)
{'mean absoute percentage error': 0.0050279116520248295,
'mean absoute error ': 1.8565906438131856,
'mean percentage error ': -0.0030232064973775142,
'root mean square ': 2.2157261786757645,
'correlation coefficient ': 0.7836715399134366,
'minmax error ': 0.005023574563809641}
```

Сейчас мы видим, что наши метрики стали значительно лучше.

Автоматические методы выбора порядка с помощью библиотеки pmdarima

Помимо ручного выбора параметров ARIMA, мы можем использовать автоматический поиск arima с использованием библиотеки pmdarima.

```
import pmdarima as pm
model = pm.auto_arima(y,
start_p=1,
start_q=1,
test='adf', # use adftest to find optimal 'd'
max_p=10,
```

```
max_q=10, # maximum p and q
m=1, # frequency of series
d=None, # let model determine 'd'
seasonal=False, # No Seasonality
start_P=0,
D=0,
trace=True,
error_action='ignore',
suppress_warnings=True,
stepwise=True)
model.summary()
Performing stepwise search to minimize aic
ARIMA(1,1,1)(0,0,0)[0] intercept : AIC=1268.116, Time=0.12 sec
ARIMA(0,1,0)(0,0,0)[0] intercept : AIC=1676.811, Time=0.01 sec
ARIMA(1,1,0)(0,0,0)[0] intercept : AIC=1328.472, Time=0.06 sec
ARIMA(0,1,1)(0,0,0)[0] intercept : AIC=1374.168, Time=0.07 sec
ARIMA(0,1,0)(0,0,0)[0] : AIC=1678.850, Time=0.01 sec
ARIMA(2,1,1)(0,0,0)[0] intercept : AIC=1010.848, Time=0.26 sec
ARIMA(2,1,0)(0,0,0)[0] intercept : AIC=1205.382, Time=0.06 sec
ARIMA(3,1,1)(0,0,0)[0] intercept : AIC=1009.758, Time=0.53 sec
ARIMA(3,1,0)(0,0,0)[0] intercept : AIC=1150.521, Time=0.17 sec
ARIMA(4,1,1)(0,0,0)[0] intercept : AIC=1008.632, Time=0.91 sec
ARIMA(4,1,0)(0,0,0)[0] intercept : AIC=1149.017, Time=0.15 sec
ARIMA(5,1,1)(0,0,0)[0] intercept : AIC=947.708, Time=1.09 sec
ARIMA(5,1,0)(0,0,0)[0] intercept : AIC=1138.051, Time=0.18 sec
ARIMA(6,1,1)(0,0,0)[0] intercept : AIC=833.411, Time=1.28 sec
ARIMA(6,1,0)(0,0,0)[0] intercept : AIC=1060.199, Time=0.29 sec
ARIMA(7,1,1)(0,0,0)[0] intercept : AIC=724.613, Time=1.20 sec
ARIMA(7,1,0)(0,0,0)[0] intercept : AIC=905.168, Time=0.51 sec
ARIMA(8,1,1)(0,0,0)[0] intercept : AIC=655.728, Time=1.48 sec
ARIMA(8,1,0)(0,0,0)[0] intercept : AIC=inf, Time=0.88 sec
ARIMA(9,1,1)(0,0,0)[0] intercept : AIC=624.495, Time=1.78 sec
ARIMA(9,1,0)(0,0,0)[0] intercept : AIC=inf, Time=1.45 sec
ARIMA(10,1,1)(0,0,0)[0] intercept : AIC=590.889, Time=2.07 sec
ARIMA(10,1,0)(0,0,0)[0] intercept : AIC=inf, Time=1.90 sec
ARIMA(10,1,2)(0,0,0)[0] intercept : AIC=496.938, Time=2.33 sec
ARIMA(9,1,2)(0,0,0)[0] intercept : AIC=506.775, Time=1.91 sec
ARIMA(10,1,3)(0,0,0)[0] intercept : AIC=505.994, Time=2.32 sec
ARIMA(9,1,3)(0,0,0)[0] intercept : AIC=532.159, Time=2.36 sec
ARIMA(10,1,2)(0,0,0)[0] : AIC=693.807, Time=1.39 sec
Best model: ARIMA(10,1,2)(0,0,0)[0] intercept
Total fit time: 26.787 seconds
<class 'statsmodels.iolib.summary.Summary'>
"""
SARIMAX Results
==========================================================================
=
Dep. Variable: y No. Observations: 526
Model: SARIMAX(10, 1, 2) Log Likelihood -234.469
Date: Tue, 04 May 2021 AIC 496.938
Time: 14:11:30 BIC 556.626
Sample: 0 HQIC 520.311
- 526
Covariance Type: opg
==========================================================================
=
```

```
coef std err z P>|z| [0.025 0.975]
------------------------------------------------------------------------------
-
intercept 0.3105 0.028 11.163 0.000 0.256 0.365
ar.L1 0.7235 0.060 12.039 0.000 0.606 0.841
ar.L2 -0.8153 0.053 -15.298 0.000 -0.920 -0.711
ar.L3 -0.1545 0.060 -2.592 0.010 -0.271 -0.038
ar.L4 -0.1730 0.061 -2.850 0.004 -0.292 -0.054
ar.L5 -0.2430 0.065 -3.737 0.000 -0.371 -0.116
ar.L6 -0.2120 0.066 -3.228 0.001 -0.341 -0.083
ar.L7 -0.2672 0.063 -4.211 0.000 -0.392 -0.143
ar.L8 -0.2796 0.064 -4.349 0.000 -0.406 -0.154
ar.L9 -0.1906 0.057 -3.368 0.001 -0.301 -0.080
ar.L10 -0.2236 0.052 -4.294 0.000 -0.326 -0.122
ma.L1 -0.8896 0.047 -18.940 0.000 -0.982 -0.798
ma.L2 0.7777 0.037 21.013 0.000 0.705 0.850
sigma2 0.1406 0.008 17.405 0.000 0.125 0.156
==========================================================================
===
Ljung-Box (L1) (Q): 0.95 Jarque-Bera (JB): 7
.85
Prob(Q): 0.33 Prob(JB): 0
.02
Heteroskedasticity (H): 0.66 Skew: 0
.11
Prob(H) (two-sided): 0.01 Kurtosis: 3
.56
==========================================================================
===
```

Автопоиск предлагает использовать модель ARIMA (10,1,2). Протестируем ее.

```
# Create Training and Test
train = y[:int(y.size*0.9)]
test = y[int(y.size*0.9):]
# Build Model
model = ARIMA(train, order=(10, 1, 2))
fitted = model.fit()
# Forecast
forecast_res = fitted.get_forecast(test.size, alpha=0.05, dynamic=False)
# 95% conf
# forecast = fitted.forecast(test.size, alpha=0.05) # 95% conf
forecast = forecast_res.predicted_mean
# Make as pandas series
fc_series = pd.Series(forecast.values, index=test.index)
lower_series = pd.Series(forecast_res.conf_int()['lower co2'], index=test.
index)
upper_series = pd.Series(forecast_res.conf_int()['upper co2'], index=test.
index)
# Plot
plt.figure(figsize=(12,3), dpi=100)
plt.plot(train, label='training')
plt.plot(test, label='actual')
plt.plot(fc_series, label='forecast')
plt.fill_between(lower_series.index,
lower_series,
upper_series,
color='k',
```

```
alpha=0.15)
plt.title('Forecast vs Actuals')
plt.legend(loc='upper left', fontsize=12)
plt.show()
forecast_accuracy(fc_series.values, test.values)
{'mape': 0.01330633295669916,
'me': -4.89114134519186,
'mae': 4.914562517264946,
'mpe': -0.013241301241932851,
'rmse': 5.379130185554309,
'corr': 0.7410145431091859,
'minmax': 0.013306277017209989}
```

Как мы видим, автопоиск не гарантирует лучших результатов в прогнозе. Это связано с отсутствием перекрестной проверки, но мы, вероятно, сможем улучшить эту модель вручную.

```
# Create Training and Test
train = y[:int(y.size*0.9)]
test = y[int(y.size*0.9):]
# Build Model
model = ARIMA(train, order=(10, 2,8))
fitted = model.fit()
# Forecast
forecast_res = fitted.get_forecast(test.size, alpha=0.05, dynamic=False)
# 95% conf
# forecast = fitted.forecast(test.size, alpha=0.05) # 95% conf
forecast = forecast_res.predicted_mean
# Make as pandas series
fc_series = pd.Series(forecast.values, index=test.index)
lower_series = pd.Series(forecast_res.conf_int()['lower co2'], index=test.
index)
upper_series = pd.Series(forecast_res.conf_int()['upper co2'], index=test.
index)
# Plot
plt.figure(figsize=(12,3), dpi=100)
plt.plot(train, label='training')
plt.plot(test, label='actual')
plt.plot(fc_series, label='forecast')
plt.fill_between(lower_series.index,
lower_series,
upper_series,
color='k',
alpha=0.15)
plt.title('Forecast vs Actuals')
plt.legend(loc='upper left', fontsize=12)
plt.show()
forecast_accuracy(fc_series.values, test.values)
{'mape': 0.0037188201927048957,
'me': -1.3349999198951579,
'mae': 1.370467806185665,
'mpe': -0.0036203570222564747,
'rmse': 1.448243682381261,
'corr': 0.984801545546153,
'minmax': 0.003718716705316538}
```

Упражнение _____1

1. Попробуйте смоделировать процесс случайного блуждания (из работы №2) и выберите для него лучшую модель ARIMA.

2. Попробуйте смоделировать некоторый временной ряд с небольшой сезонностью, логистическим трендом, небольшим эффектом праздников и найдите для этого лучшую модель ARIMA.
3. Возьмите набор данных пассажира авиалайнера (из работы № 4) и попытайтесь найти лучшую модель для его прогнозирования.

Сезонная модель ARIMA (SARIMA)

Важность сезонной производной

Проблема с простой моделью ARIMA в том, что она не подразумевает нестационарную сезонность. Если для временного ряда имеет место значительный эффект сезонности, тогда следует выбирать модель SARIMA. В этой модели используются сезонные производные.
Сезонная производная аналогична обычной производной, но вместо вычитания последовательных членов вы вычитаете значение из предыдущего периода сезонности.
Примечания. При работе с сезонными эффектами мы используем сезонный ARIMA (SARIMA), который обозначается как SARIMA (p, d, q) (P, D, Q) s. Здесь (p, d, q) являются несезонными параметрами, описанными выше, а (P, D, Q) следуют тому же порядку определений, но применяются к сезонной составляющей временного ряда. Член s - это периодичность временного ряда (4 для квартальных периодов, 12 для годовых периодов и т.д.).
Для начала давайте посмотрим, как работает сезонное дифференцирование

```
SEASON = 12
# Plot
fig, axes = plt.subplots(4, 1, figsize=(12,8), dpi=100, sharex=True)
# Original Series
axes[0].plot(y[:])
axes[0].set_title('Original Series')
# Usual Differencing
axes[1].plot(y[:].diff(1))
axes[1].set_title('Usual Differencing')
# Seasinal Differencing
axes[2].plot(y[:].diff(SEASON))
axes[2].set_title('Seasonal Differencing')
# Seasinal and Usual Differencing
axes[3].plot(y[:].diff(1).diff(SEASON))
axes[3].set_title('Usual and Seasonal Differencing')
plt.suptitle('Dataset $CO_2$', fontsize=12)
plt.show()
```

Как мы видим, сезонная разница может помочь сделать данные более стационарными.
Примечание. Как мы можем видеть на графике обычной разницы, у нас есть как минимум две сезонные составляющие с разными периодами, но, взяв одну сезонную разницу, мы исключаем почти все сезонные влияния. Посмотрим на спектр.

```
SEASON = 12
def afft(x):
x_np = x.dropna().values
return np.abs(np.fft.fft(x_np))[:x_np.size//2] # the spectrum is mirro
red relative to the middle point
# Plot
fig, axes = plt.subplots(5, 2, figsize=(12,12), dpi=100)
# Original Series
axes[0,0].plot(y[:])
axes[0,0].set_title('Original Series')
axes[0,1].plot(afft(y))
axes[0,1].set_title('Spectrum of Original Series')
# Usual Differencing
axes[1,0].plot(y[:].diff(1))
```

```
axes[1,0].set_title('Usual Differencing')
axes[1,1].plot(afft(y.diff(1)))
axes[1,1].set_title('Spectrum of Usual Differencing')
# Usual Differencing
axes[2,0].plot(y[:].diff(1).diff(1))
axes[2,0].set_title('Second Usual Differencing')
axes[2,1].plot(afft(y.diff(1).diff(1)))
axes[2,1].set_title('Spectrum of Second Usual Differencing')
# Seasinal Differencing
axes[3,0].plot(y[:].diff(SEASON))
axes[3,0].set_title('Seasonal Differencing')
axes[3,1].plot(afft(y.diff(SEASON)))
axes[3,1].set_title('Spectrum of Seasonal Differencing')
# Seasinal and Usual Differencing
axes[4,0].plot(y[:].diff(1).diff(SEASON))
axes[4,0].set_title('Usual and Seasonal Differencing')
axes[4,1].plot(afft(y.diff(1).diff(SEASON)))
axes[4,1].set_title('Spectrum of Usual and Seasonal Differencing')
plt.suptitle('Dataset $CO_2$', fontsize=12)
fig.tight_layout()
plt.show()
```

Как мы видим на графиках выше, спектр обычной разности содержит по крайней мере 4 компоненты (пика), но все с одним и тем же шагом (то есть один с периодом 12, следующий с периодом 24 и так далее). Благодаря этому используя разность с периодом 12 мы исключаем все сезонные составляющие на нижнем графике.

Также необходимо отметить, что тренд - это самая низкочастотная часть (см. Начало спектра). Таким образом, мы практически исключаем влияние тренда. При этом оставшуюся часть влияния тренда мы исключаем, беря вторую разницу. Это подтверждает наше предположение о необходимости дифференцирования порядка в приведенных выше примерах.

Упражнения 2

1. Проверьте значения KPSS и ADF для исходных данных; данных с обычной производной; данных со второй обычной производной; данные с сезонной производной; и данные с обоими производными.
2. Проверьте графики ACF и PACF и сделайте выводы о порядке модели для модели со второй обычной производной; для данных с сезонной производной; и для данных с обоими производными.
3. Проверьте и докажите, почему вам не нужно брать 3-ю обычную производную и вторую сезонную производную.

Выбор порядка модели SARIMA

В объяснении выше мы отметили, что сезонная разница делает данные более стационарными. Затем нам нужно выбрать наилучшее начальное предположение о порядках модели SARIMA.

Правила выбора начальных порядков

· Правильный порядок d - это порядок дифференцирования, который дает временной ряд с шумоподобным поведением, т.е. случайные колебания около четко определенного среднего значения с почти постоянным разбросом, проверьте на стационарность по критериям, указанным выше. Если временной ряд имеет положительные значения ACF с большим значением лага добавить обычную производную.

· Используйте сезонную производную D только в случае сильного влияния сезонности для модели.

· Количество слагаемых AR (порядок) определяется как последнее значение лага PACF перед быстрым уменьшением от положительных значений до нуля.

· Количество слагаемых скользящего среднего (MA) определяется как последнее значение лага ACF перед быстрым увеличением от отрицательных значений до

нуля.
· Добавьте слагаемое SAR, если значения ACF периодически положительная.
· Помимо этого, порядок SAR может быть оценен из PACF. Посмотрите на количество значений лагов, которые кратны периоду сезона. Например, если период равен 24, и мы видим, что 24-е и 48-е запаздывания значительны в PACF, это означает, что начальное P должно быть 2.
· Добавьте член SMA, если значения ACF периодически отрицательный. Используйте те же правила определения количества лагов, что и для SAR.
· Если ваш временной ряд немного недодифференцирован, добавьте дополнительное слагаемое в AR.
· Если ваши ряды немного передифференцирован, добавьте дополнительные слагаемое в MA.
· Старайтесь избегать использования более одного или двух сезонных порядков (SAR + SMA) в одной модели, так как это может привести к переобучению данных и/или проблемам в точности оценок.

