Azure Machine Learning Meetup

Oct 10, 2019

Anomaly Detection as A Service

Azure Cognitive Services

agenda

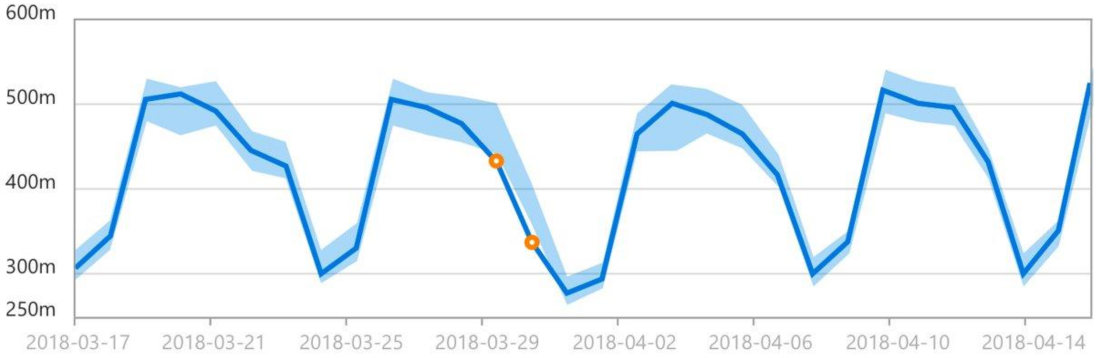
- Welcome and intros
- Presentation on the Anomaly Detection Service
- Demo
- Q&A

What is Anomaly Detection

Recognition of unusual patterns of behavior in data that don’t conform to expected outcomes.

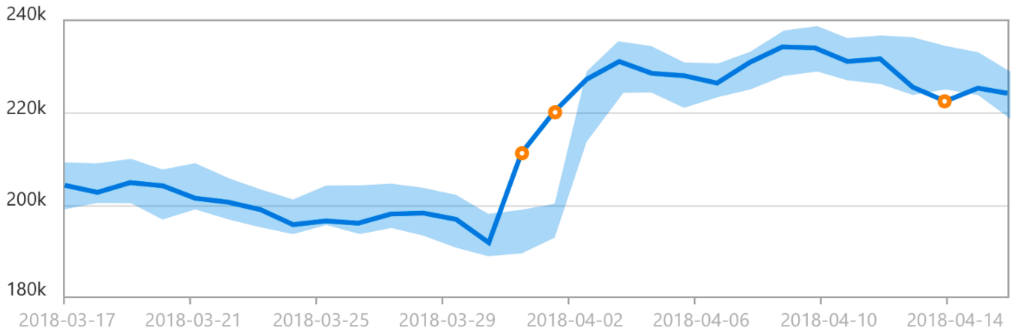
Example:

To ensure the health of your business, you want to track your key metrics like revenue and understand whether something is out of historical pattern.

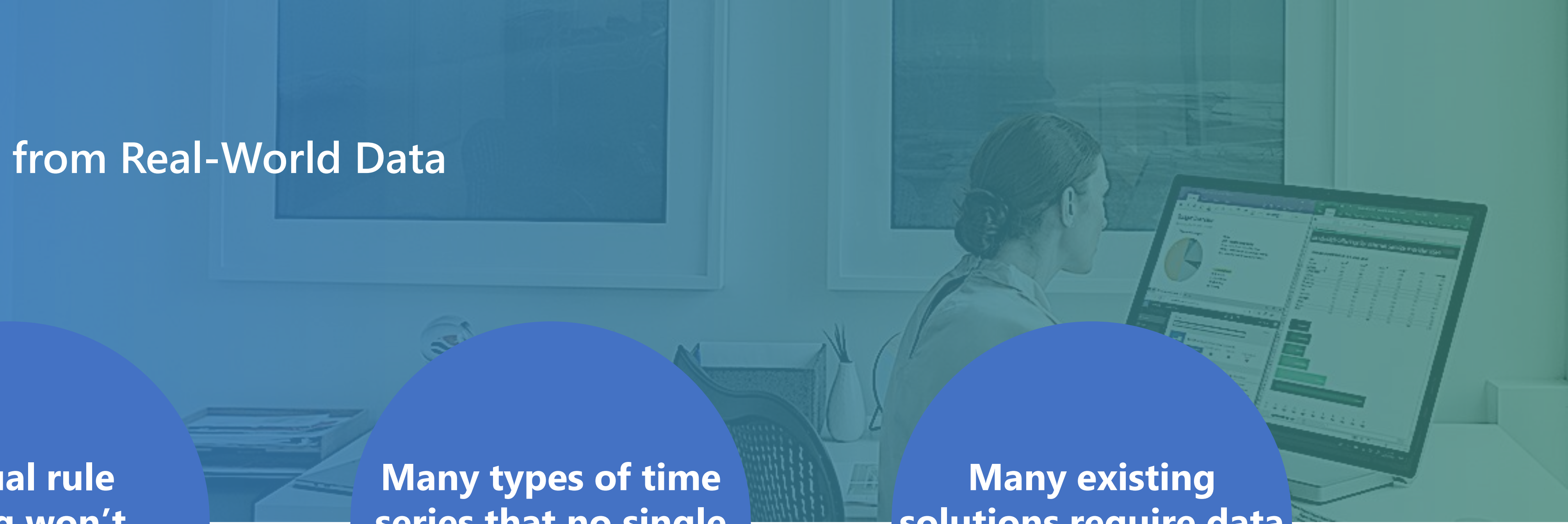


Example:

Sensor time series data, you want to be alerted on the drifting which could imply system malfunctions.



