ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

Visualization - Merlion Library

GVHD: Lê Hồng Trang
SV thực hiện: Nguyễn Văn Hoàn – 1711376
Đoàn Quang Chính – 1710685

Tp. Hồ Chí Minh, Tháng 10/2021

Mục lục

1 Giới thiệu data visualization

Đối với các mô hình học máy, thuật toán và dữ liệu là hai yếu tố quan trọng nhất, trong đó, mô hình sẽ cho kết quả tốt hay không tốt phụ thuộc rất nhiều vào dữ liệu mà ta cung cấp cho nó. Vậy nên, để có đưa ra một mô hình, kết quả tốt, việc đầu tiên, ta cần hiểu rõ về dữ liệu.
Có 2 hướng để quan sát và hiểu dữ liệu:

- Quan sát raw data.
- Quan sát dự vào visualization.

Quan sát raw data, là chúng ta tìm hiểu về kích thước, thuộc tính,kiểu và các giá trị của dữ liệu. Quan sát dựa vào visualization, chúng ta dựa vào các biểu đồ, hình vẽ để nhìn thấy những mối quan hệ, xu hướng và những giá trị giữa các dữ liệu.

Trong bài viết này, chúng ta sẽ đi tìm hiểu sâu hơn về time series data visualization sử dụng thư viện Merlion.

2 Time series data visualization sử dụng Prophet

Để sử dụng thư viện prophet, cách tốt nhất là sử dụng conda với các câu lệnh sau:

- conda install -c conda-forge fbprophet
- conda install -c conda-forge/label/cf201901 fbprophet

Với dữ liệu đầu vào là bảng gồm 2 cột ds và y, biểu diễn cho thời gian và y là biểu diễn cho giá trị nhật kí thăm page của người dùng.

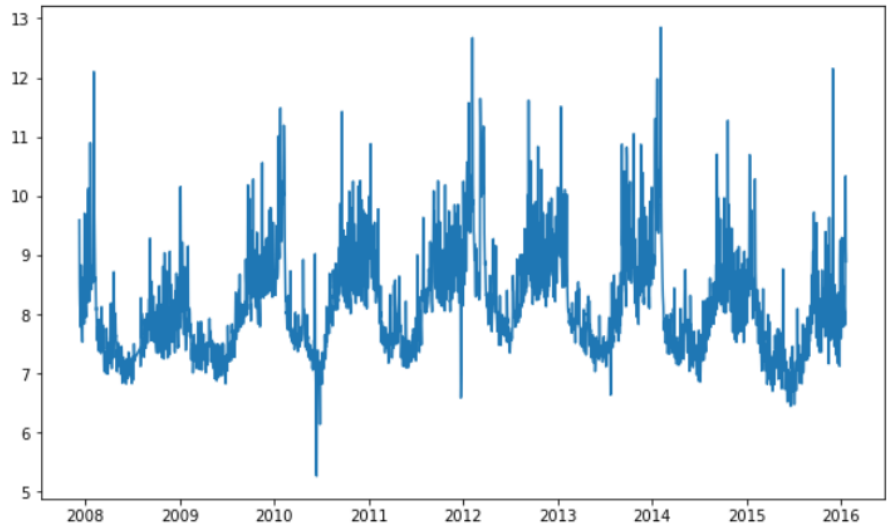
```
In [2]: url = "https://raw.githubusercontent.com/facebook/prophet/master/examples/example_wp_log_peyton_manning.csv"
        df = pd.read_csv(url)
        df.head()
```

Out[2]:

	ds	y
0	2007-12-10	9.590761
1	2007-12-11	8.519590
2	2007-12-12	8.183677
3	2007-12-13	8.072467
4	2007-12-14	7.893572

Bằng cách visualize sử dụng matplotlib, ta nhận được kết quả như hình dưới đây.

Out[2]: [<matplotlib.lines.Line2D at 0x1fb12b6b670>]



Thực hiện train model được cho bởi prophet và forecast.