```
import pmdarima as pm
# Seasonal - fit stepwise auto-ARIMA
smodel = pm.auto_arima(train,
start_p=10, #Search for Usual AR order
start_q=1, #Search for Usual MA order
test='adf',
max_p=10,
max_q=10,
d=None, #Search for Usual Difference Order
m=12, #The period for seasonal differencing
seasonal=True, #SARIMA ENABLE
start_P=0, #Search for Seasonal AR order
start_Q=0, #Search for Seasonal MA order
D=None, #Search for Seasonal Difference Order
trace=True,
error_action='ignore',
suppress_warnings=True,
stepwise=True)
smodel.summary()
Performing stepwise search to minimize aic
ARIMA(10,1,1)(0,0,0)[12] intercept : AIC=519.755, Time=1.98 sec
ARIMA(0,1,0)(0,0,0)[12] intercept : AIC=1500.491, Time=0.01 sec
ARIMA(1,1,0)(1,0,0)[12] intercept : AIC=inf, Time=0.49 sec
ARIMA(0,1,1)(0,0,1)[12] intercept : AIC=1025.470, Time=0.29 sec
ARIMA(0,1,0)(0,0,0)[12] : AIC=1502.018, Time=0.01 sec
ARIMA(10,1,1)(1,0,0)[12] intercept : AIC=513.823, Time=5.88 sec
ARIMA(10,1,1)(2,0,0)[12] intercept : AIC=483.623, Time=20.12 sec
ARIMA(10,1,1)(2,0,1)[12] intercept : AIC=inf, Time=18.83 sec
ARIMA(10,1,1)(1,0,1)[12] intercept : AIC=inf, Time=5.53 sec
ARIMA(9,1,1)(2,0,0)[12] intercept : AIC=481.660, Time=15.57 sec
ARIMA(9,1,1)(1,0,0)[12] intercept : AIC=564.281, Time=4.26 sec
ARIMA(9,1,1)(2,0,1)[12] intercept : AIC=435.565, Time=16.06 sec
ARIMA(9,1,1)(1,0,1)[12] intercept : AIC=inf, Time=4.43 sec
ARIMA(9,1,1)(2,0,2)[12] intercept : AIC=inf, Time=27.89 sec
ARIMA(9,1,1)(1,0,2)[12] intercept : AIC=433.047, Time=10.23 sec
ARIMA(9,1,1)(0,0,2)[12] intercept : AIC=534.469, Time=13.22 sec
ARIMA(9,1,1)(0,0,1)[12] intercept : AIC=537.707, Time=2.59 sec
ARIMA(8,1,1)(1,0,2)[12] intercept : AIC=428.678, Time=7.85 sec
ARIMA(8,1,1)(0,0,2)[12] intercept : AIC=552.529, Time=5.32 sec
ARIMA(8,1,1)(1,0,1)[12] intercept : AIC=431.771, Time=4.36 sec
ARIMA(8,1,1)(2,0,2)[12] intercept : AIC=inf, Time=16.16 sec
```

```
ARIMA(8,1,1)(0,0,1)[12] intercept : AIC=557.384, Time=2.26 sec
ARIMA(8,1,1)(2,0,1)[12] intercept : AIC=440.569, Time=13.39 sec
ARIMA(7,1,1)(1,0,2)[12] intercept : AIC=391.199, Time=7.25 sec
ARIMA(7,1,1)(0,0,2)[12] intercept : AIC=601.475, Time=5.55 sec
ARIMA(7,1,1)(1,0,1)[12] intercept : AIC=382.841, Time=3.62 sec
ARIMA(7,1,1)(0,0,1)[12] intercept : AIC=611.099, Time=2.03 sec
ARIMA(7,1,1)(1,0,0)[12] intercept : AIC=564.036, Time=2.93 sec
ARIMA(7,1,1)(2,0,1)[12] intercept : AIC=385.415, Time=11.30 sec
ARIMA(7,1,1)(0,0,0)[12] intercept : AIC=638.723, Time=1.16 sec
ARIMA(7,1,1)(2,0,0)[12] intercept : AIC=480.526, Time=9.47 sec
ARIMA(7,1,1)(2,0,2)[12] intercept : AIC=inf, Time=10.34 sec
ARIMA(6,1,1)(1,0,1)[12] intercept : AIC=382.791, Time=4.19 sec
ARIMA(6,1,1)(0,0,1)[12] intercept : AIC=682.219, Time=1.92 sec
ARIMA(6,1,1)(1,0,0)[12] intercept : AIC=559.635, Time=2.86 sec
ARIMA(6,1,1)(2,0,1)[12] intercept : AIC=384.434, Time=10.86 sec
ARIMA(6,1,1)(1,0,2)[12] intercept : AIC=389.497, Time=6.65 sec
ARIMA(6,1,1)(0,0,0)[12] intercept : AIC=732.258, Time=0.74 sec
ARIMA(6,1,1)(0,0,2)[12] intercept : AIC=663.277, Time=4.05 sec
ARIMA(6,1,1)(2,0,0)[12] intercept : AIC=479.604, Time=10.68 sec
ARIMA(6,1,1)(2,0,2)[12] intercept : AIC=inf, Time=11.57 sec
ARIMA(5,1,1)(1,0,1)[12] intercept : AIC=387.764, Time=2.54 sec
ARIMA(6,1,0)(1,0,1)[12] intercept : AIC=386.595, Time=3.07 sec
ARIMA(6,1,2)(1,0,1)[12] intercept : AIC=385.992, Time=3.48 sec
ARIMA(5,1,0)(1,0,1)[12] intercept : AIC=388.336, Time=2.22 sec
ARIMA(5,1,2)(1,0,1)[12] intercept : AIC=384.240, Time=2.86 sec
ARIMA(7,1,0)(1,0,1)[12] intercept : AIC=437.263, Time=3.05 sec
ARIMA(7,1,2)(1,0,1)[12] intercept : AIC=382.793, Time=3.90 sec
ARIMA(6,1,1)(1,0,1)[12] : AIC=365.821, Time=2.90 sec
ARIMA(6,1,1)(0,0,1)[12] : AIC=765.356, Time=0.67 sec
ARIMA(6,1,1)(1,0,0)[12] : AIC=552.143, Time=1.60 sec
ARIMA(6,1,1)(2,0,1)[12] : AIC=363.673, Time=11.63 sec
ARIMA(6,1,1)(2,0,0)[12] : AIC=477.943, Time=3.26 sec
ARIMA(6,1,1)(2,0,2)[12] : AIC=365.390, Time=9.48 sec
ARIMA(6,1,1)(1,0,2)[12] : AIC=363.652, Time=5.92 sec
ARIMA(6,1,1)(0,0,2)[12] : AIC=724.284, Time=1.50 sec
ARIMA(5,1,1)(1,0,2)[12] : AIC=inf, Time=9.47 sec
ARIMA(6,1,0)(1,0,2)[12] : AIC=365.332, Time=4.45 sec
ARIMA(7,1,1)(1,0,2)[12] : AIC=364.297, Time=6.05 sec
ARIMA(6,1,2)(1,0,2)[12] : AIC=inf, Time=7.69 sec
ARIMA(5,1,0)(1,0,2)[12] : AIC=365.808, Time=3.46 sec
ARIMA(5,1,2)(1,0,2)[12] : AIC=inf, Time=6.37 sec
ARIMA(7,1,0)(1,0,2)[12] : AIC=364.445, Time=5.01 sec
ARIMA(7,1,2)(1,0,2)[12] : AIC=inf, Time=7.35 sec
Best model: ARIMA(6,1,1)(1,0,2)[12]
Total fit time: 411.861 seconds
<class 'statsmodels.iolib.summary.Summary'>
"""
SARIMAX Results
==========================================================================
===
Dep. Variable: y No. Observations:
473
Model: SARIMAX(6, 1, 1)x(1, 0, [1, 2], 12) Log Likelihood
-170.826
Date: Mon, 03 May 2021 AIC
363.652
```

```
Time: 16:45:14 BIC
409.378
Sample: 0 HQIC
381.638
- 473
Covariance Type: opg
==============================================================================
=
coef std err z P>|z| [0.025 0.975]
------------------------------------------------------------------------------
-
ar.L1 0.2816 0.144 1.956 0.050 -0.001 0.564
ar.L2 -0.0111 0.033 -0.337 0.736 -0.076 0.053
ar.L3 -0.1518 0.063 -2.425 0.015 -0.274 -0.029
ar.L4 0.0018 0.036 0.050 0.960 -0.069 0.073
ar.L5 0.0127 0.045 0.282 0.778 -0.076 0.101
ar.L6 -0.0934 0.045 -2.079 0.038 -0.181 -0.005
ma.L1 -0.5693 0.136 -4.187 0.000 -0.836 -0.303
ar.S.L12 0.9995 0.001 1926.967 0.000 0.998 1.000
ma.S.L12 -0.8511 0.044 -19.237 0.000 -0.938 -0.764
ma.S.L24 -0.0268 0.042 -0.634 0.526 -0.110 0.056
sigma2 0.1088 0.006 19.283 0.000 0.098 0.120
==============================================================================
===
===
Ljung-Box (L1) (Q): 0.04 Jarque-Bera (JB): 99
.33
Prob(Q): 0.83 Prob(JB): 0
.00
Heteroskedasticity (H): 0.60 Skew: 0
.37
Prob(H) (two-sided): 0.00 Kurtosis: 5
.12
==============================================================================
===
```

Метод автопоиска дал порядок модели SARIMAX(6, 1, 1)x(1, 0, [1, 2], 12)

```
fitted, confint = smodel.predict(n_periods=test.size,
return_conf_int=True)
# make series for plotting purpose
fc_series = pd.Series(fitted, index=test.index)
lower_series = pd.Series(confint[:, 0], index=test.index)
upper_series = pd.Series(confint[:, 1], index=test.index)
# Plot
plt.figure(figsize=(12,3), dpi=100)
plt.plot(train, label='training')
plt.plot(test, label='actual')
plt.plot(fc_series, label='forecast')
plt.fill_between(lower_series.index,
lower_series,
upper_series,
color='k',
alpha=0.15)
plt.title('$SARIMA_{12}$ Forecast vs Actuals')
plt.legend(loc='upper left', fontsize=12)
plt.show()
forecast_accuracy(fc_series.values, test.values)
```

```
{'mean absoute percentage error': 0.003947772742976413,
'mean absoute error ': 1.454722139505576,
'mean percentage error ': -0.0038805878660586987,
'root mean square ': 1.5338558138656706,
'correlation coefficient ': 0.9846420347534222,
'minmax error ': 0.003947716428085446}
```

Теперь попробуем с функцией из statsmodels

```
from statsmodels.tsa.statespace.sarimax import SARIMAX
model = sm.tsa.statespace.SARIMAX(train,
order=(10, 2, 8),
seasonal_order=(1, 1, 2, 12))
fitted = model.fit()
# Forecast
forecast_res = fitted.get_forecast(test.size, alpha=0.05, dynamic=False)
# 95% conf
# forecast = fitted.forecast(test.size, alpha=0.05) # 95% conf
forecast = forecast_res.predicted_mean
# Make as pandas series
fc_series = pd.Series(forecast.values, index=test.index)
lower_series = pd.Series(forecast_res.conf_int()['lower co2'], index=test.
index)
upper_series = pd.Series(forecast_res.conf_int()['upper co2'], index=test.
index)
# Plot
plt.figure(figsize=(12,3), dpi=100)
plt.plot(train, label='training')
plt.plot(test, label='actual')
plt.plot(fc_series, label='forecast')
plt.fill_between(lower_series.index,
lower_series,
upper_series,
color='k',
alpha=0.15)
plt.title('Forecast vs Actuals')
plt.legend(loc='upper left', fontsize=12)
plt.show()
forecast_accuracy(fc_series.values, test.values)
{'mean absoute percentage error': 0.0031670916924645934,
'mean absoute error ': 1.166196574986573,
'mean percentage error ': -0.0031093048607534345,
'root mean square ': 1.2596568798934529,
'correlation coefficient ': 0.9850223690012644,
'minmax error ': 0.0031670475141546417}
```

Упражнение 3

1. Сравните результаты, полученные с помощью автопоиска, с нашим исходным предположением о первой обычной производной и первой сезонной производной. Сравните предыдущие результаты с теми, что получены по второй сезонной производной. Выберите порядок модели из двух предыдущих проведите для него основные тесты.

Модель SARIMAX (SARIMA с экзогенными регрессорами)

В некоторых случаях мы можем повысить точность прогноза модели, введя экзогенную переменную. Основным требованием для использования экзогенной переменной является то, что вам необходимо знать значение переменной в течение тренировочной выборки и в период прогноза. В целом экзогенная переменная может быть какой угодно, независимо от обучающих данных.

Для примера в наших примерах мы можем извлечь компонент сезона из тестовых

данных и рассматривать его как экзогенную переменную для обучающих данных. Мы сделаем это с помощью процедуры Season_decompose.