The Challenges from Real-World Data



Manual rule setting won't scale and adapt

AD learns from the data on rules which differentiate outliers from normal pattern automatically

Many types of time series that no single algorithm fits all

AD automatically selects the best pre-built model from model pool behind the scene

Many existing solutions require data science knowledge

AD hides the complexity and provide ONE intuitive parameter to change sensitivity

Anomaly Detector PREVIEW

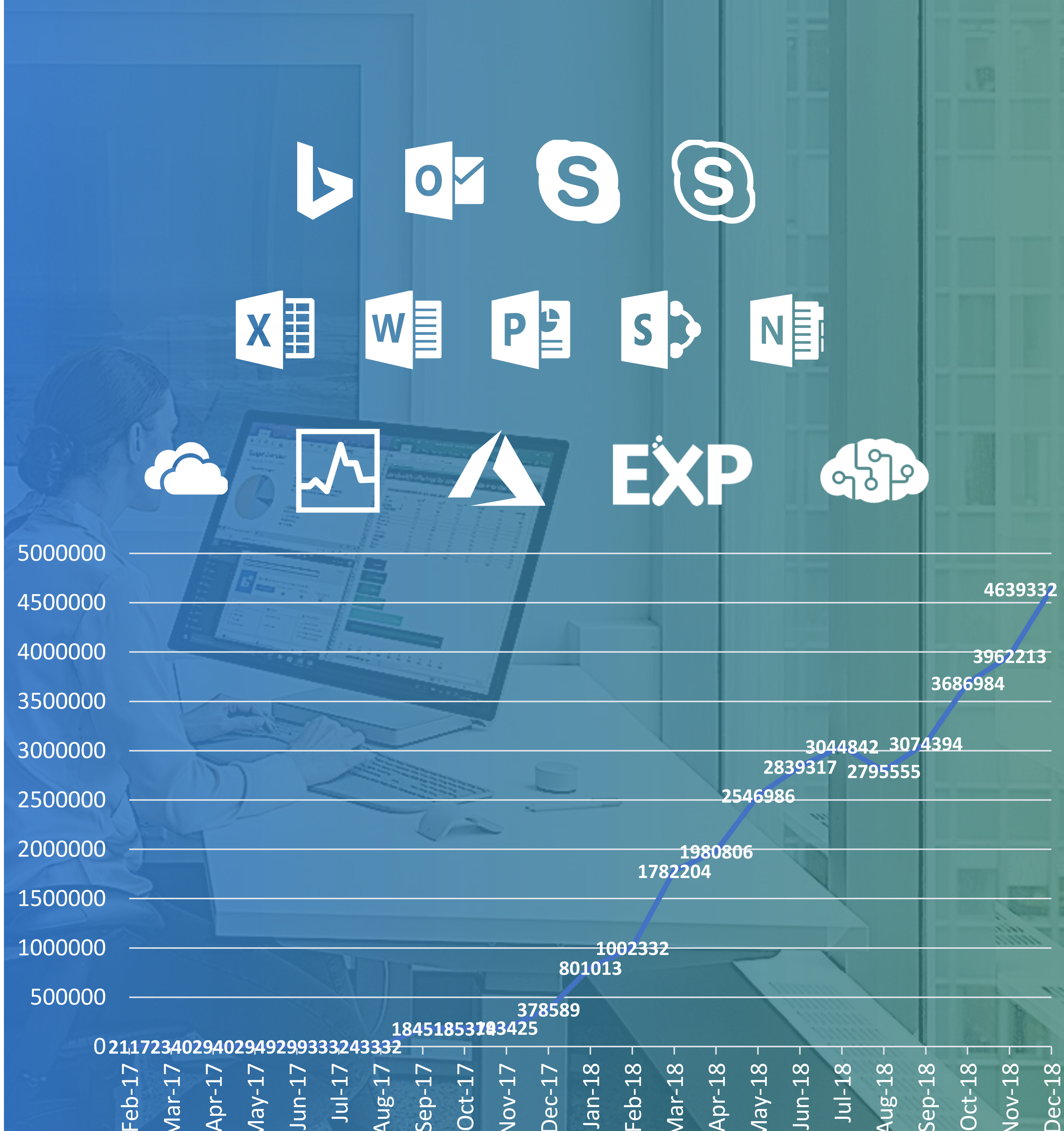
An AI service that helps you foresee problems before they occur

Proven Technology within Microsoft

- 400+ teams across Azure, Windows, Office, Bing…
- Millions of time series
- Thousands of active users within Microsoft

Popular in industry

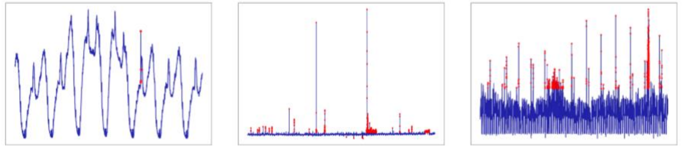
- 300+ Tenants, 600+ Azure subscriptions
- Millions of daily transactions
- Industries: telecom, network, consulting, manufacturing, real estate, finance, education…



Anomaly Detection as a Service

- Anomaly detection aims to discover unexpected events or rare items in data. Accurate anomaly detection leads to prompt intervention.
- Anomaly Detector provides two APIs:
  - Batch
  - Item
  - Both detect anomalies automatically in time series with simple parameters, which require little machine learning expertise.
- It is designed for the scenarios of operational monitoring, business KPI monitoring, and IoT monitoring.
- Using Anomaly Detector APIs, you can infuse anomaly detection capabilities into your own platform and business process.