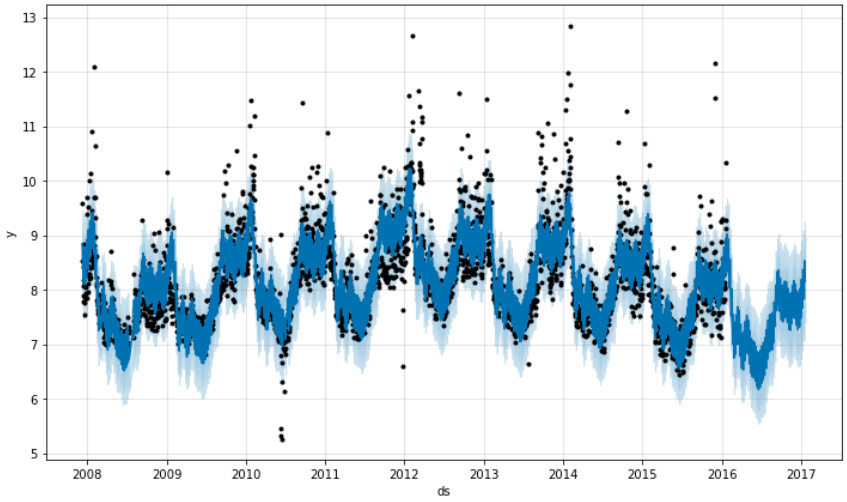
In [16]:
```
m = Prophet(daily_seasonality=True)
m.fit(df)

future = m.make_future_dataframe(periods=365)
future.tail()

forecast = m.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

Out[16]:

	ds	yhat	yhat_lower	yhat_upper
3265	2017-01-15	8.207233	7.487619	8.928623
3266	2017-01-16	8.532254	7.762911	9.249594
3267	2017-01-17	8.319668	7.621351	9.105755
3268	2017-01-18	8.152270	7.385049	8.907525
3269	2017-01-19	8.164205	7.424061	8.955579

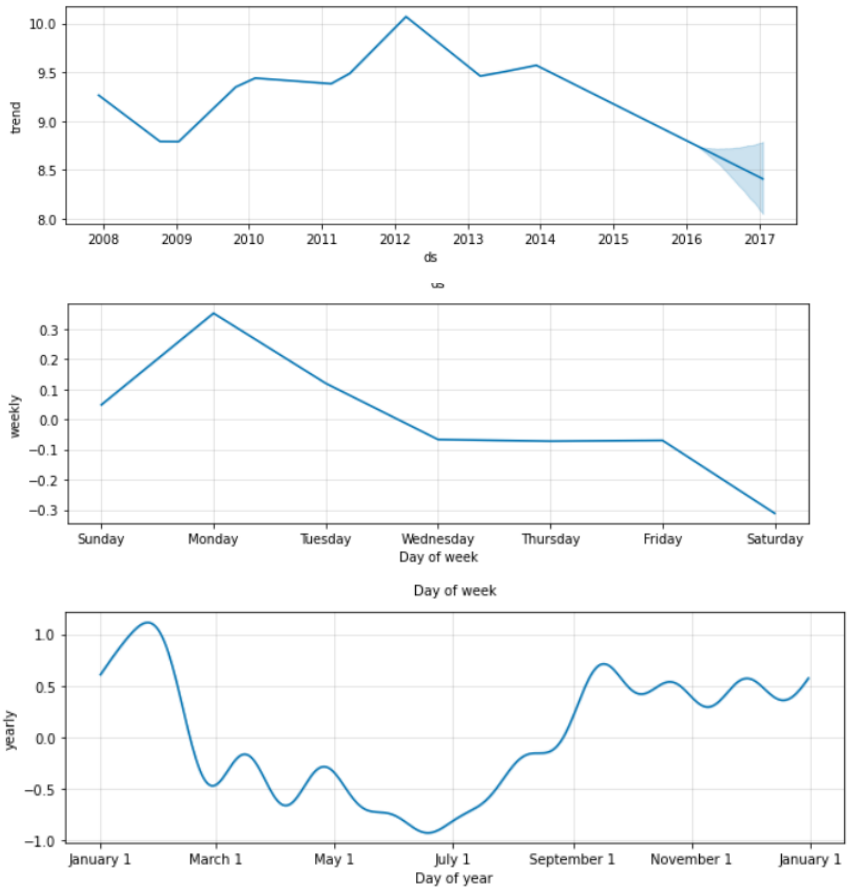
Sử dùng method plot để visualize kết quả forecast, ta được:

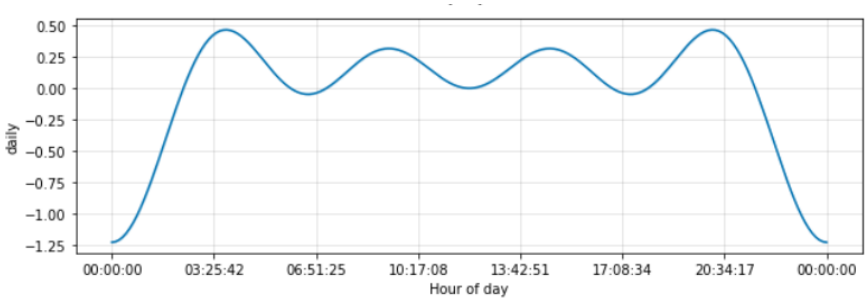
```
fig1 = m.plot(forecast)
```



Sử dụng method plot_components ta nhận được các kết quả sau:

```
plt2 = m.plot_components(forecast)
```





Nhận xét: so sánh khi vẽ bằng matplotlib và prophet, ta thấy prophet cung cấp những điểm sau đây:

- prophet vẽ thêm những vùng xanh nhạt hiển thị những khoảng dự đoán, giúp ta hình dung rõ hơn về kết quả.
- prophet vẽ những điểm giá trị, từ đó giúp ta dễ nhận ra những giá trị khác biệt (anomaly detection).
- prophet cung cấp method để vẽ chi tiết các component giúp ta nhận thấy rõ hơn xu hướng trong các khoảng thời gian khác nhau.

3 Thư viện Merlion

-Merilion là thư viện mã nguồn mở cho các mô hình học máy chuỗi thời gian cụ thể là ở hai bài toán chính là anomaly detection và forecasting trên cả đơn biến và đa biến. Nó gồm các module giúp thuận tiện cho việc visualizing dữ liệu cho người dùng.

3.1 Install Merlion

Cài đặt merlion rất đơn giản: chỉ cần gitclone về và install theo câu lệnh như hình bên dưới.

