Дата последней сборки документа:
11 июня 2023 г. в 01:44

Заметки по Анализу временных рядов и сопряженным вопросам

Подвойский А.О.

Здесь приводятся заметки по некоторым вопросам, касающимся машинного обучения, анализа данных, программирования на языках `Python`, `R` и прочим сопряженным вопросам так или иначе, затрагивающим работу с временными рядами.

Краткое содержание

Содержание

1. Приемы работы с библиотекой ETNA

1.1. Полезные рерсурсы

Домашняя страница проекта https://github.com/tinkoff-ai/etna.

1.2. Установка

Установить библиотеку можно как обычно с помощью менеджера пакетов `pip`

```
pip install etna
```

1.3. Сжатая сводка по библиотеке, рекомендации

- Перекрестную проверку с расширяющимся окном (или на скользящем окне) в библиотеке ETNA можно выполнить с помощью метода `backtest()`.
- Размер тестовой выборки, как правило, определяется *горизонтом прогнозирования* h, а тот в свою очередь определяется бизнес-требованиями. Если, скажем, интересует прогноз на 14 дней вперед, то и тестовая выборка должна включать 14 более поздних наблюдений.
- Размер тестовой выборки остается постоянным. Это значит, что метрики качества, полученные в результате вычислений прогнозов каждой обученной модели по тестовому набору, будут последовательны и их можно объединять и сравнивать.
- Размер обучающей выборки не может быть меньше тестовой выборки.
- Если данные содержат сезонность, обучающая выборка должна содержать не менее двух полных сезонных циклов (правило $2L$, где L – количество периодов в полном сезонном цикле, необходимое для инициализации параметров некоторых моделей, например, для вычисления исходного значения тренда в модели тройного экспоненциального сглаживания), учитывая уменьшение длины ряда при выполнении процедур обычного и сезонного дифференциирования.
- Ширина окна w для скользящих статистик (и лаговых признаков) должна быть неменьше горизонта прогнозирования, $w \geqslant h$ (видимо, для того чтобы поймать паттерн соответствующего горизонта, например, недельный). И порядок лага lag_ord должен быть неменьше горизонта прогнозирования $lag\_ord \geqslant h$. Так как в противном случае признаки тестового поднабора данных, построенные на лагах с порядком меньшим горизонта прогнозирования, будут использовать значения целевой переменной из тестового поднабора данных (а это утечка).
- Скользящее среднее используется не только для конструирования признаков, но и в качестве прогнозной модели[1] (когда прогноз – это скользящее среднее n последних наблюдений), а также для сглаживания выбросов, краткосрочных колебаний и более четкого выделения долгосрочных тенденций в ряде.

1.4. Примеры использования

1.4.1. Начало

Кадр данных, представляющих временной ряд должен содержать следующие столбцы:

- `target`: столбец, который нужно предсказывать,
- `timestamp`: столбец с временными метками,
- `segment`: имя сегмента, так как в общем случае ETNA ориентируется на многомерные временные ряды. В случае одномерного ряда получается такой вот атрибут-артефакт.

```
import pandas as pd

original_df = pd.read_csv("data/monthly-australian-wine-sales.csv")
# month -> timestamp, sales -> target
original_df["timestamp"] = pd.to_datetime(original_df["month"])
original_df["target"] = original_df["sales"]
original_df.drop(columns=["month", "sales"], inplace=True)
original_df["segment"] = "main"
```

[1] Как и фильтр Калмана, и преобразование Фурье

Библиотека ETNA работает со специальной структурой данных `TSDataset`, поэтому сначала классический `DataFrame` преобразовать в `TSDataset`

```
from etna.datasets.tsdataset import TSDataset

original_df: pd.DataFrame
df: pd.DataFrame = TSDdataset.to_dataset(original_df)
```

А вот теперь можно построить `TSDataset`

```
ts: TSDataset = TSDataset(df, freq="MS")
```

Можно посмотреть базовую инфомрацию

```
ts.info()
"""
<class 'etna.datasets.TSDataset'>
num_segments: 1
num_exogs: 0
num_regressors: 0
num_known_future: 0
freq: MS
start_timestamp end_timestamp  length  num_missing
segments
main           1980-01-01    1994-08-01     176            0
"""
```

Или в формате кадра данных

```
ts.describe()
```

Построим прогноз с помощью простой модели `NaivModel`

```
train_ts, test_ts = ts.train_test_split(
    train_start="1980-01-01",
    train_end="1993-12-01",
    test_start="1994-01-01",
    test_end="1994-08-01",
)
```

```
from etna.models import NaiveModel
HORIZON = 8

# Fit the model
model = NaiveModel(lag=12)
model.fit(train_ts)

# Make the forecast
future_ts = train_ts.make_future(
    future_steps=HORIZON,
    tail_steps=model.context_size
)
forecast_ts = model.forecast(
    future_ts,
    prediction_size=HORIZON
)
```