- Note: this is available as a self-hosted container for on-premises or lower latency scenarios



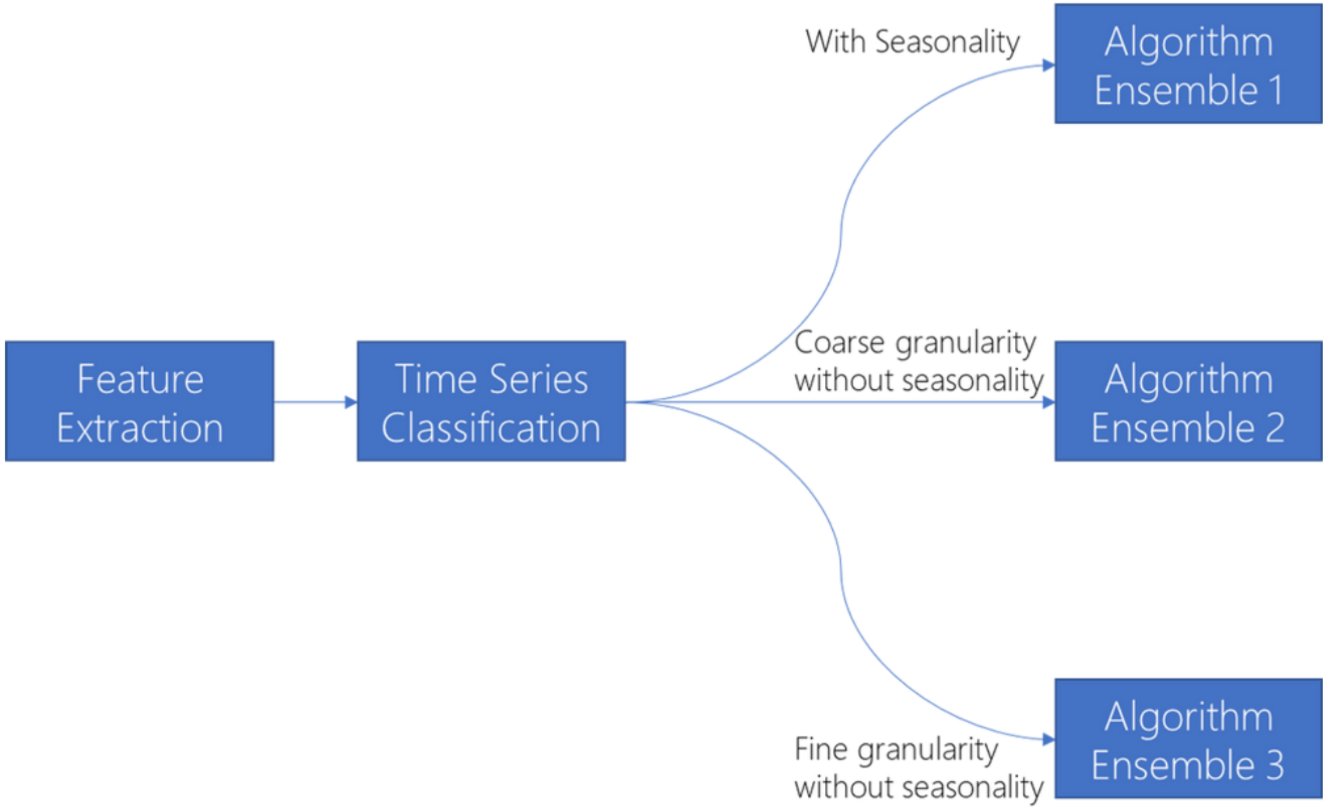
(a) seasonal (b) stable (c) unstable

Figure 1: Different types of time-series.

Algorithm Challenges

- Challenges
  - Lack of labels
  - Generalization
  - Efficiency

The following picture shows the algorithm selecting flow of Anomaly Detector. We will use another blog for more details on the algorithms.