```
[ ] !git clone  https://github.com/salesforce/Merlion.git
```

```
[ ] %cd Merlion
    #cố định version của commit để đảm bảo code chạy ổn định
    !git checkout  0a7c0b465507d084cb9320d0be5688e000846d16
    %cd ..
```

```
[ ] !pip install Merlion/
    %pip install Merlion/ts_datasets/
    %pip install numpy
    %pip install pandas
```

Trong bài tập lớn nhóm có sử dụng tập data mẫu của merlion nên ta sẽ cài đặt luôn ts_dataset của Merlion.
Đồng thời để code chạy ổn định nên nhóm sẽ sử dụng cố định source code thư viện Merlion ở version commit như hình trên.

3.2 Loading data

Merlion tương tác với dữ liệu thông qua 2 format chính là TimeSeries và UnivariateTimeSeries.

3.2.1 UnivariateTimeSeries

UnivariateTimeSeries dùng để load kiểu dữ liệu đơn biến. Gồm 2 parameter là timestamp và value. Nó hỗ trợ chuyển đổi từ pandas dataframe sang datetime với timestamp đơn vị là mili.

	timestamp_millis	kpi	kpi_label
0	1583140320000	667.118	0
1	1583140380000	611.751	0
2	1583140440000	599.456	0
3	1583140500000	621.446	0
4	1583140560000	1418.234	0
...	...	...	...
86802	1588376760000	874.214	0
86803	1588376820000	937.929	0
86804	1588376880000	1031.279	0
86805	1588376940000	1099.698	0
86806	1588377000000	935.405	0

[86807 rows x 3 columns]

timestamp_millis	kpi	kpi_label
2020-03-02 09:12:00	667.118	0
2020-03-02 09:13:00	611.751	0
2020-03-02 09:14:00	599.456	0
2020-03-02 09:15:00	621.446	0
2020-03-02 09:16:00	1418.234	0
...	...	...
2020-05-01 23:46:00	874.214	0
2020-05-01 23:47:00	937.929	0
2020-05-01 23:48:00	1031.279	0
2020-05-01 23:49:00	1099.698	0
2020-05-01 23:50:00	935.405	0

[86807 rows x 2 columns]

3.2.2 TimeSeries

-TimeSeries dùng để load kiểu dữ liệu đa biến. Nó tạo một wrapper bọc xung quanh các UnivariateTimeSeries. dùng kiểu dữ liệu này vì nó giúp tốt hơn trong việc xử lí với các dữ liệu bị thiếu hay khác thời gian giữa các biến.Nó cũng hỗ trợ chuyển đổi từ pandas dataframe sang datetime với timestamp đơn vị là mili. -Một số tính năng thao tác với dữ liệu trước khi visualize -TimeSeries sẽ biểu diễn các biến dữ liệu xen kẽ với nhau khi ta chưa kiểm tra nó có đồng bộ về timestamp.

```
Univariate kpi_renamed
2020-03-02 09:12:00     667.118
2020-03-02 09:13:00     611.751
2020-03-02 09:14:00     599.456
2020-03-02 09:15:00     621.446
2020-03-02 09:16:00    1418.234
                          ...
2020-05-01 23:46:00     874.214
2020-05-01 23:47:00     937.929
2020-05-01 23:48:00    1031.279
2020-05-01 23:49:00    1099.698
2020-05-01 23:50:00     935.405
Name: kpi_renamed, Length: 86807, dtype: float64

Univariate kpi_label
2020-03-02 09:12:00    0.0
2020-03-02 09:13:00    0.0
2020-03-02 09:14:00    0.0
2020-03-02 09:15:00    0.0
2020-03-02 09:16:00    0.0
                      ...
2020-05-01 23:46:00    0.0
2020-05-01 23:47:00    0.0
2020-05-01 23:48:00    0.0
2020-05-01 23:49:00    0.0
2020-05-01 23:50:00    0.0
Name: kpi_label, Length: 86807, dtype: float64
```

Nó hỗ trợ method is_aligned để giúp kiểm tra đồng bộ về thời gian giữ các biến dữ liệu

```
not_aligned = TimeSeries({"kpi": kpi[1:],                # 2020-03-02 09:13:00 to
                          "kpi_label": kpi_label[:-1]})  # 2020-03-02 09:12:00 to
print(f"Is aligned? {not_aligned.is_aligned}")
```

```
Is aligned? False
```

Và một số method để xem dữ liệu một cách rõ với tiện hơn: +method window sẽ hiện dữ liệu trong khung thời gian ta mong muốn. Các biến không đồng bộ khi trước sẽ có giá trị là Nah. Và dữ liệu các biến sẽ hiện chung trong một bảng.

```
# Note that the first value of the KPI (which is missing in not_aligned) is NaN
not_aligned.window(1583140320, 1583226720)
```

```
                         kpi  kpi_label
2020-03-02 09:12:00      NaN        0.0
2020-03-02 09:13:00  611.751        0.0
2020-03-02 09:14:00  599.456        0.0
2020-03-02 09:15:00  621.446        0.0
2020-03-02 09:16:00 1418.234        0.0
...                      ...        ...
2020-03-03 09:07:00 1132.564        0.0
2020-03-03 09:08:00 1087.037        0.0
2020-03-03 09:09:00  984.432        0.0
2020-03-03 09:10:00 1085.008        0.0
2020-03-03 09:11:00 1020.937        0.0

[1440 rows x 2 columns]
```

+method bisect sẽ hiện dữ liệu theo 2 bảng trái phải theo móc thời gian.