Оценим качество прогноза

```
from etna.metrics import SMAPE

smape = SMAPE()
```

```
smape(y_true=test_ts, y_pred=forecast_ts)  # {'main': 11.492045838249387}
```

Теперь построим прогноз с помощью Catboost

```
from etna.transforms import LagTransform, LogTransform

lags = LagTransform(
    in_column="target",
    lags=list(rante(8, 24, 1))
)
log = LogTransform(in_column="target")
transforms = [log, lags]
# Преобразования применяются к обчающему фрагменту ряда на месте
train_ts.fit_transform(transforms)
```

```
from etna.models import CatBoostMultiSegmentModel

model = CatBoostMultiSegmentModel()
model.fit(train_ts)
future_ts = train_ts.make_future(future_steps=HORIZON, transforms=transforms)
forecast_ts = model.forecast(future_ts)
forecast_ts.inverse_transform(transforms)
```

```
from etna.metrics import SMAPE

smape = SMAPE()
smape(y_true=test_ts, y_pred=forecast_ts)  # {'main': 10.657026308972483}
```

Все шаги можно собрать в конвейер

```
from etna.pipeline import Pipeline

train_ts, test_ts = ts.train_test_split(
  train_start="2019-01-01",
  train_end="2019-10-31",
  test_start="2019-11-01",
  test_end="2019-11-30",
)

model = Pipeline(
  model=CatBoostMultiSegmentModel(),
  transforms=transforms,
  horizon=HORIZON,
)
model.fit(train_ts)
forecast_ts = model.forecast()

smape = SMAPE()
smape(y_true=test_ts, y_pred=forecast_ts)
```

1.5. Обратное тестирование. Backtest

```
import pandas as pd
import matplotlib.pyplot as plt

from etna.datasets.tsdataset import TSDataset
from etna.metrics import MAE
from etna.metrics import MSE
from etna.metrics import SMAPE
```

```
from etna.pipeline import Pipeline
from etna.models import ProphetModel
from etna.analysis import plot_backtest
```

Пример обратного тестирования на 3-х фолдах (рис. 1).

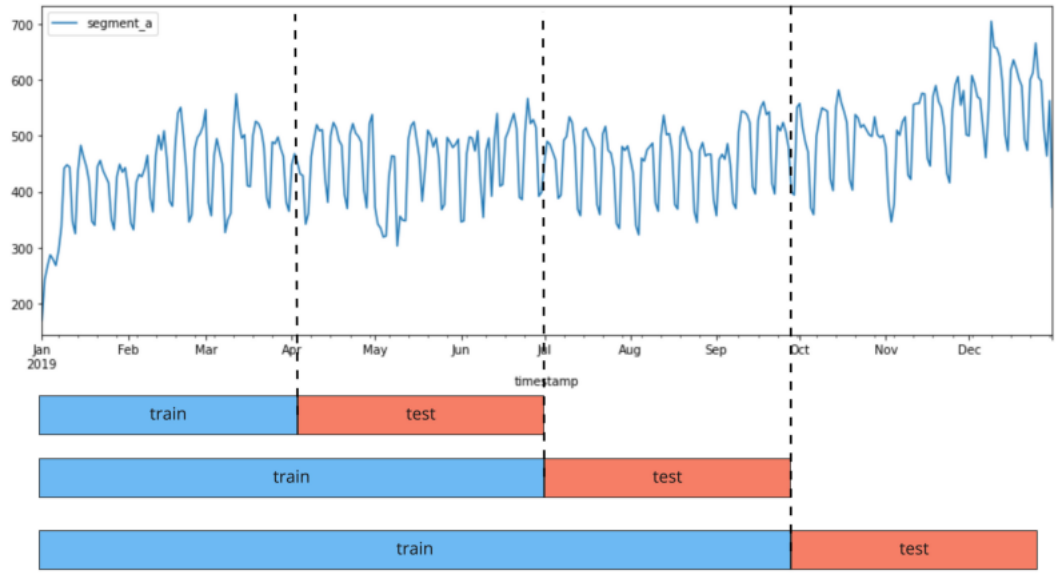

Рис. 1. Обратное тестирование на 3-х фолдах

```
df = pd.read_csv("./data/example_dataset.csv")
df = TSDataset.to_dataset(df)
ts = TSDataset(df, freq="D")
```