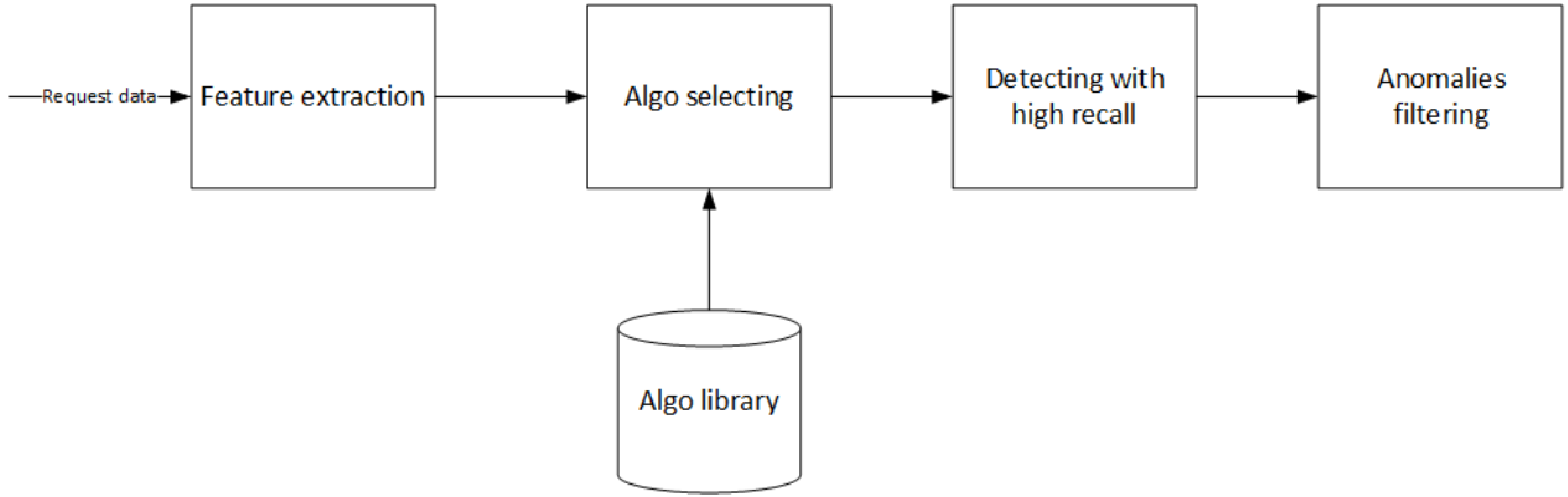
Handling various types of flows

Detecting all kinds of anomalies through one single endpoint

Besides spikes and dips, Anomaly Detector also detects many other kinds of anomalies, such as trend change and off-cycle softness, all in one single API endpoint.

Also

- Maintain simplicity outside: One parameter tuning
- Maintain sophistication inside: Selecting / Detecting / Filtering
- main parameter you need to customize is “Sensitivity”, which is from 1 to 99
- the system selects the most appropriate algorithm.



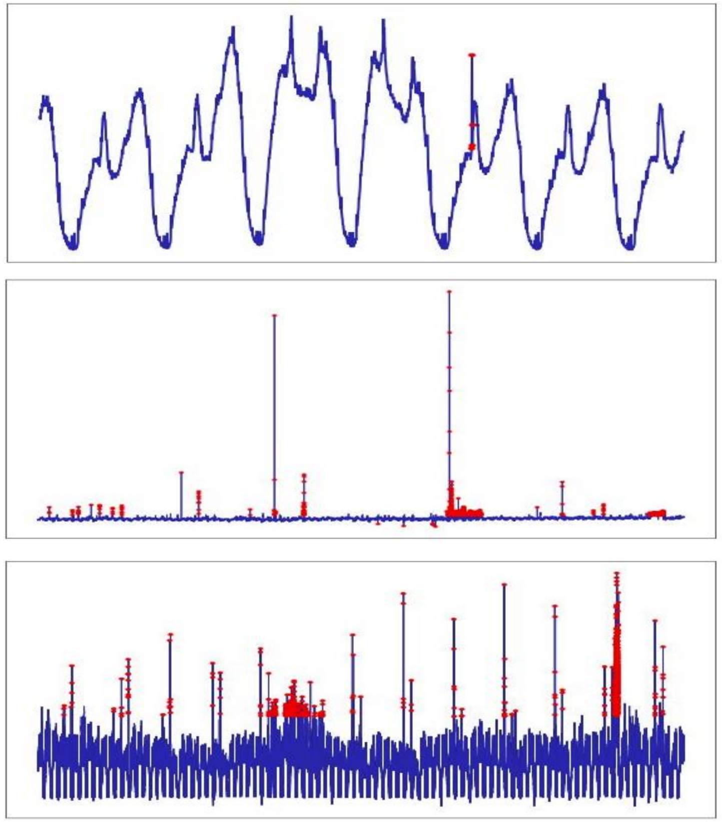
SR-CNN algorithm

Inspiration

visual saliency detection

https://blog.algorithmia.com/introduction-image-saliency-detection/

time-series anomaly detection



Saliency Detection Inspiration

- From the research paper:

The motivation is that the time-series anomaly detection task is similar to the problem of visual saliency detection essentially.

Saliency is what "stands out“ in a photo or scene, enabling our eye-brain connection to quickly (and essentially unconsciously) focus on the most important regions. Meanwhile, when anomalies appear in time-series curves, they are always the most salient part in vision.

- KDD ’19, August 4–8, 2019

Prepping for code and demo

Batch API calls use: /timeseries/entire/detect.

sending your time series data at once, the API will generate a model using the entire series and analyze each data point with it.

Streaming API calls use: /timeseries/last/detect

send new data points as you generate them, you can monitor your data in real time. A model will be generated with the data points you send, and the API will determine if the latest point in the time series is an anomaly

And you always need:

subscription_key = ‘’ “ #your private key

endpoint_latest = 'https://westus2.api.cognitive.microsoft.com/anomalydetector/v1.0/ #append as appropriate

Core routine for calling Streaming detection

```
single_sample_data = {}
    single_sample_data['series'] = points[i-29:i]
    single_sample_data['granularity'] = 'daily'
    single_sample_data['maxAnomalyRatio'] = 0.25  #cap on % of anomalies to detect
           # max is 50%
    single_sample_data['sensitivity'] = sensitivity
    single_point = detect(endpoint_latest, subscription_key, single_sample_data)
    if single_point['isAnomaly'] == True:
      anom_count = anom_count + 1

    result['expectedValues'][i-1] = single_point['expectedValue']
    result['upperMargins'][i-1] = single_point['upperMargin']
    result['lowerMargins'][i-1] = single_point['lowerMargin']
    result['isNegativeAnomaly'][i-1] = single_point['isNegativeAnomaly']
    result['isPositiveAnomaly'][i-1] = single_point['isPositiveAnomaly']
    result['isAnomaly'][i-1] = single_point['isAnomaly']
```

The detect function

```
def detect(endpoint, subscription_key, request_data):
    headers = {'Content-Type': 'application/json', 'Ocp-Apim-Subscription-
Key’:    subscription_key}
    response = requests.post(endpoint, data=json.dumps(request_data),
headers=headers)
    if response.status_code == 200:
        return json.loads(response.content.decode("utf-8"))
    else:
        print(response.status_code)
        raise Exception(response.text)
```