```
left, right = aligned.bisect("2020-05-01")
print(f"Left\n{left}\n")
print()
print(f"Right\n{right}\n")
```

```
Left
                         kpi  kpi_label
2020-03-02 09:12:00  667.118        0.0
2020-03-02 09:13:00  611.751        0.0
2020-03-02 09:14:00  599.456        0.0
2020-03-02 09:15:00  621.446        0.0
2020-03-02 09:16:00 1418.234        0.0
...                      ...        ...
2020-04-30 23:55:00 1296.091        0.0
2020-04-30 23:56:00 1323.743        0.0
2020-04-30 23:57:00 1203.672        0.0
2020-04-30 23:58:00 1278.720        0.0
2020-04-30 23:59:00 1217.877        0.0

[85376 rows x 2 columns]


Right
                         kpi  kpi_label
2020-05-01 00:00:00 1381.110        0.0
2020-05-01 00:01:00 1807.039        0.0
2020-05-01 00:02:00 1833.385        0.0
2020-05-01 00:03:00 1674.412        0.0
2020-05-01 00:04:00 1683.194        0.0
...                      ...        ...
2020-05-01 23:46:00  874.214        0.0
2020-05-01 23:47:00  937.929        0.0
2020-05-01 23:48:00 1031.279        0.0
2020-05-01 23:49:00 1099.698        0.0
2020-05-01 23:50:00  935.405        0.0

[1431 rows x 2 columns]
```

3.3 Visualization

Visualization ở merlion chủ yếu để phục vụ cho 2 vấn đề học máy chuỗi thời gian là anomaly với forecasting.
Với mỗi vấn đề nó cung cấp 2 loại method viết kết hợp với 1 trong 2 thư viện matplotlib và plotly cho người dùng dễ dàng chọn lựa và sử dụng.
Đồng thời nếu họ không chỉ muốn visualize chỉ dụng các method của merlion thì merlion còn cung cấp cách visualize thông qua tạo và truyền dữ liệu vào hai lớp đối tượng là Figure(dành cho visualize đơn biến) và MTSFigure(dành cho visualize đa biến)

3.3.1 Visualization for anomaly

Sau đây là thực hành với anomaly với method của merlion là plot_anomaly kết hợp với thư viện plt của matplotlib.
-Trong model sẵn có trong merlion sẽ được hỗ trợ hàm này để hiện ra được anomaly score của

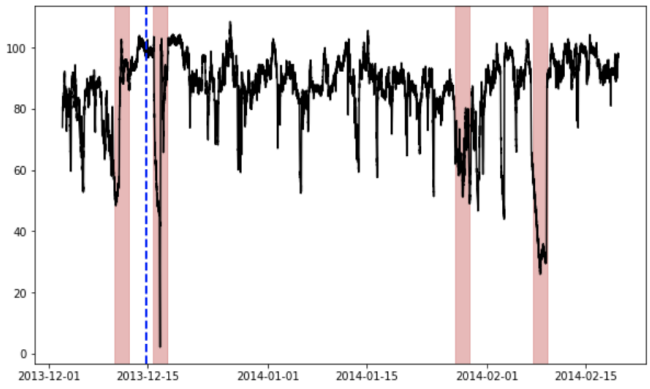
model đã dự đoán ra.
-Còn dùng model ngoài ta sẽ dùng model bên ngoài thì ta sẽ dùng với các class Figure cho đơn biến và MTSFigure cho đa biến. Ngoài ra còn sử dụng được thêm hai hàm là plot_anoms để tạo cột thẳng đứng màu chìm nền để dễ quan sát được dữ liệu đã label.
Đầu tiên là load data vào. Ta có visualize data chia ra hai tập train và test. Tập test với train được chia ra tại timestamp là: "2013-12-14 16:45:00"

3.3.1.a Visualize dữ liệu ban đầu

-Visualize kết hợp với thư viện matplotlib
Visualize tập dữ liệu ban đầu với hàm plot_anoms kết hợp matplotlib

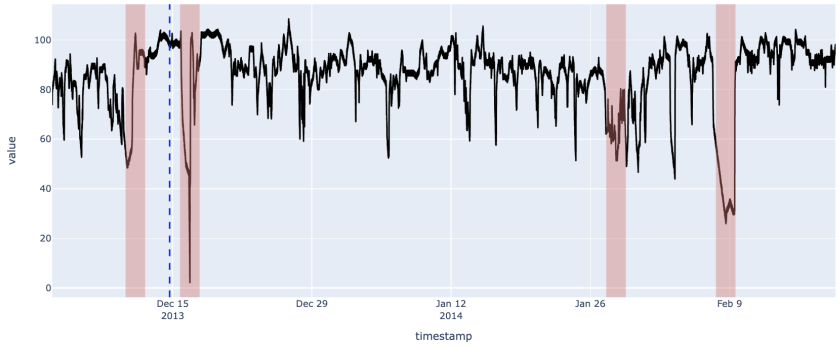
```
[226] import matplotlib.pyplot as plt


      from merlion.plot import plot_anoms
      from merlion.utils import TimeSeries
      from ts_datasets.anomaly import NAB

      np.random.seed(1234)

      time_series, metadata = NAB(subset="realKnownCause")[3]