```
#ReSample to DataFrame
y = y_['co2'].resample('MS').mean()
# The term bfill means that we use the value before filling in missing val
ues
y = y.fillna(y.bfill())
ydf = pd.DataFrame(y)
ydf.columns = ['endog']
ydf.head()
endog
1958-03-01 316.100000
1958-04-01 317.200000
1958-05-01 317.433333
1958-06-01 315.625000
1958-07-01 315.625000
```

Создадим новую выборку co2 как тренировочную (endog) и создадим экзогенную выборку (exog)

```
# Compute Seasonal Index
from statsmodels.tsa.seasonal import seasonal_decompose
y = y_['co2'].resample('MS').mean()
# The term bfill means that we use the value before filling in missing val
ues
y = y.fillna(y.bfill())
ydf = pd.DataFrame(y)
ydf.columns = ['endog']
ydf.head()
# multiplicative seasonal component
result_mul = seasonal_decompose(ydf.endog, # 3 years
model='multiplicative')
seasonal_index = result_mul.seasonal[-24:].to_frame()
seasonal_index['month'] = pd.to_datetime(seasonal_index.index).month
# merge with the base data
ydf['month'] = ydf.index.month
ydf = pd.merge(ydf, seasonal_index, how='left', on='month')
ydf.columns = ['endog', 'month', 'exog']
ydf.head(12)
endog month exog
0 316.100000 3 1.004239
1 316.100000 3 1.004239
2 317.200000 4 1.007380
3 317.200000 4 1.007380
4 317.433333 5 1.008577
5 317.433333 5 1.008577
6 315.625000 6 1.006689
7 315.625000 6 1.006689
8 315.625000 7 1.002177
9 315.625000 7 1.002177
10 314.950000 8 0.996102
11 314.950000 8 0.996102
ydf.exog.plot();
```

Разделим выбоки

```
train , test = pm.model_selection.train_test_split(ydf,test_size=0.1)
# Seasonal - fit stepwise auto-ARIMA
sxmodel = pm.auto_arima(train.endog,
exogenous = train.exog.values.reshape(-1,1),
```

```
start_p=10, #Search for Usual AR order
start_q=1, #Search for Usual MA order
test='adf',
max_p=10,
max_q=10,
d=None, #Search for Usual Difference Order
m=12, #The period for seasonal differencing
seasonal=True, #SARIMA ENABLE
start_P=0, #Search for Seasonal AR order
start_Q=0, #Search for Seasonal MA order
D=None, #Search for Seasonal Difference Order
trace=True,
error_action='ignore',
suppress_warnings=True,
stepwise=True)
sxmodel.summary()
Performing stepwise search to minimize aic
ARIMA(10,1,1)(0,0,0)[12] intercept : AIC=296.674, Time=2.31 sec
ARIMA(0,1,0)(0,0,0)[12] intercept : AIC=332.551, Time=0.08 sec
ARIMA(1,1,0)(1,0,0)[12] intercept : AIC=316.706, Time=0.30 sec
ARIMA(0,1,1)(0,0,1)[12] intercept : AIC=311.755, Time=0.35 sec
ARIMA(0,1,0)(0,0,0)[12] : AIC=372.107, Time=0.08 sec
ARIMA(10,1,1)(1,0,0)[12] intercept : AIC=296.680, Time=4.98 sec
ARIMA(10,1,1)(0,0,1)[12] intercept : AIC=296.619, Time=2.98 sec
ARIMA(10,1,1)(1,0,1)[12] intercept : AIC=299.271, Time=5.72 sec
ARIMA(10,1,1)(0,0,2)[12] intercept : AIC=298.551, Time=9.68 sec
ARIMA(10,1,1)(1,0,2)[12] intercept : AIC=300.372, Time=8.92 sec
ARIMA(9,1,1)(0,0,1)[12] intercept : AIC=295.174, Time=2.97 sec
ARIMA(9,1,1)(0,0,0)[12] intercept : AIC=295.872, Time=3.38 sec
ARIMA(9,1,1)(1,0,1)[12] intercept : AIC=298.494, Time=6.89 sec
ARIMA(9,1,1)(0,0,2)[12] intercept : AIC=297.171, Time=18.97 sec
ARIMA(9,1,1)(1,0,0)[12] intercept : AIC=295.241, Time=5.43 sec
ARIMA(9,1,1)(1,0,2)[12] intercept : AIC=299.335, Time=15.26 sec
ARIMA(8,1,1)(0,0,1)[12] intercept : AIC=295.533, Time=2.22 sec
ARIMA(9,1,0)(0,0,1)[12] intercept : AIC=295.512, Time=1.05 sec
ARIMA(9,1,2)(0,0,1)[12] intercept : AIC=297.423, Time=2.92 sec
ARIMA(8,1,0)(0,0,1)[12] intercept : AIC=293.675, Time=0.96 sec
ARIMA(8,1,0)(0,0,0)[12] intercept : AIC=295.455, Time=0.55 sec
ARIMA(8,1,0)(1,0,1)[12] intercept : AIC=295.873, Time=1.87 sec
ARIMA(8,1,0)(0,0,2)[12] intercept : AIC=295.664, Time=2.39 sec
ARIMA(8,1,0)(1,0,0)[12] intercept : AIC=293.714, Time=1.43 sec
ARIMA(8,1,0)(1,0,2)[12] intercept : AIC=296.049, Time=6.62 sec
ARIMA(7,1,0)(0,0,1)[12] intercept : AIC=296.960, Time=0.66 sec
ARIMA(7,1,1)(0,0,1)[12] intercept : AIC=296.689, Time=1.33 sec
ARIMA(8,1,0)(0,0,1)[12] : AIC=360.138, Time=1.20 sec
Best model: ARIMA(8,1,0)(0,0,1)[12] intercept
Total fit time: 111.542 seconds
<class 'statsmodels.iolib.summary.Summary'>
"""
SARIMAX Results
==========================================================================
===
Dep. Variable: y No. Observations:
473
Model: SARIMAX(8, 1, 0)x(0, 0, [1], 12) Log Likelihood
-134.837
```

```
Date: Mon, 03 May 2021 AIC
293.675
Time: 22:38:33 BIC
343.558
Sample: 0 HQIC
313.297
- 473
Covariance Type: opg
==============================================================================
=
coef std err z P>|z| [0.025 0.975]
------------------------------------------------------------------------------
-
intercept 0.2309 0.030 7.823 0.000 0.173 0.289
x1 336.3614 4.631 72.635 0.000 327.285 345.438
ar.L1 -0.2627 0.037 -7.096 0.000 -0.335 -0.190
ar.L2 -0.1614 0.039 -4.099 0.000 -0.239 -0.084
ar.L3 -0.2515 0.047 -5.318 0.000 -0.344 -0.159
ar.L4 -0.1415 0.052 -2.707 0.007 -0.244 -0.039
ar.L5 -0.0650 0.055 -1.173 0.241 -0.174 0.044
ar.L6 -0.1158 0.053 -2.173 0.030 -0.220 -0.011
ar.L7 -0.1117 0.048 -2.312 0.021 -0.206 -0.017
ar.L8 -0.1129 0.048 -2.358 0.018 -0.207 -0.019
ma.S.L12 0.0983 0.048 2.044 0.041 0.004 0.193
sigma2 0.1036 0.005 20.190 0.000 0.094 0.114
==============================================================================
===
Ljung-Box (L1) (Q): 0.00 Jarque-Bera (JB): 157
.98
Prob(Q): 0.96 Prob(JB): 0
.00
Heteroskedasticity (H): 0.60 Skew: 0
.38
Prob(H) (two-sided): 0.00 Kurtosis: 5
.73
==============================================================================
===
"""
```

Попробуем предсказание.

```
ex4test = pd.DataFrame(test.exog)
ex4test.head()
exog
20752 0.996102
20753 0.996102
20754 0.996102
20755 0.996102
20756 0.996102
fitted, confint = sxmodel.predict(n_periods=np.shape(test.exog.values)[0],
exogenous=ex4test,
return_conf_int=True)
# make series for plotting purpose
fc_series = pd.Series(fitted, index=test.index)
lower_series = pd.Series(confint[:, 0], index=test.index)
upper_series = pd.Series(confint[:, 1], index=test.index)
# Plot
plt.figure(figsize=(12,3), dpi=100)
```

```
plt.plot(train.endog, label='training')
plt.plot(test.endog, label='actual')
plt.plot(fc_series, label='forecast')
plt.fill_between(lower_series.index,
lower_series,
upper_series,
color='k',
alpha=0.15)
plt.title('$SARIMAX_{12}$ Forecast vs Actuals')
plt.legend(loc='upper left', fontsize=12)
plt.show()
forecast_accuracy(fc_series.values, test.endog)
{'mean absoute percentage error': 0.005060117981539687,
'mean absoute error ': 1.8659586807355004,
'mean percentage error ': -0.0049627858435570835,
'root mean square ': 1.9576150319647736,
'correlation coefficient ': 0.9820501688597131,
'minmax error ': 0.0050600257762539735}
```

Теперь попробуем модель SARIMAX из statsmodels.

```
from statsmodels.tsa.statespace.sarimax import SARIMAX
model = sm.tsa.statespace.SARIMAX(endog=train.endog,
exog =train.exog,
order=(8, 2, 8),
seasonal_order=(0, 0, 1, 12))
fitted = model.fit()
# Forecast
forecast_res = fitted.get_forecast(test.endog.size,
exog = test.exog,
alpha = 0.05,
dynamic = False) # 95% conf
forecast = forecast_res.predicted_mean
# Make as pandas series
fc_series = pd.Series(forecast.values,
index=test.index)
lower_series = pd.Series(forecast_res.conf_int()['lower endog'], index=tes
t.index)
upper_series = pd.Series(forecast_res.conf_int()['upper endog'], index=tes
t.index)
# Plot
plt.figure(figsize=(12,3), dpi=100)
plt.plot(train.endog, label='training')
plt.plot(test.endog, label='actual')
plt.plot(fc_series, label='forecast')
plt.fill_between(lower_series.index,
lower_series,
upper_series,
color='k',
alpha=0.15)
plt.title('Forecast vs Actuals')
plt.legend(loc='upper left', fontsize=12)
plt.show()
forecast_accuracy(fc_series.values, test.endog.values)
{'mean absoute percentage error': 0.0036812992027410066,
'mean absoute error ': 1.356824961133212,
'mean percentage error ': -0.0035763191213044553,
'root mean square ': 1.4331867044616442,
```

```
'correlation coefficient ': 0.9855394369808252,
'minmax error ': 0.0036812000258082955}
```

Для нашего игрушечного случая мы не получили большого прироста точности, вероятно это связано с тем, что экзогенные данные выбраны из той же выборки, что и тренировочные, и не вносят новой информации.

Упражнение 4

1. Попробуйте использовать тренд тестовых данных в качестве экзогенных факторов вместо сезонности.__

Лабораторная работа 6

Классификация временных рядов

Классификация одномерных временных рядов с использование методов машинного обучения библиотек sklearn и sktime. Представление временных рядов для задач классификации. Использование традиционных методов машинного обучения библиотеки sklearn для классификации временных рядов. Использование специальных методов sktime: временное дерево и временной лес, расстояние DTW и метод dtw-knn, классификаторы на основе словарей. Классификатор rocket.

Импорт библиотек и данных

```
!pip install -U pyts
!pip install -U tslearn
!pip install -U sktime
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
import pandas as pd
```

Для тестирования классификаторов временных рядов мы будем использовать небольшой набор данных GunPoint из библиотеки pyts. Набор данных включает в себя одну женщину-актера и одного актера-мужчину, которые делают движение рукой как будто достают пистолет. В первом классе они вынимают копию пистолета из кобуры, закрепленной на бедре, наводят его на цель примерно на одну секунду, затем возвращают пистолет в кобуру и складывают руки по бокам. Во втором классе актеры держат пистолеты по бокам. Они указывают указательными пальцами на цель примерно на одну секунду, а затем разводят руками по бокам. Для обоих классов отслеживается центроиды правой руки актера по осям X и Y, которые, по-видимому, сильно коррелированы.

```
from pyts.datasets import load_gunpoint
x_train,x_test, y_train, y_test = load_gunpoint(return_X_y=True)
```

Давайте посмотрим на классы.

```
classes = np.unique(np.concatenate((y_train, y_test), axis=0))
print('calss lables',classes)
print('train x',x_train.shape,'train lables', y_train.shape,'test
x',x_test.shape,'test lables', y_test.shape)
plt.figure()
for c in classes:
c_x_train = x_train[y_train == c]
plt.plot(c_x_train[0], label="class " + str(c))
plt.legend(loc="best")
plt.show()
plt.close()
```

```
calss lables [1 2]
train x (50, 150) train lables (50,) test x (150, 150) test lables (150,)
```

Библиотека SKTime использует различные форматы данных в задачах классификации. Вы можете прочитать об этом в документации. Для преобразования данных из обычных данных в набор для SKTime мы будем использовать функцию from_2d_array_to_nested из sktime.utils.data_processing.

Примечание. Многие хранилища временных рядов используют собственные форматы данных. Для преобразования формата данных вы можете использовать некоторые утилиты, из tslearn.utils.

```
from tslearn.utils import from_pyts_dataset, to_sktime_dataset
X_train = to_sktime_dataset(from_pyts_dataset(x_train))
X_test = to_sktime_dataset(from_pyts_dataset(x_test))
X_train.head()
```

```
from sktime.utils.data_processing import from_2d_array_to_nested
X_train = from_2d_array_to_nested(x_train)
X_test = from_2d_array_to_nested(x_test)
X_train.head()
0
0 0 -0.647885
1 -0.641992
2 -0.63818...
1 0 -0.644427
1 -0.645401
2 -0.64705...
2 0 -0.778353
1 -0.778279
2 -0.77715...
3 0 -0.750060
1 -0.748103
2 -0.74616...
4 0 -0.599539
1 -0.597422
2 -0.59926...
```

Исследование классификаторов

Методы пакета Sklearn

Для начала попробуем использовать классификатор SKlearn в качестве основы. Здесь мы не будем использовать измененный формат данных. Если вы собираетесь использовать вложенные данные SKTime в SKLearn, используйте from_nested_to_2d_arrayfrom sktime.utils.data_processing или Tabularizer в конвейерах sklearn..

```
# let's get a baseline for comparison
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators=100)
classifier.fit(x_train, y_train)
classifier.score(x_test, y_test)
0.9266666666666666
```

Упражнение 1

1. Попробуйте еще несколько классификаторов из `sklearn.` Например, SVM, KNN или еще какие-то на ваш выбор.

Методы пакета Sktime

Дерево временных рядов.

Дерево временных рядов представляет собой модификацию алгоритма дерева классификации для временных рядов. В этом методе предлагается разбить выборку на случайные интервалы, извлечь три признака (среднее, стандартное отклонение и наклон) из каждого интервала, обучить дерево решений по извлеченным признакам.

```
from sktime.transformations.panel.summarize import
RandomIntervalFeatureExtractor
from sklearn.tree import DecisionTreeClassifier
from sktime.utils.slope_and_trend import _slope
steps = [
("extract",
RandomIntervalFeatureExtractor(
n_intervals="sqrt",
features=[np.mean, np.std, _slope])),
("clf", DecisionTreeClassifier()),
]
time_series_tree = Pipeline(steps)
time_series_tree.fit(X_train, y_train)
time_series_tree.score(X_test, y_test)
0.8533333333333334
```

Лес временных рядов

Фактически, мы можем объединить несколько деревьев для повышения точности и надежности.

```
tsf = ComposableTimeSeriesForestClassifier(
estimator=time_series_tree,
n_estimators=100)
tsf.fit(X_train, y_train)
tsf.score(X_test, y_test)
0.9466666666666667
```

На самом деле мы можем использовать встроенный классификатор лес временных рядов (TSF). TSF здесь представляет собой ансамбль древовидных классификаторов, построенных на полученных статистиках по случайно выбранным интервалам. Для каждого дерева интервалы выбираются случайным образом. Полное количество интервалов $\sqrt{series\_length}$. Из каждого из этих интервалов извлекаются среднее значение, стандартное отклонение и наклон и объединяются в вектор признаков. Эти новые функции затем используются для построения дерева, которое добавляется к ансамблю.

```
from sktime.classification.interval_based import
TimeSeriesForestClassifier
tsf = TimeSeriesForestClassifier(n_estimators=100, random_state=47)
tsf.fit(X_train, y_train)
tsf.score(X_test, y_test)
0.9666666666666667
```

Спектральный ансамбль со случайными интервалами (RISE)

Еще одним популярным вариантом леса временных рядов является так называемый спектральный ансамбль со случайными интервалами (RISE), который использует несколько преобразователей выделения признаков от ряда к ряду, в том числе: коэффициенты автокорреляции и коэффициенты спектральной мощности. Подобно Лесу временных рядов, здесь извлечение всех признаков производится на случайных интервалах временных рядов для нескольких деревьев.