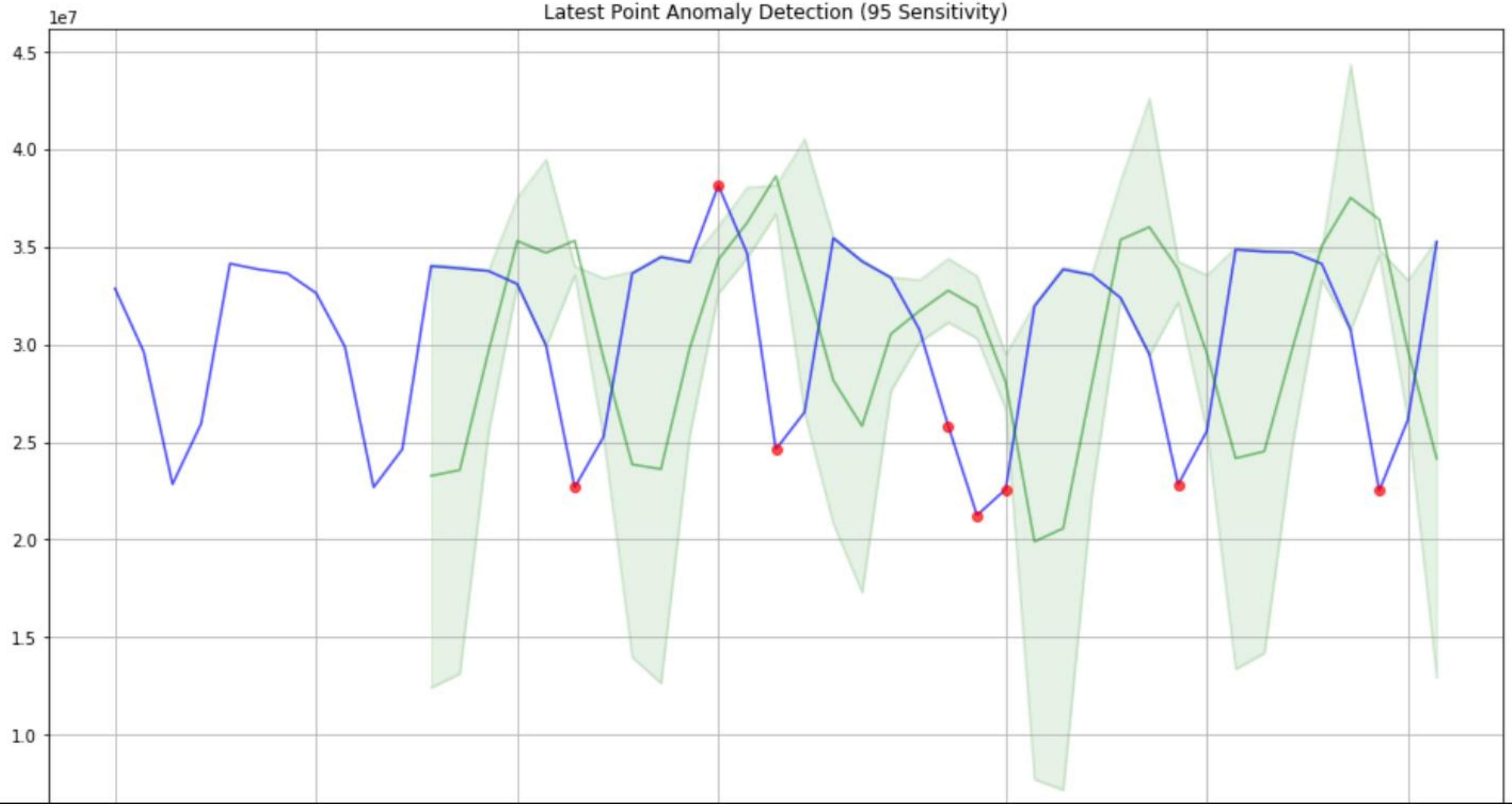
demo

Walkthrough

Window size matters for latest detection

```
stream_data(df_daily, 12, sensitivity=95) #was 29
```