      # Visualize the full time series
      fig = plt.figure(figsize=(10, 6))
      ax = fig.add_subplot(111)
      ax.plot(time_series,c='k')


      ax.axvline(metadata[metadata.trainval].index[-1], ls="--", lw=2, c="b")
      plot_anoms(ax, TimeSeries.from_pd(metadata.anomaly))
```

```
WARNING:ts_datasets.anomaly.nab:Time series /usr/local/lib/python3.7/data/nab/realKnownCau
WARNING:ts_datasets.anomaly.nab:Time series /usr/local/lib/python3.7/data/nab/realKnownCau
```



Sau khi train với một model có sẵn trong thư viện merlion. Ta visualize kết quả cuối cùng. Ở đây là ta dùng hàm plot_anomaly tương ứng với thư viện matplotlib.
-Visualize kết hợp với thư viện plotly
Có thể sử dụng plotly_anomaly_plotly thay cho các bạn quen dùng thư viện plotly.

```
from merlion.plot import plot_anoms_plotly
import plotly.express as px
fig=px.line(time_series,x=time_series.index,y='value')
fig.update_traces(line_color='black')
plot_anoms_plotly(fig,TimeSeries.from_pd(metadata.anomaly))
fig.add_vline(x=metadata[metadata.trainval].index[-1], line_width=2, line_dash="dash", line_color="blue")
fig.show()
```



3.3.1.b Visualize cho model anomly mà merlion hỗ trợ

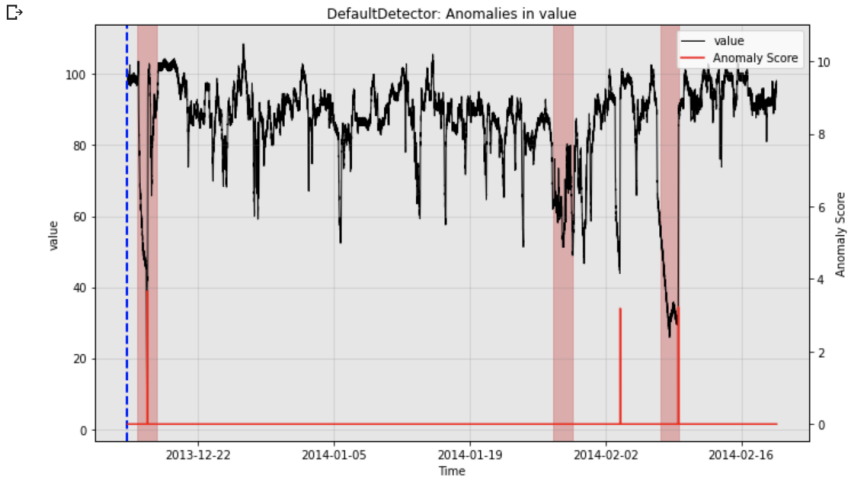
Train model có sẵn của merlion

Train model Merlion hỗ trợ

```
[ ] from merlion.utils import TimeSeries

    # Get training split
    train = time_series[metadata.trainval]
    train_data = TimeSeries.from_pd(train)
    train_labels = TimeSeries.from_pd(metadata[metadata.trainval].anomaly)

    # Get testing split
    test = time_series[~metadata.trainval]
    test_data = TimeSeries.from_pd(test)
    test_labels = TimeSeries.from_pd(metadata[~metadata.trainval].anomaly)
    from merlion.models.defaults import DefaultDetectorConfig, DefaultDetector
    model = DefaultDetector(DefaultDetectorConfig())
    model.train(train_data=train_data)
    test_pred = model.get_anomaly_label(time_series=test_data)
```

Visualize cho model

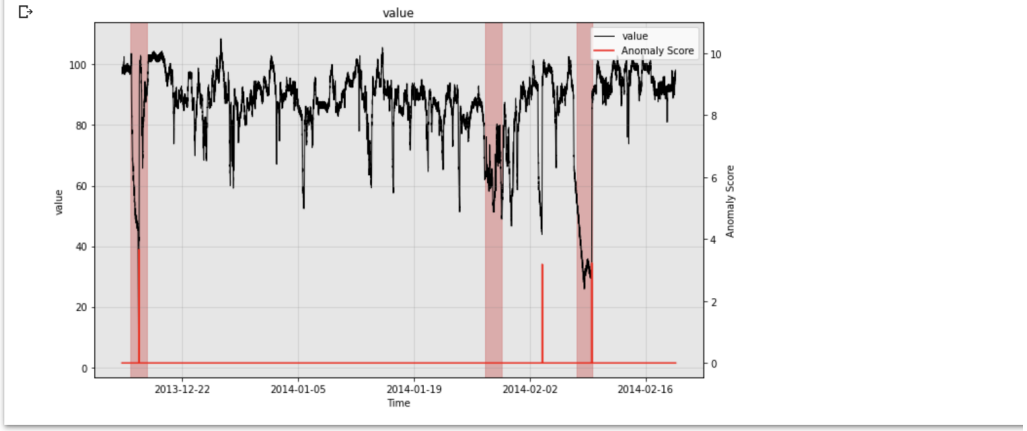
```
fig, ax = model.plot_anomaly(
        time_series=test_data, time_series_prev=train_data,
        )
ax.axvline(metadata[metadata.trainval].index[-1], ls="--", lw=2, c="b")
plot_anoms(ax=ax, anomaly_labels=test_labels)
plt.show()