```
from sktime.classification.interval_based import
RandomIntervalSpectralForest
rise = RandomIntervalSpectralForest(n_estimators=10, random_state=47)
rise.fit(X_train, y_train)
rise.score(X_test, y_test)
0.94
```

Классификатор на основе расстояния с динамическим искажением времени (DTW)

Для временных рядов наиболее популярный алгоритм k-ближайших соседей основан на измерении расстояния с динамическим искажением времени (DTW). Алгоритм DTW состоит из следующих шагов:

· Вычислить расстояние между первой точкой в сегменте первом сегменте ряда и каждой точкой во втором сегменте.

· Выберите минимум вычисленных значений и сохраните его (это этап «деформации времени»).

· Перейдите ко второй точке и повторите этап 1.

· Двигайтесь шаг за шагом по точкам и повторяйте этап 1, пока не будут исчерпаны все точки.

· Вычислите и выберите минимальное расстояние между первой точкой во втором сегменте серии и каждой точкой в первой серии.

· Двигайтесь шаг за шагом по точкам во втором сегменте и повторяйте этап 3, пока не будут исчерпаны все точки.

· Просуммируйте все сохраненные минимальные расстояния.

Алгоритм DTW формально можно описать как

$D(i,j) = 0$

$D(i,j) = dist(xi, yj\ ) + min\{D(i-1,j), D(i,j-1), D(i-1,j-1)\}$

$dDWT = mini$
$\Sigma\ D\ Kj$
$=0\ (i, j)$
K
где
· D это элемент виртуальной матрицы $D(i, j)$ с размером $Mx \times My$;
· $dist$ функция расстояния, например эвклидово;
· x и y сегменты ряда, с размерами Mx и My соответственно; данные размеры могут быть не равными;
· K - расстояние по виртуальной траектории от нижнего правого угла D ($D(Mx , My)$) к верхнему левому ($D(0,0)$), по такой траектории, чтобы сумма расстояния была минимальной.
Пример расчета DTW
Dynamic Time Wrapping (DTW) нелинейный алгоритм, основанный на поиске максимального сходства между точками независимо от позиции индекса. Фактически этот алгоритм в сочетании с К ближайших соседей можно рассматривать как основу для классификации всех временных рядов.
Основным преимуществом расстояния DTW является независимость (инвариантность) от сдвигов или других незначительных изменений в сегментах (например, его сжатие и растяжение). Другими словами, метод DTW пытается построить матрицу отображения (или преобразования) одного сегмента в другой и найти лучшее (расстояние между ними с минимальной стоимостью). Главный недостаток DTW - высокая сложность и неявность поиска подобия. Таким образом, в некоторых случаях DTW может показывать сходство там, где его быть не должно.
Примечание. Здесь мы будем использовать так называемую разностную dtw distance - расстояние первых произоводных временных рядов.

```
from sktime.classification.distance_based import
KNeighborsTimeSeriesClassifier
knn = KNeighborsTimeSeriesClassifier(n_neighbors=1, distance="ddtw")
knn.fit(X_train, y_train)
knn.score(X_test, y_test)
0.9866666666666667
```

Dictionary based Classifier
Подходы классификации временных рядов на основе словаря адаптируют модель набора слов, обычно используемую в обработке сигналов, компьютерном зрении и обработке звука, для классификации временных рядов.
· Скользящее окно длины w пробегается по серии.
· Для каждого окна действительно-значный ряд длины w преобразуется посредством процессов аппроксимации и дискретизации в символьную строку длины l, состоящую из α возможных букв.
· Подсчитывается появление в сегменте ряда каждого «слова» из словаря, определенного l и α,
· После завершения прохождения скользящего окна серия преобразуется в гистограмму.
· Классификация производится основана на гистограммах слов, извлеченных из ряда.
Среди всех классификаторов на основе словаря мы будем рассматривать Мешок символов SFA (BOSS). Мешок символов SFA (BOSS) BOSS - это совокупность отдельных классификаторов BOSS, использующих преобразование SFA (символьное приближение Фурье). Классификатор выполняет поиск по сетке однотипных классификаторов для разных значений параметров l, α, w и p (нормализует каждое окно). Из искомых классификаторов сохраняются только те, точность которых не менее 92% от лучшего классификатора. Отдельные классификаторы BOSS используют несимметричную функцию расстояния, расстояние BOSS, в сочетании с классификатором ближайшего соседа.
Contractable BOSS (cBOSS)

cBOSS значительно ускоряет BOSS без существенной разницы в точности за счет улучшения способа формирования ансамбля. cBOSS использует отфильтрованный выбор параметров для поиска членов своего ансамбля. Каждый член ансамбля построен на подвыборке размером 70% от выборки данных с использованием метода случайной выборки без замены. Также в данном методе введена экспоненциальная схема взвешивания прогнозов базовых классификаторов.

```
from sktime.classification.dictionary_based import ContractableBOSS,
TemporalDictionaryEnsemble
boss = ContractableBOSS(random_state=47,max_ensemble_size=100)
boss.fit(X_train, y_train)
boss.score(X_test, y_test)
0.9933333333333333
```

Классификация на основе Шейплетов (Shapelets)

Шейплеты определяются как подвыборки временного ряда (или сегмента временного ряда), которые в некотором смысле являются максимально репрезентативными для своего класса. Если предполагается задача бинарной классификации, то шейплет - это часть серии (интервал, диапазон), которая по некоторой мере представлена в большинстве сегментов одного класса и отсутствует в сериях другого класса.

ROCKET Classifer - это тип классификаторов на основе Shapelets, основанный на так называемых ROCKET преобразованиях. **Преобразования ROCKET** - это преобразования временных рядов с использованием случайных сверточных ядер (случайная длина, веса, смещение, расширение и заполнение). ROCKET Classifer вычисляет две характеристики из результирующих карт признаков (полученных после преобразований): максимальное значение и соотношение положительных значений ко всем (ppv). Преобразованные объекты используются для обучения линейного классификатора.

```
from sktime.classification.shapelet_based import ROCKETClassifier
shapelet = ROCKETClassifier(random_state=47)
shapelet.fit(X_train, y_train)
shapelet.score(X_test, y_test)
0.9933333333333333
```

Упражнение 2

1. Сравните результаты работы классификаторов из SKTime и sklearn для исследуемого выше набора данных.

Упражнение 3

1. Работа со встроенными данными.
a. Загрузите набор данных *load_italy_power_demand* из *sktime. datasets*.
b. Используйте _____split = "train", return_X_y = True для тренировочных данных и split = "test", return_X_y = True для тестовых данных.
c. Выберите тестовую часть с размером до 50 экземпляров.
d. Попробуйте визуализировать тестовые и тренировочные данные.
e. Выберите 4 классификатора SKlearn и 4 классификатора SKTIme для этого набора данных.

Упражнение 4

1. Подберите произвольный набор данных для набора данных из интернета
2. Загрузите набор данных по следующей ссылке:
https://raw.githubusercontent.com/jbrownlee/Datasets/master/IndoorMovement.zip
3. Набор данных включает сигналы (временные ряды) от людей, перемещающихся через 2 комнаты. Сигналы принимаются с 4 датчиков. Сигналы разделены на 3 пути (группы) того, как движутся люди, и два класса (цели): "+1" - когда человек проходит между двумя комнатами "-1" - когда человек не проходит между двумя комнатами.
4. Задача - по этим датчикам классифицировать переходы между комнатами.
5. Для начала загрузим данные.
The task is inspired by this tutorial: https://machinelearningmastery.com/indoormovement-time-series-classification-with-machine-learning-algorithms/

```
import os
def load_dataset(prefix='Indoor/'):
```

```
'''
load_dataset of Indoor Movements.
Parameters
----------
prefix: str
path to the directory with the dataset
Returns
----------
sequences: list,
data of 4-th sensors for each instance (human).
targets: 1d ndarray,
targets for each instance +1 or -1.
groups: 1d ndarray,
group id for each instance 1,2 or 3.
'''
grps_dir, data_dir = prefix+'groups/', prefix+'dataset/'
# load mapping files
targets = pd.read_csv(data_dir + 'MovementAAL_target.csv', header=0)
groups = pd.read_csv(grps_dir + 'MovementAAL_DatasetGroup.csv',
header=0)
paths = pd.read_csv(grps_dir + 'MovementAAL_Paths.csv', header=0)
# load traces
sequences = list()
for name in os.listdir(data_dir):
filename = os.path.join(data_dir,name)
if filename.endswith('_target.csv'): continue
df = pd.read_csv(filename, header=0)
values = df.values.reshape(-1,1)
sequences.append(values.tolist())
return sequences, targets.values[:,1], groups.values[:,1]
sequences, targets, groups =load_dataset(prefix='Indoor/')
labels, counts = np.unique(targets, return_counts=True)
print('targets',labels, counts)
labels, counts = np.unique(groups, return_counts=True)
print('targets',labels, counts)
targets [-1 1] [156 158]
targets [1 2 3] [104 106 104]
```

6. Теперь создаем наборы данных. Мы будем использовать 1 и 2 группы в качестве данных для обучения и 3 группы в качестве тестовых данных. Обратите внимание, что все данные имеют разную длину, и мы дополняем все данные до максимальной длины.

```
def create_dataset(sequences, targets, groups, max_len = None):
'''
Create train and test datasets from raw data.
The data set is created with padding all instance
to the max_len variable.
Here group 1 and 2 are appended to the train data,
and froup 3 for test data.
Parameters
----------
sequences: list,
data of 4-th sensors for each instance (human).
targets: 1d ndarray,
targets for each instance +1 or -1.
groups: 1d ndarray,
group id for each instance 1,2 or 3.
```

```
max_len: int or None,
if None: max_len is the maximum length of instance.
if int: all instance with length higher will be cut,
all instance with length smaller
will be padded at the end.
Returns
---------
x_train,x_test: 2d ndarrays,
train and test data.
y_train,y_test: 1d ndarrays,
train and test targets (labels).
'''
if max_len is None:
max_len =0
for i in range(len(sequences)):
if len(sequences[i])>max_len:
max_len = len(sequences[i])
else:
max_len = int(max_len)
labels, counts = np.unique(groups, return_counts=True)
test_size = counts[2]
train_size = counts[0]+counts[1]
x_train = np.zeros((train_size,max_len))
y_train = np.zeros(train_size)
x_test = np.zeros((test_size,max_len))
y_test = np.zeros(test_size)
cnt_test = 0
cnt_train = 0
for i in range(len(sequences)):
signal = np.squeeze(np.asarray(sequences[i]))
if (max_len-len(signal) >0):
signal = np.pad(signal,(0,max_len-len(signal)))
elif (max_len-len(signal) <0):
signal = signal[:max_len]
if groups[i]==3:
x_test[cnt_test] = signal
y_test[cnt_test] = targets[i]
cnt_test +=1
else:
x_train[cnt_train] = signal
y_train[cnt_train] = targets[i]
cnt_train +=1
return x_train,x_test,y_train,y_test
x_train,x_test,y_train,y_test = create_dataset(sequences, targets, groups,
max_len = 100)
print('train :',x_train.shape,y_train.shape,'test :',
x_test.shape,y_test.shape)
plt.plot(x_train[90])
print(y_train[90])
plt.show()
plt.plot(x_test[100])
print(y_test[100])
plt.show()
train : (210, 100) (210,) test : (104, 100) (104,)
1.0
-1.0
```

7. Задача этого упражнения - поиск лучшего классификатора для описываемой проблемы.__

Лабораторная работа 7

Работа сопровождается методическими указаниями.

Классификация и регрессия многомерных временных рядов

Классификация и регрессия многомерных временных рядов с использование специальных методов машинного обучения. Особенности представления многомерных временных рядов в sktime. Изучение метода WEASEL. Изучение методов векторной авторегрессии библиотеки statsmodels.

Импорт и даные

```
!pip install -U sktime
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
%matplotlib inline
```

Набор данных

Набор данных BasicMotions. Набор данных BasicMotions состоит из четырех классов: ходьбы, отдыха, бега и бадминтона. Данные собранны с помощью умных часов с 3D-акселерометром и 3D-гироскопом. От участников эксперимента по сбору данных требовалось записывать движение в общей сложности пять раз. При это данные отбирался каждые десятые доли секунды в течение десяти секунд.

```
from sklearn.model_selection import train_test_split
from sktime.datasets import load_basic_motions
X, y = load_basic_motions(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
(60, 6) (60,) (20, 6) (20,)
# multivariate input data
X_train.head()
# multi-class target variable
np.unique(y_train)
array(['badminton', 'running', 'standing', 'walking'], dtype=object)
```

Методы классификации в Sktime

Sktime предлагает три основных способа решения задач многомерной классификации временных рядов:

· Конкатенация столбцов временных рядов в один столбец длинных временных рядов с помощью ColumnConcatenator и применение одномерных классификаторов к объединенным данным.

· Объединение данных по столбцам с помощью ColumnEnsembleClassifier, в котором один и тот же классификатор проходит для каждого столбца временных рядов, как для одномерных данных, и их прогнозы, в конце, агрегированы.

· Специальные методы обработки многомерных временных рядов, например поиск объединяющих функций или общих представлений данных, таких как шейплеты или словари в многомерных пространствах.

Конкатенация временных рядов

Способ конкатенации - использовать метод ColumnConcatenator. Метод может быть объединен в конвейер с любым одномерным классификатором.

```
from sktime.transformations.panel.compose import ColumnConcatenator
from sktime.classification.interval_based import
TimeSeriesForestClassifier
from sklearn.pipeline import Pipeline
steps = [
("concatenate", ColumnConcatenator()),
("classify", TimeSeriesForestClassifier(n_estimators=100)),
]
clf = Pipeline(steps)
clf.fit(X_train, y_train)
```

```
clf.score(X_test, y_test)
1.0
```

Объединение данных по столбцам

Мы также можем установить один классификатор для каждого столбца временных рядов как для одномерного, а затем агрегировать их прогнозы. Для этого можо использовать класс методов ColumnEnsembleClassifier. Интерфейс класса похож на знакомый ColumnTransformer от sklearn. Однако, ColumnEnsembleClassifier позволяет использовать разные методы оценки для разных столбцов или подмножеств столбцов входных данных. Все они будут обработаны отдельно, а признаки, генерируемые каждым преобразователем, будут объединены для формирования единого выхода.

```
from sktime.classification.compose import ColumnEnsembleClassifier
from sktime.classification.interval_based import
RandomIntervalSpectralForest
from sktime.classification.dictionary_based import ContractableBOSS
from sktime.classification.shapelet_based import ROCKETClassifier
from sktime.classification.interval_based import
TimeSeriesForestClassifier
clf = ColumnEnsembleClassifier(
estimators=[
("TSF0", TimeSeriesForestClassifier(n_estimators=50), [0]),
#column 0
("cBOSS1", ContractableBOSS(), [1]), #column 1
("cBOSS2", ContractableBOSS(), [2]), #column 2
("RISF", RandomIntervalSpectralForest(n_estimators=50), [3]),
#column 3
]
)
clf.fit(X_train, y_train)
clf.score(X_test, y_test)
1.0
```

Специальные методы обработки многомерных временных рядов

WEASEL (Извлечение слов для классификации временных рядов) + MUSE (Многовариантное символьное расширение)

Здесь мы будем использовать метод WEASEL, преобразуя временные ряды в векторы признаков. Для этого будет использоваться подход скользящего окна. Полученные признки затем анализируются с помощью классификатора машинного обучения. Метод WEASEL основан на методе BOSS и его модификациях. В расширении WEASEL MUSE для многомерных данных применяется особая техника многомерного символьного расширения.

```
from sktime.classification.dictionary_based import MUSE
from sktime.classification.dictionary_based import WEASEL
muse = MUSE()
muse.fit(X_train, y_train)
muse.score(X_test, y_test)
1.0
```

Mr-SEQL

Другой специальный метод - это **Mr-SEQL**, который извлекает признаки из каждого измерения данных независимо. Для этого используя несколько символьных представлений временных рядов (SAX, SFA). Затем к извлеченным признакам применяется логистическая регрессия.