Создадим цепочку преобразований

```
horizon = 31  # Set the horizon for predictions
model = ProphetModel()  # Create a model
transforms = []  # A list of transforms -  we will not use any of them

pipeline = Pipeline(model=model, transforms=transforms, horizon=horizon)
```

Метод `backtest()` возвращает три кадра данных:

- кадр данных с метриками для каждого фолда и каждого сегмента,
- кадр данных с прогнозами,
- кадр данных с информацией по каждому фолду.

```
metrics_df, forecast_df, fold_info_df = pipeline.backtest(
    ts=ts,
    metrics=[
        MAE(),
        MSE(),
        SMAPE(),
    ]
)
```

Можно получить метрики, усредненные по фолдам

```
metrics_df, forecast_df, fold_info_df = pipeline.backtest(
    ts=ts,
    metrics=[
```

```
        MAE(),
        MSE(),
        SMAPE()
    ],
    aggregate_metrics=True,
)
```

Обратное тестирование с масками для фолдов. Рассмотрим 3 стратегии: `SlidingWindowSplitter`, `ExpandingWindowSplitter` и `SingleWindowSplitter` (из `sktime`).

Чтобы использовать стратегию расширяющегося окна `ExpandingWindowSplitter`, достаточно просто использовать `mode="expand"`

```
metrics_df, _, _ = pipeline.backtest(
    ts=ts,
    metrics=[
        MAE(),
        MSE(),
        SMAPE()
    ],
    n_folds=3,
    mode="expand"
)
```

Для того чтобы использовать стратегию `SlidingWindowSplitter`

```
from etna.pipeline import FoldMask
import numpy as np

# 1 Without mask
metrics_df, _, _ = pipeline.backtest(
    ts=ts,
    metrics=[
        MAE(),
        MSE(),
        SMAPE()
    ],
    n_folds=1
)
```

Или с маской

```
# 2 With specific mask
window_size = 85
first_train_timestamp = ts.index.min() + np.timedelta64(100, "D")
last_train_timestamp = first_train_timestamp + np.timedelta64(window_size, "D")
target_timestamps = pd.date_range(start=last_train_timestamp + np.timedelta64(1, "D"), periods=
    horizon)
mask = FoldMask(
  first_train_timestamp=first_train_timestamp,
  last_train_timestamp=last_train_timestamp,
  target_timestamps=target_timestamps,
)

metrics_df, _, _ = pipeline.backtest(ts=ts, metrics=[MAE(), MSE(), SMAPE()], n_folds=[mask])
```

Чтобы использовать стратегию скользящего окна `SlidingWindowSplitter`, нужно создать список масок для фолдов `FoldMask`

```
n_folds = 3
```

```
def sliding_window_masks(window_size, n_folds):
  masks = []
  for n in range(n_folds):
    first_train_timestamp = ts.index.min() + np.timedelta64(100, "D") + np.timedelta64(n, "D")
    last_train_timestamp = first_train_timestamp + np.timedelta64(window_size, "D")
    target_timestamps = pd.date_range(start=last_train_timestamp + np.timedelta64(1, "D"),
    periods=horizon)
    mask = FoldMask(
      first_train_timestamp=first_train_timestamp,
      last_train_timestamp=last_train_timestamp,
      target_timestamps=target_timestamps,
    )
    masks.append(mask)
    return masks
```

```
masks = sliding_window_masks(window_size=window_size, n_folds=n_folds)
metrics_df, _, _ = pipeline.backtest(ts=ts, metrics=[MAE(), MSE(), SMAPE()], n_folds=masks)
```

Список иллюстраций

Список литературы

1. *Лутц М.* Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 1280 с.
2. *Жерон О.* Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow: концепции, инструменты и техники ля создания интеллектуальных систем. – СПб.: ООО «Альфа-книга», 2018. – 688 с.
3. *Бурков А.* Машинное обучение без лишних слов. – СПб.: Питер, 2020. – 192 с.
4. *Бурков А.* Инженерия машинного обучения. – М.:ДМК Пресс, 2022. – 306 с.
5. *Лакшманан В.* Машинное обучение. Паттерны проектирования. – СПб.: БХВ-Перетбург, 2022. – 448 с.
6. *Бизли Д.* Python. Подробный справочник. – Пер. с англ. – СПб.: Символ-Плюс, 2010. – 864 с.
7. *Rashmi K.V., Gilad-Bachrach R.* DART: Dropouts meet Multiple Additive Regression Trees, 2015
8. *Ke G. etc.* LightGBM: A Highly Efficient Gradient Boosting Decision Tree, 2017