# Label the train/test split with a dashed line & plot anomalies
```



3.3.1.c Visualize cho model anomly mà merlion không hỗ trợ

Hoặc ta có thể cho visualize model anomaly bên ngoài merlion cung cấp và truyền output vào class Figure. Hàm figure thì visualize cho đơn biến nên dữ liệu truyền vào là dạng UnvariateTimeSeries

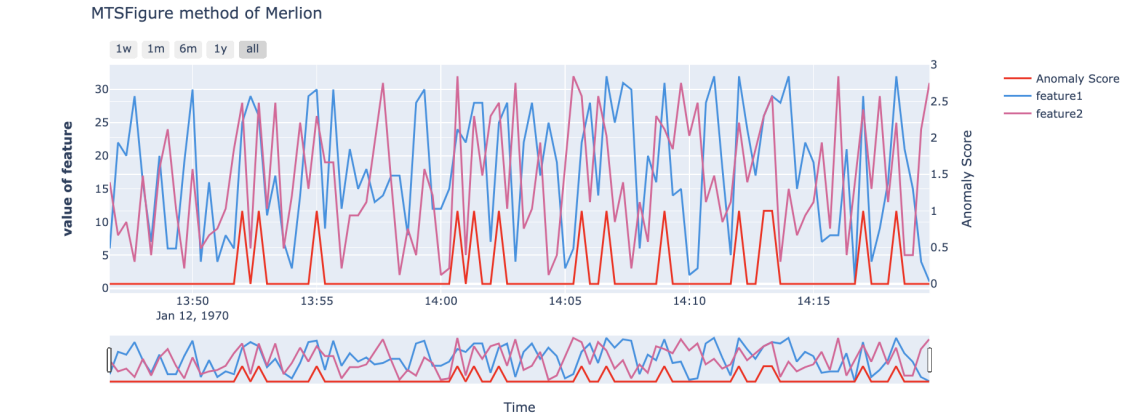
```
from merlion.plot import Figure


#metric_name=['H6','H6']
fig,ax= Figure(test_data.univariates['value'],anom=test_pred.univariates['anom_score']).plot()
plot_anoms(ax=ax, anomaly_labels=test_labels)
```



3.3.1.d Visualize cho dữ liệu đa biến

Ngược lại với dữ liệu đa biến thì sử dụng hàm MTSFigure dữ liệu truyền vào sẽ dưới dạng TimeSeries

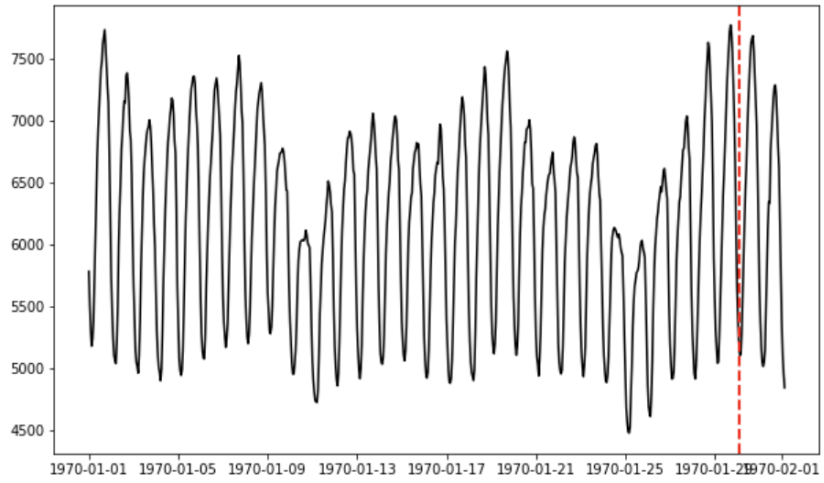
```
from merlion.plot import MTSFigure
fig= MTSFigure(y=TimeSeries([kpi,kpi2]),anom=TimeSeries([kpi_label])).plot_plotly()
fig.update_yaxes(title_text="<b>value of feature</b> ", secondary_y=False)
fig.update_layout(
    title_text="MTSFigure method of Merlion"
)
fig.show()
```



3.3.2 Visualization for forecasting

3.3.2.a Visualization cho dữ liệu ban đầu

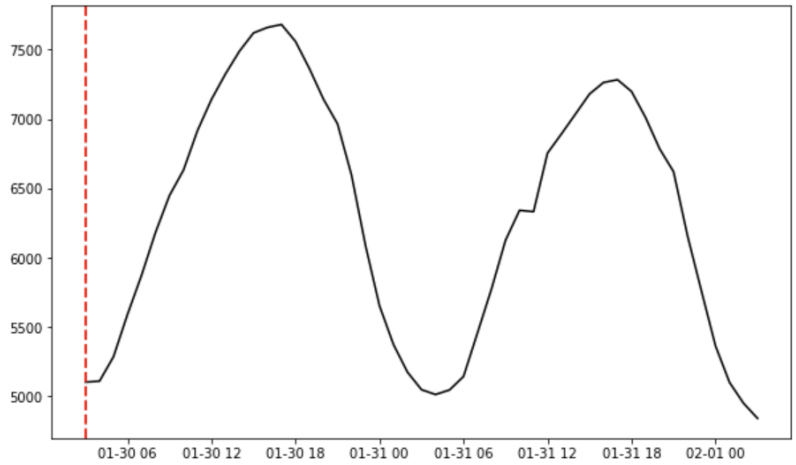
Sau đây là thực hành với forecasting với method của merlion là plot_forecast kết hợp với thư viện plt của matplotlib. Đầu tiên là load data vào. Ta có visualize data chia ra hai tập train và test. Tập test là 2 ngày cuối trong data. 30-1 và 1-2