```
from sktime.classification.shapelet_based import MrSEQLClassifier
clf = MrSEQLClassifier()
clf.fit(X_train, y_train)
clf.score(X_test, y_test)
1.0
```

Упражнение 1

1. Попробуйте использовать стандартные классификаторы из sklearn для описанного набора данных. Сравните результаты.

Упражнение 2

1. Загрузите набор данных *japanese_vowels* с многомерными данными (*load_japanese_vowels*) попробуйте применить изученные классификаторы к этому набору данных и сравните результаты.

Предсказание многомерных данных с помощью векторной авторегрессии.

Набор данных.

VAR (векторная авторегрессия) - это частный случай методов на основе AR для многомерных данных (см. работу про ARMA).

Здесь мы воспользуемся набором данных эмпирического исследования инфляции, взятого на слух.

```
filepath =
'https://raw.githubusercontent.com/selva86/datasets/master/Raotbl6.csv'
df = pd.read_csv(filepath, parse_dates=['date'], index_col='date')
print(df.shape) # (123, 8)
df.tail()
(123, 8)
rgnp pgnp ulc gdfco gdf gdfim gdfcf gdfce
date
1988-07-01 4042.7 3971.9 179.6 131.5 124.9 106.2 123.5 92.8
1989-07-01 4162.9 4068.4 187.4 137.2 130.2 109.8 129.9 98.2
```

Давайте исследуем и визуализируем его.

```
# Plot
fig, axes = plt.subplots(nrows=4, ncols=2, dpi=120, figsize=(10,6))
for i, ax in enumerate(axes.flatten()):
data = df[df.columns[i]]
ax.plot(data, color='red', linewidth=1)
# Decorations
ax.set_title(df.columns[i])
ax.xaxis.set_ticks_position('none')
ax.yaxis.set_ticks_position('none')
ax.spines["top"].set_alpha(0)
ax.tick_params(labelsize=6)
plt.tight_layout();
```

Кроме того, мы можем проверить причинно-следственную связь (казуальность) между этими рядами с помощью теста причинности Грейнджера и теста коинтеграции. Наличие казуальности означает влияние одного ряда на другой (например, поведение одного ряда является причиной запаздывающего поведения другого ряда). В случае причинно-следственной связи для любого ряда вы можете предсказать его значения с учетом прошлых значений самого себя вместе с результатами для других рядов в системе. В идеале в многомерном ряду должна быть казуальность.

```
from statsmodels.tsa.stattools import grangercausalitytests
maxlag=12
test = 'ssr_chi2test'
def grangers_causation_matrix(data, variables, test='ssr_chi2test',
verbose=False):
"""
Check Granger Causality of all possible combinations of the Time
series.
The rows are the response variable, columns are predictors. The
values in the table
are the P-Values.
P-Values lesser than the significance level (0.05), i
mplies the Null Hypothesis that the coefficients
of the corresponding past values is zero, that is,
```

```
the X does not cause Y can be rejected.
Paramteres
------------
data : pandas dataframe,
containing the time series variables.
variables : list,
containing names of the time series variables.
Returns
----------
casual matrx: pandas dataframe,
matrix with p-values.
"""
df = pd.DataFrame(np.zeros((len(variables), len(variables))),
columns=variables, index=variables)
for c in df.columns:
for r in df.index:
test_result = grangercausalitytests(data[[r, c]],
maxlag=maxlag, verbose=False)
p_values = [round(test_result[i+1][0][test][1],4) for i in
range(maxlag)]
if verbose:
print(f'Y = {r}, X = {c}, P Values = {p_values}')
min_p_value = np.min(p_values)
df.loc[r, c] = int(100*min_p_value)/100 #for rounding to
second sign after dot.
df.columns = [var + '_x' for var in variables]
df.index = [var + '_y' for var in variables]
return df
grangers_causation_matrix(df, variables = df.columns)
rgnp_x pgnp_x ulc_x gdfco_x gdf_x gdfim_x gdfcf_x gdfce_x
rgnp_y 1.0 0.00 0.0 0.02 0.0 0.06 0.0 0.0
pgnp_y 0.0 1.00 0.0 0.00 0.0 0.00 0.0 0.0
ulc_y 0.0 0.00 1.0 0.00 0.0 0.00 0.0 0.0
gdfco_y 0.0 0.00 0.0 1.00 0.0 0.00 0.0 0.0
gdf_y 0.0 0.00 0.0 0.00 1.0 0.00 0.0 0.0
gdfim_y 0.0 0.00 0.0 0.00 0.0 1.00 0.0 0.0
gdfcf_y 0.0 0.00 0.0 0.00 0.0 0.00 1.0 0.0
gdfce_y 0.0 0.04 0.0 0.00 0.0 0.00 0.0 1.0
```

Итак, мы видим, что все переменные имеют причинность (p-значение меньше уровня значимости 0,05 - таким образом, мы отвергаем нулевую гипотезу об их независимости). Другими словами, все переменные (временные ряды) в системе взаимозаменяемо вызывают друг друга. Это позволяет использовать модели VAR для прогнозирования этой системы временных рядов.

Кроме того, мы протестируем нашу систему на тесте коинтеграции. Коинтеграционный тест. помогает установить наличие статистически значимой связи между двумя или более временными рядами. Когда два или более временных ряда коинтегрируются, это означает, что они имеют долгосрочную статистически значимую взаимосвязь.

```
from statsmodels.tsa.vector_ar.vecm import coint_johansen
def cointegration_test(df, alpha=0.05):
"""Perform Johanson's Cointegration Test and Report Summary"""
out = coint_johansen(df,-1,5)
d = {'0.90':0, '0.95':1, '0.99':2}
traces = out.lr1
cvts = out.cvt[:, d[str(1-alpha)]]
def adjust(val, length= 6): return str(val).ljust(length)
# Summary
```

```
print('Name :: Test Stat > C(95%) => Signif \n', '--'*20)
for col, trace, cvt in zip(df.columns, traces, cvts):
print(adjust(col), ':: ', adjust(round(trace,2), 9), ">",
adjust(cvt, 8), ' => ' , trace > cvt)
cointegration_test(df)
Name :: Test Stat > C(95%) => Signif
----------------------------------------
rgnp :: 248.0 > 143.6691 => True
pgnp :: 183.12 > 111.7797 => True
gdfcf :: 14.06 > 12.3212 => True
gdfce :: 0.45 > 4.1296 => False
```

Таким образом, все, кроме последнего ряда, могут быть коингерированными. Сделаем разделение тренировочной и тест выборок.

```
TEST_SIZE = 20
df_train, df_test = df[0:-TEST_SIZE], df[-TEST_SIZE:]
# Check size
print(df_train.shape) #
print(df_test.shape) #
(103, 8)
(20, 8)
```

Для того, чтобы сделать VAR нам также нужно проверить наши ряды на стационарность. Очевидно, что исходный ряд не стационарен. Мы можем проверить производную (или двойную производную), чтобы привести временной ряд к стационарному.

```
df_differenced = df_train.diff().diff().dropna()
df_differenced.shape
(101, 8)
from statsmodels.tsa.stattools import adfuller
def adfuller_test(series, signif=0.05, name='', verbose=False):
"""Perform ADFuller to test for Stationarity of given series and print
report"""
r = adfuller(series, autolag='AIC')
output = {'test_statistic':round(r[0], 4), 'pvalue':round(r[1], 4),
'n_lags':round(r[2], 4), 'n_obs':r[3]}
p_value = output['pvalue']
def adjust(val, length= 6): return str(val).ljust(length)
# Print Summary
print(f' Augmented Dickey-Fuller Test on "{name}"', "\n ", '-
'*47)
if p_value <= signif:
print(f" => P-Value = {p_value}. Rejecting Null Hypothesis.")
print(f" => Series is Stationary.")
else:
print(f" => P-Value = {p_value}. Weak evidence to reject the Null
Hypothesis.")
print(f" => Series is Non-Stationary.")
for name, column in df_differenced.iteritems():
adfuller_test(column, name=column.name)
print('\n')
Augmented Dickey-Fuller Test on "rgnp"
-----------------------------------------------
=> P-Value = 0.0013. Rejecting Null Hypothesis.
=> Series is Stationary.
```

Исследование предсказаний методом VAR

Попытаемся найти лучший порядок модели VAR для изучаемого временного ряда.

```
from statsmodels.tsa.api import VAR
from statsmodels.tools.eval_measures import rmse, aic
```

```
model = VAR(df_differenced)
orders_grid = range(1,11)
for order in orders_grid:
result = model.fit(maxlags=order)
print('Lag Order =', order, end = ' \t' )
print('AIC : ', (result.aic * 100)//100 , end = ' \t' )
print('BIC : ', (result.bic* 100)//100 , end = ' \t' )
print('HQIC: ', (result.hqic* 100)//100 , end = ' \n' )
Lag Order = 1 AIC : -3.0 BIC : -1.0 HQIC: -2.0
Lag Order = 10 AIC : -17.0 BIC : 1.0 HQIC: -10.0
```

Здесь мы можем выбрать лаг номер 10 из-за минимальности критериев AIC и BIC. Альтернативный метод выбора порядка (p) моделей VAR - использовать метод model.select_order (maxlags). Выбранный порядок (p) - это порядок, который дает самые низкие оценки целевых показателей.

```
x = model.select_order(maxlags=10)
x.summary()
<class 'statsmodels.iolib.table.SimpleTable'>
```

Здесь же можно выбрать лаг 10. Попробуем модель и сделам для нее прогноз.

```
model_fitted = model.fit(10)
model_fitted.summary()
Summary of Regression Results
==================================
Model: VAR
Method: OLS
Date: Thu, 06, May, 2021
Time: 13:29:06
--------------------------------------------------------------------
No. of Equations: 8.00000 BIC: 1.55401
Nobs: 91.0000 HQIC: -9.11225
Log likelihood: 357.824 FPE: 0.000406468
AIC: -16.3255 Det(Omega_mle): 2.49534e-06
--------------------------------------------------------------------
Results for equation rgnp
=============================================================================
coefficient std. error t-stat prob
-----------------------------------------------------------------------------
const 10.961108 8.108555 1.352 0.176
L1.rgnp -0.289441 0.437760 -0.661 0.508
L1.gdfco 59.807552 37.660066 1.588 0.112
L1.gdf 21.551681 47.962697 0.449 0.653
L1.gdfim 50.469749 40.318172 1.252 0.211
L10.gdfce 43.066363 28.964944 1.487 0.137
=============================================================================
Results for equation pgnp
=============================================================================
coefficient std. error t-stat prob
-----------------------------------------------------------------------------
const 0.071770 0.225331 0.319 0.750
L1.rgnp -0.007481 0.012165 -0.615 0.539
L4.gdfce 0.329685 0.727130 0.453 0.650
L5.rgn
```

Для полученной модели мы можем проверить некоторые оставшиеся паттерны в остатках с помощью теста durbin_watson. Для этого теста значение статистики durbin_watson может варьироваться от 0 до 4. Чем ближе к значению 2, тем больше вероятности, что значимой корреляции в остатках нет. Чем ближе к 0, тем больше вероятность, что имеется положительная корреляция в ряду, а чем ближе к 4, тем больше

отрицательная корреляция во временном ряду.

```
from statsmodels.stats.stattools import durbin_watson
out = durbin_watson(model_fitted.resid)
print(out)
[2.48414886 2.12539032 1.87510872 2.70538591 1.82807508 2.67882018
1.69115062 2.23853132]
```

Вероятно, модель вполне неплохо выбрана. Теперь сделаем прогноз. Для первого прогнозируемого (вне выборки) значения нам нужно взять последние значения lag_order наших тренировочных данных. Потому что это объем данных необходим для дальнейшего прогнозирования.

```
# Get the lag order
lag_order = model_fitted.k_ar
print(lag_order) #7
# Input data for forecasting
forecast_input = df_differenced.values[-lag_order:]
forecast_input.shape
10
(10, 8)
```

Теперь сделаем предсказание.

```
# Forecast
fc = model_fitted.forecast(y=forecast_input, steps=TEST_SIZE)
df_forecast = pd.DataFrame(fc, index=df.index[-TEST_SIZE:],
columns=df.columns + '_2d')
df_forecast
rgnp_2d pgnp_2d ulc_2d gdfco_2d gdf_2d gdfim_2d \
date
1984-10-01 -254.223407 0.794179 3.635249 -2.090034 -1.231496 -3.999701
1989-07-01 -3443.879553 3.708056 82.614658 -2.101355 -1.304433 -19.856987
gdfcf_2d gdfce_2d
date
1984-10-01 -3.542227 -10.426202
1989-04-01 23.904385 45.714004
1989-07-01 -35.944685 -47.065721
def invert_transformation(df_train, df_forecast, second_diff=False):
"""Revert back the differencing to get the forecast to original
scale."""
df_fc = df_forecast.copy()
columns = df_train.columns
for col in columns:
# Roll back 2nd Diff
if second_diff:
df_fc[str(col)+'_1d'] = (df_train[col].iloc[-1]-
df_train[col].iloc[-2]) + df_fc[str(col)+'_2d'].cumsum()
# Roll back 1st Diff
df_fc[str(col)+'_forecast'] = df_train[col].iloc[-1] +
df_fc[str(col)+'_1d'].cumsum()
return df_fc
df_results = invert_transformation(df_train, df_forecast,
second_diff=True)
df_results.loc[:, ['rgnp_forecast', 'pgnp_forecast', 'ulc_forecast',
'gdfco_forecast',
'gdf_forecast', 'gdfim_forecast', 'gdfcf_forecast',
'gdfce_forecast']]
rgnp_forecast pgnp_forecast ulc_forecast gdfco_forecast \
date
1984-10-01 3288.976593 3630.694179 166.635249 109.409966
```

```
1989-07-01 5185.807885 4220.488049 45.600926 42.089575
gdf_forecast gdfim_forecast gdfcf_forecast gdfce_forecast
date
1984-10-01 108.368504 92.700299 104.157773 91.273798
1989-04-01 70.483337 2.245113 127.659314 -67.387692
1989-07-01 66.253314 -6.853442 108.796272 -96.531441
fig, axes = plt.subplots(nrows=len(df.columns), ncols=1, dpi=150,
figsize=(10,20))
for i, (col,ax) in enumerate(zip(df.columns, axes)):
df_results[col+'_forecast'].plot(legend=True,
ax=ax).autoscale(axis='x',tight=True)
df_test[col][-TEST_SIZE:].plot(legend=True, ax=ax);
ax.set_title(col + ": Forecast vs Actuals")
plt.tight_layout();
```

В дополнение мы можем оценить результаты по некоторым метрикам