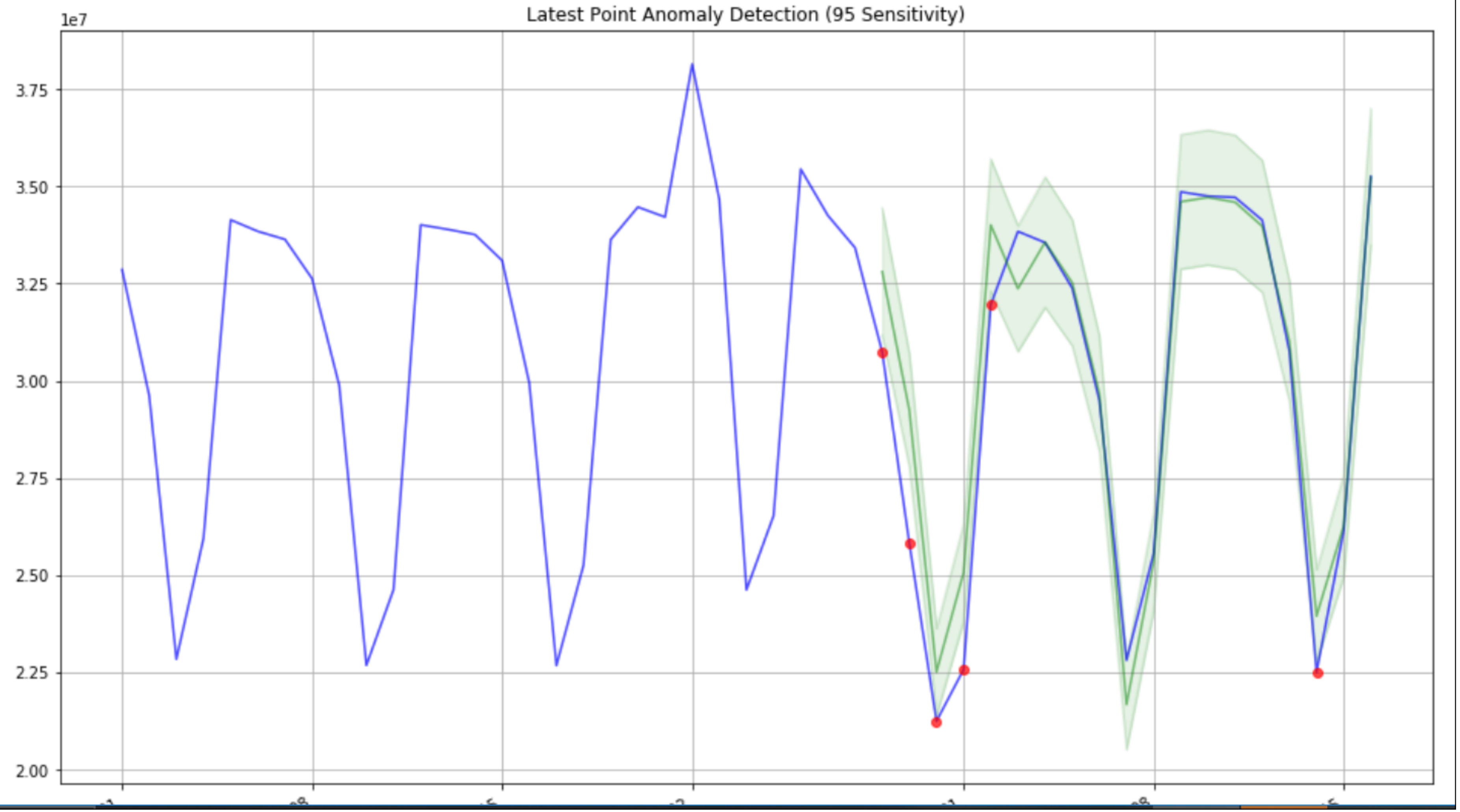
```
detect latest window size is  12
```



Compare with prior slide – makes sense since this is a window by window type analysis

```
stream_data(df_daily, 29, sensitivity=95)
```

```
detect latest window size is  29
```



Guidance

- For **best results** when using the Anomaly Detector API, your JSON-formatted time series data should include:
  - data points separated by the same interval, with no more than 10% of the expected number of points missing.
  - at least 12 data points if your data doesn't have a clear seasonal pattern.
  - at least 4 pattern occurrences if your data does have a clear seasonal pattern.

Best practices 1

- Data points sent to the Anomaly Detector API must have a valid Coordinated Universal Time (UTC) timestamp, and numerical value


```
{
  "granularity": "daily",
  "series": [
   {
     "timestamp": "2018-03-01T00:00:00Z",
     "value": 32858923
   },
   {
     "timestamp": "2018-03-02T00:00:00Z",
     "value": 29615278
   },
  ]
}
```

Best practices 2

- For non-standard intervals

```
{
    "granularity" : "minutely",
    "customInterval" : 5
}
```

Missing data points

- Missing data points are common in evenly distributed time series data sets, especially ones with a fine granularity (A small sampling interval. For example, data sampled every few minutes).
- Missing less than 10% of the expected number of points in your data shouldn't have a negative impact on your detection results.
- Consider filling gaps in your data based on its characteristics like substituting data points from an earlier period, linear interpolation, or a moving average.

Seasonal trends

If you know that your time series data has a seasonal pattern (one that occurs at regular intervals), you can improve the accuracy and API response time.

Specifying a period when you construct your JSON request can reduce anomaly detection latency by up to 50%.

The period is an integer that specifies roughly how many data points the time series takes to repeat a pattern. For example, a time series with one data point per day would have a period as 7, and a time series with one point per hour (with the same weekly pattern) would have a period of 7*24.

If you're unsure of your data's patterns, you don't have to specify this parameter.

For best results, provide 4 period's worth of data point, plus an additional one. For example, hourly data with a weekly pattern as described above should provide 673 data points in the request body (7 * 24 * 4 + 1).

How this works – mapping raw events to the Saliency Map – note how the anomaly is more prominent on the saliency map