```
700 points in train split, 48 points in test split.
```

Sau đó ta lấy tập test ra train model. visualize matplotlib lại tập test cho dễ so sánh của

visualize của merlion

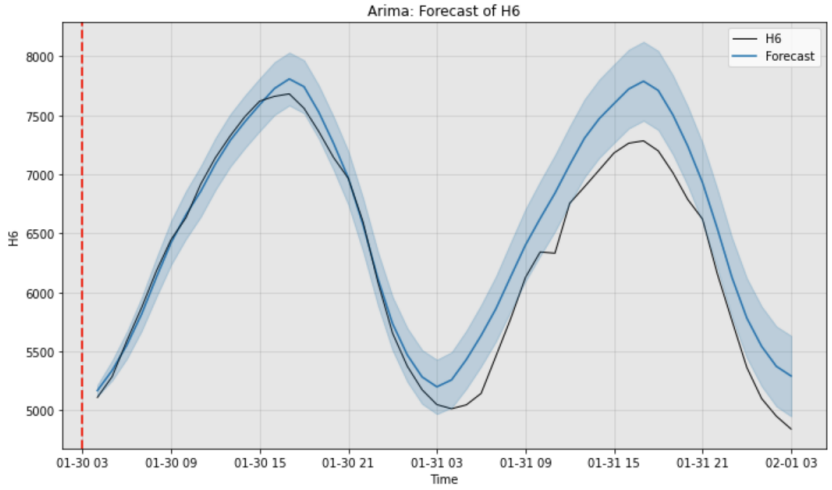


3.3.2.b Visualization cho model merlion hỗ trợ bằng plot_forecast hoặc plot_forecast_plotly

Sau khi train với một model có sẵn trong thư viện merlion. Ta visualize kết quả cuối cùng. Ở đây là ta dùng hàm plot_forecast tương ứng với thư viện matplotlib. Có thể sử dụng plotly_forecast thay cho các bạn quen dùng thư viện ploty. Với hàm này ta có thể hiện dễ dàng dễ hiểu về lowbound và upbound của model dự đoán Dãi màu xanh trong hình là khoảng dự đoán giá trị có thể của y sẽ xãy ra.

```
from merlion.evaluate.forecast import ForecastMetric

smape1 = ForecastMetric.sMAPE.value(ground_truth=sub_test_data,
                                    predict=forecast1)

fig, ax = model1.plot_forecast(time_series=sub_test_data,
                               plot_forecast_uncertainty=True)

ax.axvline(time_series[trainval].index[-1], ls="--", lw="2", c="r")
```

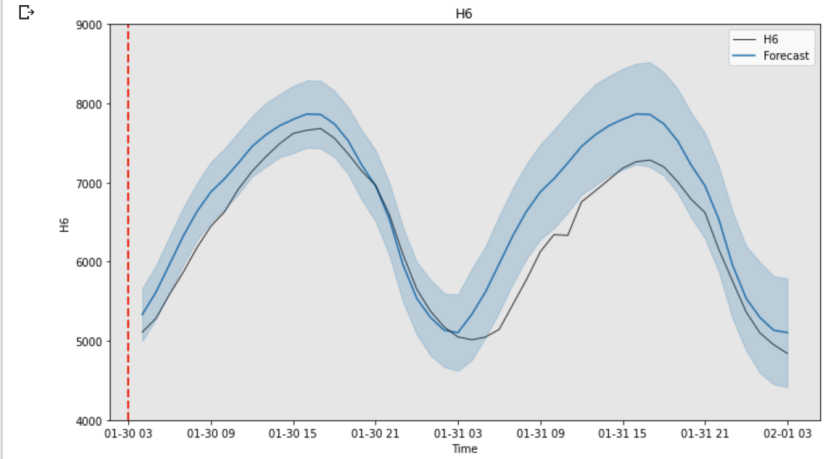
```
<matplotlib.lines.Line2D at 0x7fe6fbdcd5d0>
```



3.3.2.c Visualization cho model bên ngoài merlion không hỗ trợ bằng Figure or MTSFigure cho đa biến

Hoặc ta có thể dùng model forecasting bên ngoài tự làm mà merlion không cung cấp sẵn và chỉ cần truyền output vào class Figure có thể visualize được dữ liệu. Lúc này thì ta chỉ cần truyền đầy đủ tham số vào đúng dạng là đơn biến với UnvaritateTimeSeries là y dự đoán và y đúng, y lowbound và upbound là có thể vẽ hình.