```
def forecast_accuracy(forecast, actual):
mape = np.mean(np.abs(forecast - actual)/np.abs(actual)) # MAPE
# me = np.mean(forecast - actual) # ME
# mae = np.mean(np.abs(forecast - actual)) # MAE
# mpe = np.mean((forecast - actual)/actual) # MPE
rmse = np.mean((forecast - actual)**2)**.5 # RMSE
corr = np.corrcoef(forecast, actual)[0,1] # corr
mins = np.amin(np.hstack([forecast[:,None],
actual[:,None]]), axis=1)
maxs = np.amax(np.hstack([forecast[:,None],
actual[:,None]]), axis=1)
minmax = 1 - np.mean(mins/maxs) # minmax
return({'mape':mape,
# 'me':me,
# 'mae': mae,
# 'mpe': mpe,
'rmse':rmse, 'corr':corr, 'minmax':minmax})
print('Forecast Accuracy of: rgnp')
accuracy_prod = forecast_accuracy(df_results['rgnp_forecast'].values,
df_test['rgnp'])
for k, v in accuracy_prod.items():
print(k, '\t : ', round(v,4), end='\t')
print('\nForecast Accuracy of: pgnp')
accuracy_prod = forecast_accuracy(df_results['pgnp_forecast'].values,
df_test['pgnp'])
for k, v in accuracy_prod.items():
print(k, '\t : ', round(v,4), end='\t')
print('\nForecast Accuracy of: ulc')
accuracy_prod = forecast_accuracy(df_results['ulc_forecast'].values,
df_test['ulc'])
for k, v in accuracy_prod.items():
print(k, '\t : ', round(v,4), end='\t')
print('\nForecast Accuracy of: gdfco')
accuracy_prod = forecast_accuracy(df_results['gdfco_forecast'].values,
df_test['gdfco'])
for k, v in accuracy_prod.items():
print(k, '\t : ', round(v,4), end='\t')
print('\nForecast Accuracy of: gdf')
accuracy_prod = forecast_accuracy(df_results['gdf_forecast'].values,
df_test['gdf'])
```

```
for k, v in accuracy_prod.items():
print(k, '\t : ', round(v,4), end='\t')
print('\nForecast Accuracy of: gdfim')
accuracy_prod = forecast_accuracy(df_results['gdfim_forecast'].values,
df_test['gdfim'])
for k, v in accuracy_prod.items():
print(k, '\t : ', round(v,4), end='\t')
print('\nForecast Accuracy of: gdfcf')
accuracy_prod = forecast_accuracy(df_results['gdfcf_forecast'].values,
df_test['gdfcf'])
for k, v in accuracy_prod.items():
print(k, '\t : ', round(v,4), end='\t')
print('\nForecast Accuracy of: gdfce')
accuracy_prod = forecast_accuracy(df_results['gdfce_forecast'].values,
df_test['gdfce'])
for k, v in accuracy_prod.items():
print(k, '\t : ', round(v,4), end='\t')
Forecast Accuracy of: rgnp
mape : 0.1735 rmse : 973.9519 corr : 0.8749 minmax :
0.1336
Forecast Accuracy of: pgnp
mape : 0.0121 rmse : 64.8452 corr : 0.9964 minmax : 0.0118
Forecast Accuracy of: ulc
mape : 0.3694 rmse : 84.6745 corr : -0.9424 minmax : 0.3693
Forecast Accuracy of: gdfco
mape : 0.286 rmse : 46.5543 corr : -0.9492 minmax : 0.286
Forecast Accuracy of: gdf
mape : 0.194 rmse : 30.9204 corr : -0.9757 minmax : 0.194
Forecast Accuracy of: gdfim
mape : 0.4762 rmse : 62.3369 corr : -0.8948 minmax : 0.4762
Forecast Accuracy of: gdfcf
mape : 0.0462 rmse : 7.9659 corr : 0.5428 minmax : 0.0461
Forecast Accuracy of: gdfce
mape : 0.8072 rmse : 96.5182 corr : 0.1955 minmax : 0.8072
```

Упражнение 3

1. Как можно заметить, сложно найти лучшую модель для всего временного ряда, о собенно на относительно большом горизонте. Задача: изменить порядок VAR и н айти наилучший для тестового набора данных – для всех составляющих многоме рного ряда.
2. Попробуйте построить модель ARMA для отдельных составляющих временного ряда, где точность оказалась наименьшей.__

Лабораторная работа 8

Работа сопровождается методическими указаниями.

Анализ временных рядов с использованием глубокого обучения нейронных сетей

Использование методов глубокого обучения в анализе временных рядов. Исслед ование одномерной сверточной нейронной сети в задаче классификации времен ных рядов. Исследование одномерной сверточной нейронной сети в задаче регре ссии временных рядов.

Импорт библиотек и данных

```
!pip install -U sktime
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import GlobalAvgPool1D
from keras.layers import Dropout
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
from keras.utils import to_categorical
from sktime.datasets import load_italy_power_demand
from sktime.utils.data_processing import from_nested_to_2d_array
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
%matplotlib inline
```

Здесь мы будем использовать набор данных о потреблении электроэнергии в Италии.

```
xdf, ydf = load_italy_power_demand( return_X_y=True)
print(xdf.shape,ydf.shape)
(1096, 1) (1096,)
```

Давайте преобразуем набор в традиционную форму.

```
x = from_nested_to_2d_array(xdf)
x = x.values
print(x.shape)
y = ydf.values
print(y.shape)
(1096, 24)
(1096,)
labels, count = np.unique(y, return_counts=True)
print('labels',lables, 'count', count)
y = [0 if yi=='2' else 1 for yi in y]
labels ['1' '2'] count [547 549]
y =np.array(y)
labels, count = np.unique(y, return_counts=True)
print('labels',labels, 'count', count)
labels [0 1] count [549 547]
```

И визуализируем его

```
instance = 9
plt.plot(x[instance,:])
print(y[instance]);
plt.show()
instance = 99
plt.plot(x[instance,:])
print(y[instance]);
0
1
```

Теперь мы можем сделать разделение на тренировочную и тестовую выборки и

преобразовать наборы данных, в обычную форму для keras.

```
TEST_SIZE = int(x.shape[0]*0.5)
x_train,x_test = x[:TEST_SIZE,:,np.newaxis],x[-TEST_SIZE:,:,np.newaxis]
y_train,y_test = y[:TEST_SIZE],y[-TEST_SIZE:]
y_train.shape,y_test.shape,x_train.shape,x_test.shape
((548,), (548,), (548, 24, 1), (548, 24, 1))
```

Here we will build a base-line model for classification

```
epochs = 100
batch_size = 32
n_timesteps = x_train.shape[1]
n_features = 1
n_outputs = labels.shape[0]
# # scale data
# x_train, x_test = scale_data(x_train, x_test, param)
model = Sequential()
#24
model.add(Conv1D(filters=64, kernel_size=3, activation='relu', input_shape
=(n_timesteps,n_features)))
#22x64
model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
#20x64*64
model.add(Dropout(0.5))
model.add(MaxPooling1D(pool_size=2))
#10x64*64
model.add(GlobalAvgPool1D())
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accu
racy'])
model.summary()
# # fit network
Model: "sequential_23"
_________________________________________________________________
Layer (type) Output Shape Param #
=================================================================
conv1d_46 (Conv1D) (None, 22, 64) 256
_________________________________________________________________
conv1d_47 (Conv1D) (None, 20, 64) 12352
_________________________________________________________________
dropout_23 (Dropout) (None, 20, 64) 0
_________________________________________________________________
max_pooling1d_23 (MaxPooling (None, 10, 64) 0
_________________________________________________________________
global_average_pooling1d_21 (None, 64) 0
_________________________________________________________________
dense_26 (Dense) (None, 1) 65
=================================================================
Total params: 12,673
Trainable params: 12,673
Non-trainable params: 0
_________________________________________________________________
```

Теперь попытаемся провести обучение

```
model.fit(x_train,
y_train,
epochs=epochs,
batch_size=batch_size,
verbose=1)
```