- The mapping is done using Fast Fourier Transforms

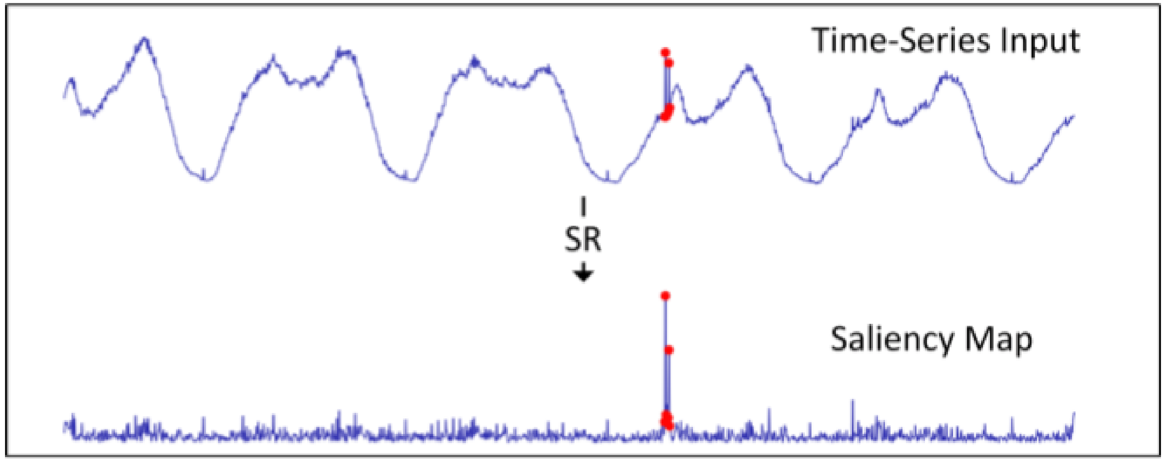
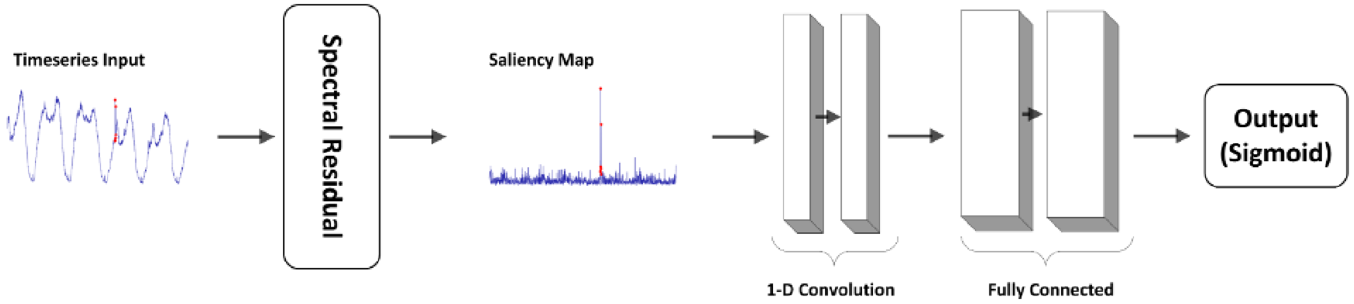


Figure 4: Example of SR model results

Algorithm Overview

- Borrow the Spectral Residual (SR) model from visual saliency detection field.
- Apply CNN on the saliency map produced by SR and train it with synthetic data.
- This is , in effect, a way to using labels (from the SR mapping) to train the CNN



The internal engine view

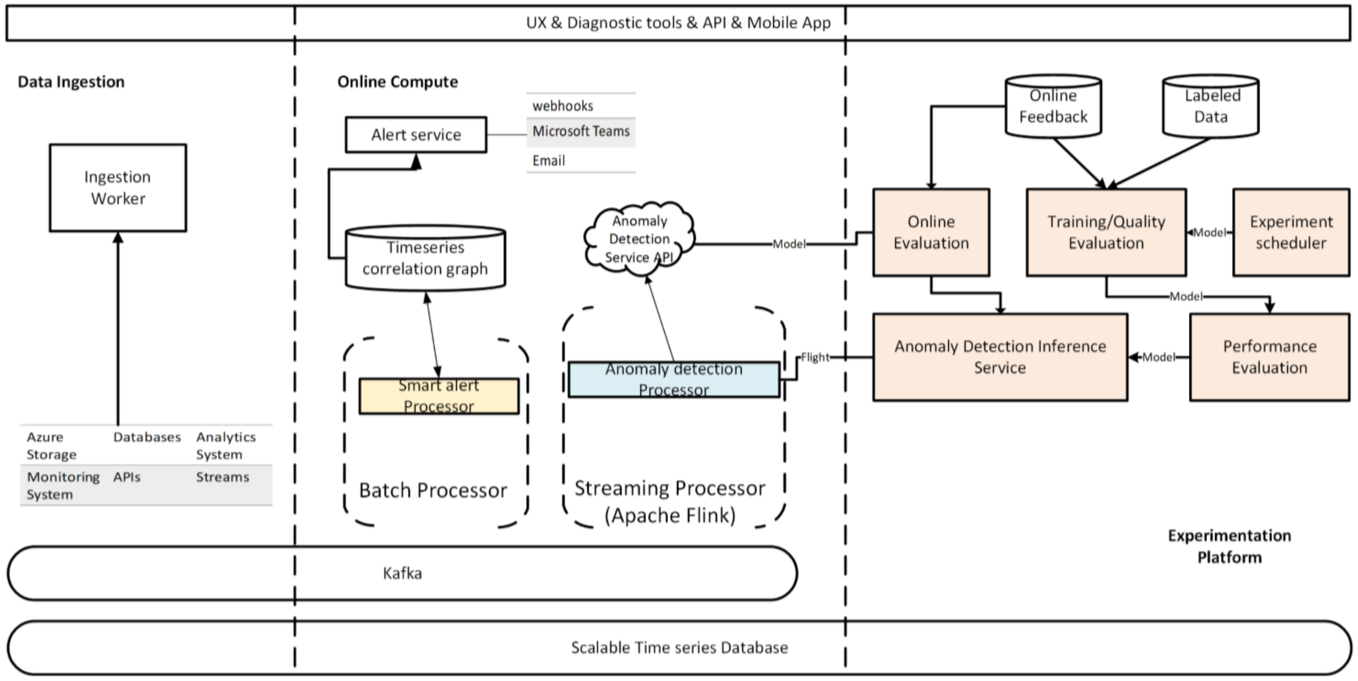
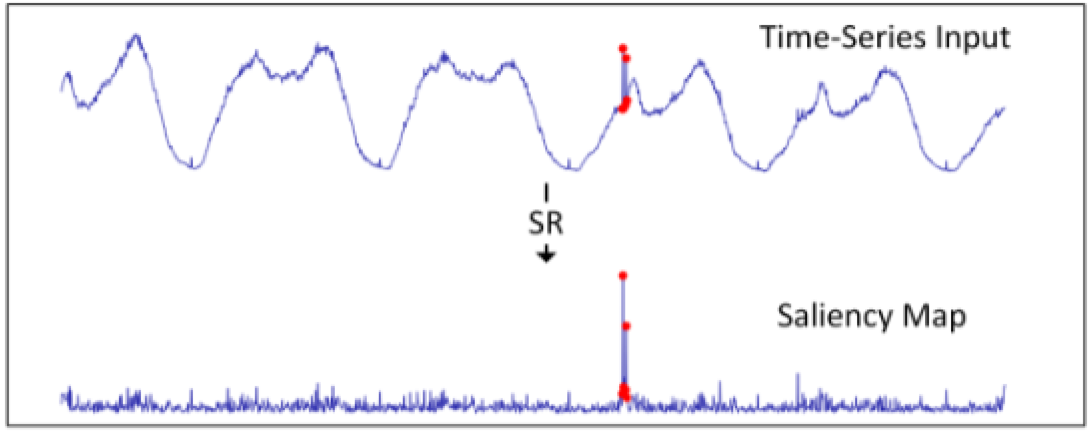