```
#giả lập ta có một tập data đã train với model bên ngoài. Ta chuyển về dạng univarites của merlion
time_stamps = sub_test_data.univariates[sub_test_data.names[0]].time_stamps
forecast1, stderr1 = model.forecast(time_stamps=time_stamps)
from merlion.utils import UnivariateTimeSeries
t=stderr1.univariates['H6_err'].to_pd()
t1=forecast1.univariates['H6'].to_pd()
t2=t1-t
t3=t1+t
t2=UnivariateTimeSeries(time_stamps=t2.index,values=t2.values)
t3=UnivariateTimeSeries(time_stamps=t3.index,values=t3.values)
#visualize data by Figure
fig,ax=Figure(sub_test_data.univariates['H6'],yhat=forecast1.univariates['H6'],yhat_lb=t2,yhat_ub=t3).plot()
ax.axvline(time_series[trainval].index[-1], ls="--", lw="2", c="r")
ax.set_ylim(4000,9000)
ax.grid(False)
```



3.3.3 Ưu điểm

- Tiện lợi nhanh dễ hiểu và dễ sử dụng.
- Hỗ trợ gần đầy đủ đa số các model cho 2 vấn đề anomaly và forecasting.
- Có thể so sánh và visualize được các model mà merlion hỗ trợ trong việc giải quyết các bài toán timeseries. Từ đó users dễ sử dụng và visualize dữ liệu. Merlion cũng cung cấp các method giúp kiểm tra và phát hiện độ chính xác qua các cột màu hay là dãi lowbound và upbound.

3.3.4 Nhược điểm

- Do tiện lợi nhanh gọn trong việc vẽ các biểu đồ cho các bài toán time series nên nó gặp một số ít nhược điểm trong việc tinh chỉnh các thông số cho việc visualize theo ý người dùng, một số method visualize merlion đã cố định dạng dữ liệu. Nếu muốn thay đổi sẽ phức tạp hoặc không thể thay đổi được.

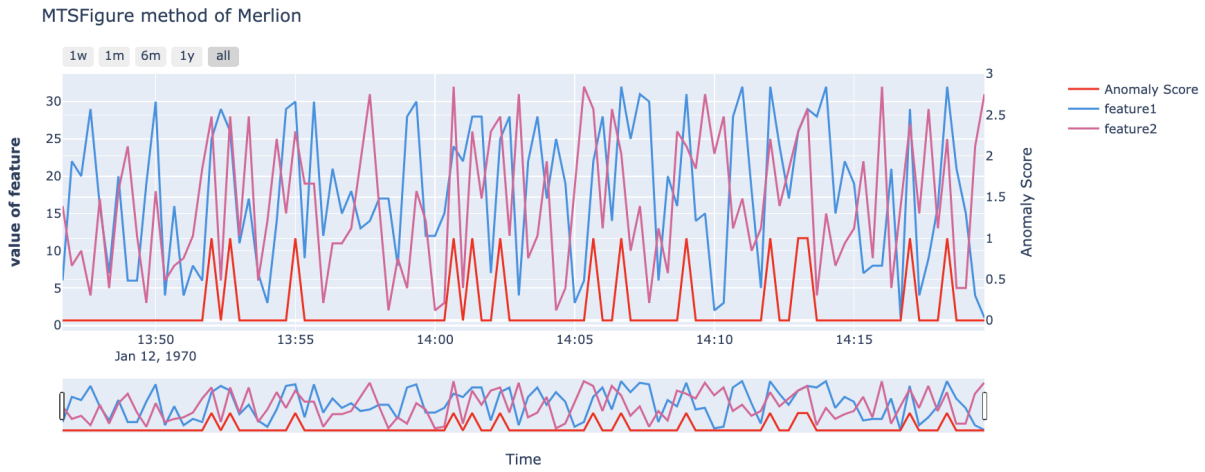
```
        anom_trace = None
        if self.anom is not None:
            v = self.anom.univariates[self.anom.names[0]]
            anom_trace = go.Scatter(
                name="Anomaly Score", x=v.index, y=v.np_values, mode="lines", line=dict(color=anom_color)
            )

        fig = make_subplots(
            specs=[[{"secondary_y": anom_trace is not None}]], figure=go.Figure(layout=self._get_layout(title, figsize))
        )
        if anom_trace is not None:
            fig.add_trace(anom_trace, secondary_y=True)
```

- Do code đã cố định mode của anomaly data. Cùng một kiểu "mode=lines" như trong mã nguồn của merlion. dẫn đến việc biểu đồ anomly cho đa biến thì anomaly score bị khó nhìn.
- Hoặc màu của anomaly không thể đổi để giúp dễ nhìn hơn.

```
            prediction_color = "#0072B2"
            error_color = "rgba(0, 114, 178, 0.2)"  # '#0072B2' with 0.2 opacity
            actual_color = "black"
            anom_color = "red"
            line_width = 2

```

- Dẫn đến việc visualize gọn nhẹ chỉ qua một dòng code nhưng lại bị khó nhìn.

```
from merlion.plot import MTSFigure
fig= MTSFigure(y=TimeSeries([kpi,kpi2]),anom=TimeSeries([kpi_label])).plot_plotly()
fig.update_yaxes(title_text="<b>value of feature</b> ", secondary_y=False)
fig.update_layout(
    title_text="MTSFigure method of Merlion"
)
fig.show()
```



Source Code

https://colab.research.google.com/drive/1rdts416JtrwwXDTOrBH65fzpPdbArKbNscrollTo=r7Z4K38EL6Qs

Tài liệu

[Prophet Documentation] https://facebook.github.io/prophet/docs/quick_start.html

[Merlion Documentation] https://https://opensource.salesforce.com/Merlion/v1.1.0/index.html