```
Epoch 1/100
Epoch 3/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6740 -
accuracy:
0.6051
…
Epoch 31/100
18/18 [============================..] - ETA: 0s - loss: 0.4392 - accuracy:
0.87
```
Теперь попробуем оценить результаты
```
# evaluate model
_, accuracy = model.evaluate(x_test, y_test, batch_size=batch_size,
verbose=1)
18/18 [==============================] - 0s 940us/step - loss: 0.1940 -
accuracy
: 0.9416
```
Упражнение 1

1. Постарайтесь повысить точность модели CNN, используя свой опыт в архитектурах нейронных сетей.
2. Сравните полученные результаты CNN с полученными для классических методов классификации для того же набора данных (см. работу №6).
3. Выберите один из наборов данных для классификации, изученных в предыдущих работах, и попробуйте построить 1D-CNN классификатор для него.

Упражнение 2

1. Загрузите набор данных для VAR регрессии из работы №7 и попробуйте построить для него модель регрессии с использованием CNN (каждый столбец данных можно рассматривать как отдельный канал входных данных).

Список вопросов для теста

1. Выберете верное определение тренда временного ряда:
    a. **Часть любого ряда с почти монотонным (или локально монотонным) поведение и высокой интенсивностью.**
    b. Часть временного ряда со сравнительно высокой частотой повторений значений.
    c. Стохастическая часть ряда, которая может быть, как стационарной, так и не стационарной.
2. Выберете неверное определение тренда временного ряда:
    a. Часть любого ряда с почти монотонным (или локально монотонным) поведение и высокой интенсивностью.
    b. Стохастическая нестационарная или детерминированная часть любого ряда с почти монотонным поведение.
    c. **Стохастическая часть ряда, которая может быть, как стационарной, так и не стационарной.**
3. Выберете верное определение тренда временного ряда:
    a. **Стохастическая нестационарная или детерминированная часть любого ряда с почти монотонным поведение**.
    b. Часть временного ряда со сравнительно высокой частотой повторений значений.
    c. Стохастическая часть ряда, которая может быть, как стационарной, так и не стационарной.
4. Выберете верное определение тренда временного ряда:
    a. **Часть любого ряда с почти монотонным (или локально монотонным) поведение и высокой интенсивностью.**
    b. Часть временного ряда со сравнительно высокой частотой повторений значений.
    c. Часть временного ряда, включающая сравнительно редкие, но регулярные события.
5. Выберете верное определение тренда временного ряда:
    a. **Часть любого ряда с почти монотонным (или локально монотонным) поведение и высокой интенсивностью.**
    b. Стохастическая часть ряда, которая может быть, как стационарной, так и не стационарной.
    c. Часть временного ряда, включающая сравнительно редкие, но регулярные события.

6. Выберете неверное утверждение о модели временного ряда:
    a. **Редкие, но регулярные события должны быть рассмотрены как циклическая часть ряда.**
    b. Редкие и нерегулярные события могут быть исключены или обработаны как аномальные явления.
    c. Цикличность может быть включена в тренд.
7. Выберете неверное утверждение о модели временного ряда:
    a. **Редкие, но регулярные события могут быть исключены или обработаны как аномальные явления.**
    b. Редкие и нерегулярные события могут быть исключены или обработаны как аномальные явления.
    c. Цикличность может быть включена в тренд.
8. Выберете верное утверждение о модели временного ряда:
    a. **Редкие, но регулярные события должны быть рассмотрены сезонная часть ряда**.
    b. Редкие и нерегулярные события могут быть исключены или обработаны как аномальные явления.
    c. Цикличность может быть включена в сезонность.
9. Выберете верное утверждение о модели временного ряда:
    a. Цикличность может быть включена в сезонность.
    b. Редкие и нерегулярные события могут быть исключены или обработаны как аномальные явления.

c. **Цикличность может быть включена в тренд**.

10. Выберете верное утверждение о модели временного ряда:
    a. Редкие, но регулярные события могут быть рассмотрены как часть тренда.
    b. Стохастические события с независимыми составляющими могут быть рассмотрены как часть тренда.
    c. Цикличность может рассмотрена как часть тренда.

11. Выберете верное выражение для процесса случайного блуждания, для временного ряда где ε_n - шумы:
    a. $y(t)=c/(1+exp(-k(t-m)))$.
    b. $y_n=y_{(n-1)}+a+\varepsilon_n$.
    c. $y(t)=a\cdot t+b\cdot t^2+c$.
12. Выберете верное выражение для процесса случайного блуждания, для временного ряда где ε_n - шумы:
    a. $y(t)=c/(1+exp(-k(t-m)))$.
    b. $y_n=y_{(n-1)}+a\cdot y_{(n-2)}+\varepsilon_n$.
    c. $y(t)=a\cdot t+b+\varepsilon_n$.
13. Выберете верное выражение для процесса случайного блуждания, для временного ряда где ε_n - шумы:
    a. $y(t)=\sum_i a_i f(t_i)$.
    b. $y_n= y_{(n-1)}+\sum_i a_i\varepsilon_{(n-i)}$.
    c. $y(t)=\sum_i a_i t^i$.
14. Выберете верное выражение для процесса случайного блуждания, для временного ряда где ε_n - шумы:
    a. $y(t)=a\cdot t+b+\varepsilon_n$.
    b. $y_n=y_{(n-1)}+a+\varepsilon_n$.
    c. $y(t)=a\cdot t+b$.
15. Выберете верное выражение для процесса случайного блуждания, для временного ряда где ε_n - шумы:
    a. $y_n=y_{(n-1)}+a\cdot t$.
    b. $y_n=y_{(n-1)}+a\cdot t+\varepsilon_n$.
    c. $y_n=y_{(n-1)}+a$.

16. Выберете верное определение нестационарного временного ряда:
    a. **Временной ряд, в котором последующие одна за другой части различаются**.
    b. Временной ряд, в котором среднее и дисперсия постоянны для любого сегмента ряда.
    c. Временной ряд, в котором каждая часть одинаковая, не зависимо от того, когда она выбрана.
17. Выберете неверное определение нестационарного временного ряда:
    a. Временной ряд, в котором последующие одна за другой части различаются.
    b. Временной ряд, в котором среднее и дисперсия различаются для любого сегмента ряда.
    c. Временной ряд, в котором каждая часть одинаковая, не зависимо от того, когда она выбрана.
18. Выберете неверное определение стационарного временного ряда:
    a. **Временной ряд, в котором последующие одна за другой части могут иметь разную дисперсию**.
    b. Временной ряд, в котором среднее и дисперсия постоянны для любого сегмента ряда.
    c. Временной ряд, в котором каждая часть одинаковая, не зависимо от того, когда она выбрана.
19. Выберете неверное определение нестационарного временного ряда:

a. Временной ряд, в котором дисперсия постоянна для любого сегмента ряда, а среднее значение различается.
b. Временной ряд, в котором среднее постоянно для любого сегмента ряда, а дисперсия различается.
c. **Временной ряд, в котором каждая часть одинаковая, не зависимо от того, когда она выбрана**.

20. Выберете верное определение нестационарного временного ряда:
    a. **Временной ряд, в котором дисперсия постоянна для любого сегмента ряда, а среднее значение различается**.
    b. Временной ряд, в котором среднее и дисперсия постоянны для любого сегмента ряда.
    c. Временной ряд, в котором каждая часть одинаковая, не зависимо от того, когда она выбрана.

21. Выберете верное выражение для автокорреляционной функции:
    a. $\frac{1}{N}\frac{\sum_{i=0}^{N-1}(y_k-ev(g_k))(g_{i-k}-ev(y_k))}{var(y)}$.
    b. $\frac{1}{N}\sum_{i=0}^{N-1}(y_i-ev(y))^2$.
    c. $\frac{1}{N}\frac{\sum_{i=0}^{N-1}(y_k-ev(y))(y_{i-k}-ev(y)))}{var(y)}$.
22. Выберете верное выражение для автокорреляционной функции:
    a. $\frac{1}{N}\frac{\sum_{i=0}^{N-1}(y_k-ev(y_k))(g_{i-k}-ev(y_k))}{\sqrt{var(y)var(g)}}$.
    b. $\frac{1}{N}\sum_{i=0}^{N-1}(y_i-ev(y))(g_i-ev(g))$.
    c. $\frac{1}{N}\frac{\sum_{i=0}^{N-1}(y_k-ev(y))(y_{i-k}-ev(y))}{var(y)}$.
23. Выберете верное выражение для автокорреляционной функции:
    a. $\frac{1}{N}\sum_{i=0}^{N-1}y_i$.
    b. $\frac{1}{N}\frac{\sum_{i=0}^{N-1}(y_i-ev)(y_i-ev)}{var(y)}$.
    c. $\frac{1}{N}\frac{\sum_{i=0}^{N-1}(y_k-ev)(y_{i-k}-ev)}{var(y)}$.
24. Выберете верное выражение для автокорреляционной функции:
    a. $\frac{1}{N}\sum_{i=0}^{N-1}(y_i-ev(y))$.
    b. $\frac{1}{N}\sum_{i=0}^{N-1}(y_i-ev(y))^2$.
    c. $\frac{1}{N}\frac{\sum_{i=0}^{N-1}(y_k-ev(y))(y_{i-k}-ev(y))}{var(y)}$.
25. Выберете верное выражение для автокорреляционной функции:
    a. $\frac{1}{N}\sum_{i=0}^{N-1}(y_k-ev(y))(y_k-ev(y))$.
    b. $\frac{1}{N}\frac{\sum_{i=0}^{N-1}(y_k-var(y))(y_{i-k}-var(y))}{ev(y)}$.
    c. $\frac{1}{N}\frac{\sum_{i=0}^{N-1}(y_k-ev)(y_{i-k}-ev)}{var(y)}$.

26. Выберите верное выражение для метрики SMAE:
    a. $\frac{1}{N}\sum_{i=1}^{n}(y_i-\widehat{y_i})^2$.
    b. $\frac{1}{N}\sum_{n=0}^{N}\frac{|\widehat{y_n}-y_n|}{|y_n|+|\widehat{y_n}|}$.
    c. $\frac{1}{N}\sum_{n=0}^{N}|\widehat{y_n}-y_n|$.
27. Выберите верное выражение для метрики SMAE:
    a. $\frac{1}{N}\frac{\sum_{i=1}^{N}(y_i-\widehat{y_i})^2}{\sum_{i=1}^{N}(y_i-ev(y))^2}$.

b. $\frac{1}{N}\sum_{n=0}^{N} \frac{|\widehat{y_n}-y_n|}{|y_n|+|\widehat{y_n}|}$.

c. $\frac{1}{N}\sum_{n=0}^{N} |\widehat{y_n}-y_n|$.

28. Выберите верное выражение для метрики SMAE:
    a. $\frac{1}{N}\sum_{n=0}^{N} \frac{|\widehat{y_n}-y_n|}{|y_n|}$.
    b. $\frac{1}{N}\sum_{n=0}^{N} \frac{|\widehat{y_n}-y_n|}{|y_n|+|\widehat{y_n}|}$.
    c. $\frac{1}{N}\frac{\sum_{i=1}^{n} (y_i-\widehat{y_i})^2}{\sum_{i=1}^{n} (y_i-ev(y))^2}$.
29. Выберите верное выражение для метрики SMAE:
    a. $\frac{1}{N}\sum_{n=0}^{N} \frac{|\widehat{y_n}+y_n|}{|y_n|-|\widehat{y_n}|}$.
    b. $\frac{1}{N}\sum_{n=0}^{N} \frac{|\widehat{y_n}-y_n|}{|y_n|+|\widehat{y_n}|}$.
    c. $\frac{1}{N}\sum_{n=0}^{N} |\widehat{y_n}-y_n|$.
30. Выберите верное выражение для метрики SMAE:
    a. $\frac{1}{N}\sum_{n=0}^{N} \frac{|\widehat{y_n}+y_n|}{|\widehat{y_n}|+|y_n|}$.
    b. $\frac{1}{N}\sum_{n=0}^{N} \frac{|\widehat{y_n}-y_n|}{|y_n|+|\widehat{y_n}|}$.
    c. $\frac{1}{N}\sum_{n=0}^{N} |\widehat{y_n}-y_n|$.

31. Выберите верное выражение для экспоненциального среднего:
    a. $\hat{y}_n = \alpha y_n + (1-\alpha)\hat{y}_{n-1}$.
    b. $y_m = \frac{1}{m}\sum_{i=n-m}^{n} w_i y_m$
    c. $\hat{y}_n = \sum_{i=1}^{p} \alpha_i y_{n-i}$.
32. Выберите верное выражение для экспоненциального среднего:
    a. $\hat{y}_n = \alpha y_n + (1-\alpha)\hat{y}_{n-1}$.
    b. $\hat{y}_n = \frac{1}{m}\sum_{i=n-m}^{n} y_i$
    c. $\hat{y}_n = \sum_{i=n-m}^{n} y_i - \frac{1}{m}\sum_{i=n-m}^{n} y_i$.
33. Выберите верное выражение для экспоненциального среднего:
    a. $\hat{y}_n = \alpha y_n + (1-\alpha)\hat{y}_{n-1}$.
    b. $\hat{y}_n = \alpha\hat{y}_n + (1-\alpha)\frac{1}{m}\sum_{i=n-m}^{n} y_i$
    c. $\hat{y}_n = \sum_{i=n-m}^{n} y_i - \frac{1}{m}\sum_{i=n-m}^{n} y_i$.
34. Выберите верное выражение для экспоненциального среднего:
    a. $\hat{y}_n = \alpha y_n + (1-\alpha)\hat{y}_{n-1}$.
    b. $\hat{y}_n = \alpha\hat{y}_n + (1-\alpha)\frac{1}{n}\sum_{i=1}^{n} y_i$
    c. $\hat{y}_n = \alpha y_n + (1-\alpha)\frac{1}{n}\sum_{i=1}^{n} \hat{y}_i$.
35. Выберите верное выражение для экспоненциального среднего:
    a. $\hat{y}_n = \alpha y_n + (1-\alpha)\hat{y}_{n-1}$.
    b. $\hat{y}_n = \alpha\hat{y}_n + (1-\alpha)\hat{y}_{n-1}$
    c. $\hat{y}_n = \frac{1}{n}\sum_{i=1}^{n} \alpha_i\hat{y}_{n-i}$.

36. Выберите верное выражение для ARMA процесса:
    a. $\hat{y}_n = \sum_{i=0}^{p} w_i x_i$.
    b. $\hat{y}_n = \sum_{i=1}^{p} \alpha_i y_{n-i} + \sum_{i=0}^{q} \beta_i \varepsilon_{n-i}$.
    c. $\hat{y}_n = \frac{1}{m}\sum_{i=n-m}^{n} w_i y_i$
37. Выберите верное выражение для ARMA процесса:
    a. $\hat{y}_n = \sum_{i=1}^{p} \alpha_i y_{n-i} + \sum_{i=0}^{q} \beta_i y_{n-i}$.

b. $\hat{y}_n = \sum_{i=1}^{p} \alpha_i y_{n-i} + \sum_{i=0}^{q} \beta_i \varepsilon_{n-i}$.
c. $\hat{y}_n = \frac{1}{m} \sum_{i=n-m}^{n} w_i y_i$

38. Выберите верное выражение для ARMA процесса:
    a. $\hat{y}_n = \sum_{i=1}^{p} \alpha_i y_{n-i} + \varepsilon_n$.
    b. $\hat{y}_n = \sum_{i=1}^{p} \alpha_i y_{n-i} + \sum_{i=0}^{q} \beta_i \varepsilon_{n-i}$.
    c. $\hat{y}_n = y_n + \sum_{i=0}^{q} \beta_i \varepsilon_{n-i}$
39. Выберите верное выражение для ARMA процесса:
    a. $\hat{y}_n = \sum_{i=0}^{p} w_i y_i + \hat{y}_{n-1}$.
    b. $\hat{y}_n = \sum_{i=1}^{p} \alpha_i y_{n-i} + \sum_{i=0}^{q} \beta_i \varepsilon_{n-i}$.
    c. $\hat{y}_n = y_n + \sum_{i=0}^{q} \beta_i \varepsilon_{n-i}$
40. Выберите верное выражение для ARMA процесса:
    a. $\hat{y}_n = \sum_{i=0}^{q} \beta_i \varepsilon_{n-i} + \hat{y}_{n-1}$.
    b. $\hat{y}_n = \sum_{i=1}^{p} \alpha_i y_{n-i} + \sum_{i=0}^{q} \beta_i \varepsilon_{n-i}$.
    c. $\hat{y}_n = \sum_{i=1}^{p} \alpha_i y_{n-i} + \varepsilon_n$.

41. Выберите верное утверждение о причине предпочтения модели ARIMA по сравнению с моделью ARMA:
    a. Выбор ARIMA в случае слишком высокого порядка AR или MA в ARMA.
    b. Выбор ARIMA в случае слишком зашумленных данных
    c. **Выбор ARIMA в случае нестационарного временного ряда**.
42. Выберите верное утверждение о причине предпочтения модели ARIMA по сравнению с моделью ARMA:
    a. Выбор ARIMA в случае слишком высокого порядка AR или MA в ARMA.
    b. Выбор ARIMA в случае слишком высокого влияния сезонности
    c. **Выбор ARIMA в случае нестационарного временного ряда**.
43. Выберите неверное утверждение о причине предпочтения модели ARIMA по сравнению с моделью ARMA:
    a. **Выбор ARIMA в случае слишком высокого порядка AR или MA в ARMA**.
    b. Выбор ARIMA в случае слишком высокого влияния тренда во временном ряду
    c. Выбор ARIMA в случае нестационарного временного ряда.
44. Выберите неверное утверждение о причине предпочтения модели ARIMA по сравнению с моделью ARMA:
    a. Выбор ARIMA в случае слишком зашумленных данных
    b. Выбор ARIMA в случае слишком высокого влияния тренда во временном ряду
    c. Выбор ARIMA в случае нестационарного временного ряда.
45. Выберите верное утверждение о причине предпочтения модели ARIMA по сравнению с моделью ARMA:
    a. Выбор ARIMA в случае слишком высокого порядка AR или MA в ARMA.
    b. Выбор ARIMA в случае стационарного временного ряда.
    c. Выбор ARIMA в случае, когда ряд напоминает модель случайного блуждания.

46. Выберите верное утверждение о причине предпочтения модели SARIMA по сравнению с моделями ARMA и ARIMA:
    a. **Выбор SARIMA в случае высокого влияния сезонности или нестационарной сезонности**.
    b. Выбор SARIMA в случае, когда ряд напоминает модель случайного блуждания.
    c. Выбор SARIMA в случае не стационарного поведения тренда.
47. Выберите неверное утверждение о причине предпочтения модели SARIMA по сравнению с моделями ARMA и ARIMA:
    a. Выбор SARIMA в случае высокого влияния сезонности.
    b. Выбор SARIMA в случае нестационарной сезонности.

c. **Выбор SARIMA в случае, когда ряд напоминает модель случайного блуждания**.

48. Выберите верное утверждение о причине предпочтения модели SARIMA по сравнению с моделями ARMA и ARIMA:
    a. **Выбор SARIMA в случае высокое влияние сезонности или нестационарное сезонное поведение**.
    b. Выбор SARIMA в случае, когда порядки модели ARIMA слишком большие.
    c. Выбор SARIMA в случае не стационарного поведения тренда.
49. Выберите неверное утверждение о причине предпочтения модели SARIMA по сравнению с моделями ARMA и ARIMA:
    a. Выбор SARIMA в случае высокого влияния сезонности.
    b. Выбор SARIMA в случае нестационарной сезонности.
    c. **Выбор SARIMA в случае, когда порядки модели ARIMA слишком большие**.
50. Выберите верное утверждение о причине предпочтения модели SARIMA по сравнению с моделями ARMA и ARIMA:
    a. **Выбор SARIMA в случае высокое влияние сезонности или нестационарное сезонное поведение**.
    b. Выбор SARIMA в случае, когда ряд напоминает модель случайного блуждания.
    c. Выбор SARIMA в случае, когда порядки модели ARIMA слишком большие.

51. Выберите неверное утверждение касательно разведывательного анализа данных:
    a. Разведывательный анализ данных позволяет получить начальные предположения об особенностях поведения временного ряда.
    b. Разведывательный анализ данных позволяет отбросить выбросы, пропуски и другие особенности аномального поведения временного ряда.
    c. **Разведывательный анализ данных необходим с целью выделения, отбора и преобразования признаков временного ряда для решения поставленной задачи**.
52. Выберите неверное утверждение касательно разведывательного анализа данных:
    a. Разведывательный анализ данных позволяет отбросить выбросы, пропуски и другие особенности аномального поведения временного ряда.
    b. Разведывательный анализ данных позволяет оценить разброс, стационарность, и другие общие особенности временного ряда.
    c. **Разведывательный анализ данных необходим с целью выделения, преобразования признаков временного ряда для решения поставленной задачи**.
53. Выберите верное утверждение касательно разведывательного анализа данных:
    a. Разведывательный анализ данных позволяет получить начальные предположения об особенностях поведения временного ряда.
    b. Разведывательный анализ данных позволяет провести операции сжатия размерности/кластеризации или другие операции без учителя в рамках поставленной задачи.
    c. Разведывательный анализ данных необходим с целью выделения, отбора и преобразования признаков временного ряда для решения поставленной задачи.
54. Выберите неверное утверждение касательно разведывательного анализа данных:
    a. Разведывательный анализ данных позволяет получить начальные предположения об особенностях поведения временного ряда.
    b. Разведывательный анализ данных позволяет провести отбор и преобразовать составляющие временного ряда.
    c. **Разведывательный анализ данных позволяет провести операции сжатия размерности/кластеризации или другие операции без учителя в рамках поставленной задачи**.
55. Выберите верное утверждение касательно разведывательного анализа данных:
    a. Разведывательный анализ данных позволяет получить начальные предположения об особенностях поведения временного ряда.

b. Разведывательный анализ данных позволяет получить точности классификации/регрессии или другого метода обучения с учителем в рамках поставленной задачи.
c. Разведывательный анализ данных необходим с целью выделения, отбора и преобразования признаков временного ряда для решения поставленной задачи.

56. Выберите неверное утверждение касательно преобразования признаков временного ряда:
    a. Преобразования признаков в рамках поставленной задачи должно быть проведено в рамках разведывательного анализа временного ряда.
    b. Выбор признаков может быть, как с учителем, так и без учителя.
    c. Выделение признаков – это задача представления данных в виде, пригодном для их последующей обработки каким-либо методом.
57. Выберите неверное утверждение касательно преобразования признаков временного ряда:
    a. **Выделение и преобразование признаков выполняются только с учителем**.
    b. Выбор признаков может быть, как с учителем, так и без учителя.
    c. Выделение признаков – это задача представления данных в виде, пригодном для их последующей обработки каким-либо методом.
58. Выберите верное утверждение касательно преобразования признаков временного ряда:
    a. Преобразования признаков в рамках поставленной задачи должно быть проведено в рамках разведывательного анализа временного ряда.
    b. Выделение и преобразование признаков выполняются только с учителем.
    c. Выбор признаков может быть, как с учителем, так и без учителя.
59. Выберите верное утверждение касательно преобразования признаков временного ряда:
    a. Преобразования признаков в рамках поставленной задачи должно быть проведено в рамках разведывательного анализа временного ряда.
    b. Выбор признаков проводится только в задачах с учителем.
    c. Выделение признаков – это задача представления данных в виде, пригодном для их последующей обработки каким-либо методом.
60. Выберите верное утверждение касательно преобразования признаков временного ряда:
    a. Преобразование признаков проводится только в задачах без учителя.
    b. **Выбор признаков может быть, как с учителем, так и без учителя**.
    c. Выбор признаков проводится только в задачах с учителем.

61. Выберите верный тип функций расстояния для кластеризации временного ряда (или его сегментов) в случае, когда у вас нет требований по единообразию временного поведения сегментов.
    a. Расстояние Эвклида.
    b. Расстояние косинусов.
    c. **Расстояние с динамическим сжатием по времени (DTW)**.
62. Выберите верный тип функций расстояния для кластеризации временного ряда (или его сегментов) в случае, когда у вас нет требований по единообразию временного поведения сегментов.
    a. Расстояние Эвклида.
    b. Расстояние Миньковского.
    c. Расстояние с динамическим сжатием по времени (DTW).
63. Выберите верный тип функций расстояния для кластеризации временного ряда (или его сегментов) в случае, когда у вас нет требований по единообразию временного поведения сегментов.
    a. Расстояние Кварталов (Манхэттенское расстояние).
    b. Расстояние косинусов.
    c. Расстояние с динамическим сжатием по времени (DTW).

64. Выберите верный тип функций расстояния для кластеризации временного ряда (или его сегментов) в случае, когда у вас нет требований по единообразию временного поведения сегментов.
    a. Расстояние Миньковского.
    b. Расстояние косинусов.
    c. Расстояние с динамическим сжатием по времени (DTW).
65. Выберите верный тип функций расстояния для кластеризации временного ряда (или его сегментов) в случае, когда у вас нет требований по единообразию временного поведения сегментов.
    a. Расстояние Кварталов (Манхэттенское расстояние).
    b. Расстояние Миньковского.
    c. Расстояние с динамическим сжатием по времени (DTW).

66. Выберите верное утверждение касательно определения аномального поведения:
    a. Использование изоляционного леса – это задача с учителем;
    b. Использование автокодирующей сети — это задача полу-контролируемого обучения;
    c. Использование одноклассового метода опорных векторов — это задача обучения без учителя.
67. Выберите неверное утверждение касательно определения аномального поведения:
    a. Использование изоляционного леса – полу-контролируемого обучения;
    b. Использование автокодирующей сети — это задача полу-контролируемого обучения;
    c. Использование одноклассового метода опорных векторов — это задача полу-контролируемого обучения.
68. Выберите верное утверждение касательно определения аномального поведения:
    a. Использование изоляционного леса – это задача с учителем;
    b. Использование метода DBSCAN — это задача полу-контролируемого обучения;
    c. Использование одноклассового метода опорных векторов — это задача полу-контролируемого обучения.
69. Выберите неверное утверждение касательно определения аномального поведения:
    a. **Использование метода BOXPLOT – это задача с учителем**;
    b. Использование автокодирующей сети — это задача полу-контролируемого обучения;
    c. Использование одноклассового метода опорных векторов — это задача полу-контролируемого обучения.
70. Выберите верное утверждение касательно определения аномального поведения:
    a. **Использование метода BOXPLOT – это задача без учителя**;
    b. Использование изоляционного леса — это задача обучения с учителем;
    c. Использование одноклассового метода опорных векторов — это задача обучения без учителя.

71. Выберите неверное утверждение касательно классификации временных рядов
    a. Шейплет – это часть временного ряда, которая в наибольшей степени характеризует его класс.
    b. Ансамблевые методы классификации (как RISE и TSF) – это комбинация определённых точечных признаков и метода случайного леса.
    c. Метод HIVE-COTE как правило уступает по точности таким методам, как классификация на основе словарей (BOSS).
72. Выберите верное утверждение касательно классификации временных рядов
    a. Шейплет – это часть временного ряда, которая в наибольшей степени характеризует его класс.

b. Метод классификации TSF – это метод случайного леса по результатам разбиения временного ряда на сегменты.
c. Метод HIVE-COTE как правило уступает по точности таким методам, как классификация на основе словарей (BOSS).

73. Выберите верное утверждение касательно классификации временных рядов
    a. Метод классификации TSF – это метод случайного леса по результатам разбиения временного ряда на сегменты.
    b. Метод классификации RISE – это метод случайного леса по результатам разбиения временного ряда на сегменты.
    c. Метод HIVE-COTE как правило превосходит по точности такие методам, как классификация на основе словарей (BOSS).
74. Выберите неверное утверждение касательно классификации временных рядов
    a. Шейплет – это часть временного ряда, которая в наибольшей степени характеризует его класс.
    b. **Метод классификации TSF – это метод случайного леса по результатам разбиения временного ряда на сегменты**.
    c. Метод HIVE-COTE как правило, как правило превосходит по точности такие методам, как классификация на основе словарей (BOSS).
75. Выберите верное утверждение касательно классификации временных рядов
    a. Шейплет – это комбинация определённых точечных признаков и метода k ближайших соседей.
    b. **Ансамблевые методы классификации (как RISE и TSF) – это комбинация определённых точечных признаков и метода случайного леса**.
    c. Метод HIVE-COTE как правило, как правило превосходит по точности такие методам, как классификация на основе словарей (BOSS).

76. Выберите неверное утверждение касательно классификации временных рядов
    a. Шейплет – это комбинация определённых точечных признаков и метода k ближайших соседей.
    b. Rocket –это метод классификации временного ряда с использованием случайных сверточных ядер.
    c. Метод HIVE-COTE это ансамблевый метод, включающий, такие методы, как шейплеты, временные леса и словари.
77. Выберите верное утверждение касательно классификации временных рядов
    a. Расстояние DTW как правило используется как признак в ансамбле деревьев RISE.
    b. **Rocket –это метод классификации временного ряда с использованием случайных сверточных ядер**.
    c. Метод BOSS подразумевает построение деревьев на результатах преобразования сегментов методом словарей.
78. Выберите неверное утверждение касательно классификации временных рядов
    a. Расстояние DTW как правило используется как признак в методе к ближайшим соседям.
    b. Rocket – это метод классификации временного ряда с использованием случайных сверточных ядер.
    c. Метод BOSS подразумевает построение деревьев на результатах преобразования сегментов методом словарей.
79. Выберите верное утверждение касательно классификации временных рядов
    a. Шейплет – это комбинация определённых точечных признаков и метода k ближайших соседей.
    b. **Rocket –это метод классификации временного ряда с использованием случайных сверточных ядер**.
    c. Расстояние DTW как правило используется как признак в ансамбле деревьев RISE.
80. Выберите неверное утверждение касательно классификации временных рядов

a. **Расстояние DTW как правило используется как признак в методе RISE.**
b. Rocket –это метод классификации временного ряда с использованием случайных сверточных ядер.
c. Метод HIVE-COTE это ансамблевый метод, включающий, такие методы, как шейплеты, временные леса и словари.

81. Выберите неверное утверждение касательно предсказания значений временных рядов:
    a. **Классические методы машинного обучения как правило дают наибольшую точность.**
    b. Метод SARIMXA (в т.ч. ARIMA) как правило плохо обрабатывают большие объемы данных.
    c. Непараметрические методы (например, Holt-Winter) позволяют достигнуть лучших показателей в случае одноперменных данных небольшого размера.
82. Выберите верное утверждение касательно предсказания значений временных рядов:
    a. Классические методы машинного обучения как правило дают наибольшую точность, но имеют высокую временную сложность.
    b. **Метод SARIMXA (в т.ч. ARIMA) как правило рекомендуется в случае небольших объемов выборки данных.**
    c. Непараметрические методы (например, Holt-Winter) позволяют достигнуть лучших показателей в случае нестационарных данных.
83. Выберите неверное утверждение касательно предсказания значений временных рядов:
    a. Методы машинного обучения как правило рекомендуется использовать в случае больших выборок многопеременных временных рядов.
    b. **Метод SARIMXA (в т.ч. ARIMA) как правило хорошо обрабатывают большие объемы выборки данных.**
    c. Не параметрические методы (например, Holt-Winter) позволяют достигнуть лучших показателей в случае одноперменных данных небольшого размера.
84. Выберите верное утверждение касательно предсказания значений временных рядов:
    a. **Методы машинного обучения как правило рекомендуется использовать в случае больших выборок многопеременных временных рядов.**
    b. Метод SARIMXA (в т.ч. ARIMA) как правило плохо обрабатывают большие объемы данных.
    c. Непараметрические методы (например, Holt-Winter) позволяют достигнуть лучших показателей в случае нестационарных временных рядов.
85. Выберите неверное утверждение касательно предсказания значений временных рядов:
    a. Классические методы машинного обучения как правило дают наибольшую точность для небольших выборок данных.
    b. Метод SARIMXA как правило рекомендуют в случае многопеременных временных рядов.
    c. **Непараметрические методы (например, Holt-Winter) позволяют достигнуть лучших показателей в случае одноперменных данных небольшого размера**.

86. Выберите неверное утверждение касательно использования методов глубокого обучения в анализе временных рядов:
    a. Одномерная расширенная свертка – это популярное решение так как обеспечивает сравнительно низкую вероятность переобучения при высокой величине рецептивного поля.
    b. Рекуррентные сети часто не позволяют достигать высоких результатов в силу высокой сложности их тренировки.
    c. **Методы нелинейной авторегрессии (NAR, NARX) показывают наилучшие результаты в задаче предсказания.**

87. Выберите верное утверждение касательно использования методов глубокого обучения в анализе временных рядов:
    a. **Одномерная расширенная свертка – это популярное решение так как обеспечивает сравнительно низкую вероятность переобучения при высокой величине рецептивного поля.**
    b. Рекуррентные сети - это популярное решение так как обеспечивает сравнительно низкую вероятность переобучения.
    c. Методы нелинейной авторегрессии (NAR, NARX) - это популярное решение так как обеспечивает сравнительно низкую вероятность переобучения.
88. Выберите верное утверждение касательно использования методов глубокого обучения в анализе временных рядов:
    a. Методы нелинейной авторегрессии (NAR, NARX) - это популярное решение так как обеспечивает сравнительно низкую вероятность переобучения.
    b. **Рекуррентные сети часто не позволяют достигать высоких результатов в силу высокой сложности их тренировки**.
    c. Одномерная свертка имеет ряд недостатков, которые не позволяют использовать ее в случае достаточно больших выборок временных рядов.
89. Выберите неверное утверждение касательно использования методов глубокого обучения в анализе временных рядов:
    a. **Одномерная свертка – это не популярное решение так как не позволяет учитывать долговременный контекст временного ряда.**
    b. Рекуррентные сети - это не популярное решение так как имеют высокую вероятность переобучения.
    c. Методы нелинейной авторегрессии (NAR, NARX) - это не популярное решение так как обеспечивает сравнительно низкую вероятность переобучения.
90. Выберите неверное утверждение касательно использования методов глубокого обучения в анализе временных рядов:
    a. Одномерная расширенная свертка – это популярное решение так как обеспечивает сравнительно низкую вероятность переобучения при высокой величине рецептивного поля.
    b. Рекуррентные сети часто не позволяют достигать высоких результатов в силу высокой сложности их тренировки.
    c. **Методы нелинейной авторегрессии (NAR, NARX) - это популярное решение так как обеспечивает сравнительно низкую вероятность переобучения.**

91. Выберите неверное утверждение касательно использования методов глубокого обучения в анализе временных рядов:
    a. Глубокие нейронные сети рекомендуется использовать в случае плохоформализуемых временных рядов с большим размером выборки.
    b. Глубокие нейронные сети рекомендуется использовать в случае больших выборок временных рядов, в том числе многопеременных временных рядов.
    c. **Глубокие нейронные сети рекомендуется использовать в случае предсказания значений нестационарных выборок однопеременных временных рядов с любым размером выборки**.
92. Выберите неверное утверждение касательно использования методов глубокого обучения в анализе временных рядов:
    a. Глубокие нейронные сети рекомендуется использовать в случае плохоформализуемых временных рядов с любым размером выборки.
    b. Глубокие нейронные сети рекомендуется использовать в случае предсказания значений нестационарных выборок многопеременных временных рядов с большим размером выборки.

c. **Глубокие нейронные сети рекомендуется в случае предсказания значений нестацоинарных выборок одноперемменных временных рядов с любым размером выборки**.

93. Выберите верное утверждение касательно использования методов глубокого обучения в анализе временных рядов:
    a. **Глубокие нейронные сети рекомендуется использовать в случае плохоформализуемых временных рядов.**
    b. Глубокие нейронные сети рекомендуется использовать в случае небольших выборок многопеременных временных рядов.
    c. Глубокие нейронные сети рекомендуется использовать в случае небольших выборок нестационарных одноперемменных временных рядов.
94. Выберите верное утверждение касательно использования методов глубокого обучения в анализе временных рядов:
    a. Глубокие нейронные сети рекомендуется использовать в случае стацоинарных временных рядов.
    b. Глубокие нейронные сети рекомендуется использовать в случае нестационарных временных рядов.
    c. **Глубокие нейронные сети рекомендуется использовать в случае плохоформализуемых временных рядов**.
95. Выберите неверное утверждение касательно использования методов глубокого обучения в анализе временных рядов:
    a. Глубокие нейронные сети рекомендуется использовать в случае плохоформализуемых временных рядов.
    b. Глубокие нейронные сети рекомендуется использовать в случае очень больших выборок временных рядов.
    c. **Глубокие нейронные сети рекомендуется использовать в случае стационарных временных рядов**.

96. Выберите верное утверждение касательно особенностей архитектур нейронных сетей анализа временных рядов:
    a. **Трансформеры, как правило, это архитектуры нейронных сетей со слоями внимания и полносвязными слоями**.
    b. Трансформеры, как правило, это рекуррентные архитектуры нейронных сетей с LSTM ячейками и слоями внимания.
    c. Трансформеры, как правило, это сверточные архитектуры нейронных сетей со слоями внимания.
97. Выберите верное утверждение касательно особенностей архитектур нейронных сетей анализа временных рядов:
    a. **Трансформеры, как правило, это архитектуры нейронных сетей со слоями внимания и полносвязными слоями**.
    b. Трансформеры, как правило, это нейронные сети с комбинацией слоев внимания, само-внимания и рекуррентных ячеек.
    c. Трансформеры, как правило, это нейронные сети с комбинацией слоев внимания, расширенной свертки и рекуррентных ячеек.
98. Выберите верное утверждение касательно особенностей архитектур нейронных сетей анализа временных рядов:
    a. **Трансформеры, как правило, это архитектуры нейронных сетей со слоями внимания и полносвязными слоями**.
    b. Трансформеры, как правило, это рекуррентные архитектуры нейронных сетей слоями внимания.
    c. Трансформеры, как правило, это сверточные архитектуры нейронных сетей со слоями нелинейной авторегрессии.

99. Выберите верное утверждение касательно особенностей архитектур нейронных сетей анализа временных рядов:
    a. **Трансформеры, как правило, это архитектуры нейронных сетей со слоями внимания и полносвязными слоями**.
    b. Трансформеры, как правило, это архитектуры нейронных сетей со сверточными слоями и рекуррентными слоями.
    c. Трансформеры, как правило, это сверточные архитектуры нейронных сетей со слоями внимания
100. Выберите верное утверждение касательно особенностей архитектур нейронных сетей анализа временных рядов:
    a. **Трансформеры, как правило, это архитектуры нейронных сетей со слоями внимания и полносвязными слоями**.
    b. Трансформеры, как правило, это нейронные сети с комбинацией слоев внимания, расширенной свертки и рекуррентных ячеек.
    c. Трансформеры, как правило, это сверточные архитектуры нейронных сетей со слоями нелинейной авторегрессии.