Figure 2: System Overview

Spectral Residual (SR)

1. Fourier Transform to get the log amplitude spectrum
2. Calculation of spectral residual
3. Inverse Fourier Transform that transforms the sequence back to spatial domain.

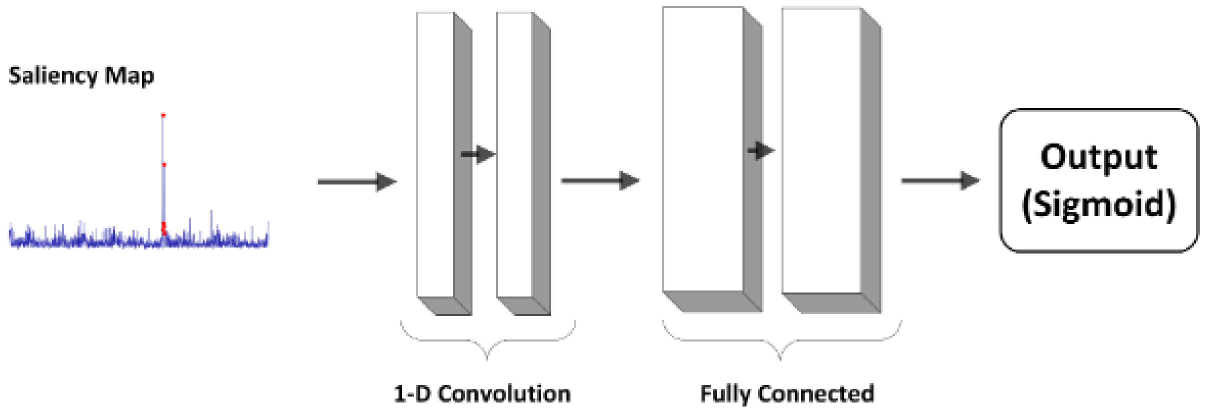


$$A(f) = Amplitude(\mathfrak{F}(\mathbf{x}))$$
$$P(f) = Phrase(\mathfrak{F}(\mathbf{x}))$$
$$L(f) = log(A(f))$$
$$AL(f) = h_n(f) \cdot L(f)$$
$$R(f) = L(f) - AL(f)$$
$$S(\mathbf{x}) = \mathfrak{F}^{-1}(exp(R(f) + P(f))^2)$$
$$n_f(f) = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix}$$

CNN & Training data synthesis

- CNN is responsible to learn a discriminative rule on saliency map
- Train through automatically generated anomalies
  - Randomly select several points in the time-series and calculate the injection value to replace the original point
  - Injection value

$$x = (\overline{x} + mean)(1 + var) \cdot r$$



Benefit of SR-CNN

- SR is unsupervised and accurate.
- SR is simple, efficient, and has good generality.
- It is unlikely to train CNN from the original time-series because of lack of labels. But we can train CNN on the saliency map using fully synthetic data.

Experimental Results - Accuracy & Efficiency

Table 2: Result comparison of cold-start

	KPI				Yahoo				Microsoft			
Model	F_1-score	*Precision*	*Recall*	*Time(s)*	F_1-score	*Precision*	*Recall*	*Time(s)*	F_1-score	*Precision*	*Recall*	*Time(s)*
FFT	**0.538**	0.478	0.615	3756.63	0.291	0.202	0.517	356.56	0.349	0.812	0.218	8.38
Twitter-AD	0.330	0.411	0.276	523232.0	0.245	0.166	0.462	301601.50	0.347	0.716	0.229	6698.80
Luminol	0.417	0.306	0.650	14244.92	**0.388**	0.254	0.818	1071.25	**0.443**	0.776	0.310	16.26
SR	0.666	0.637	0.697	1427.08	0.529	0.404	0.765	43.59	0.484	0.878	0.334	2.45
SR-CNN	**0.732**	0.811	0.667	6805.13	**0.655**	0.786	0.561	279.97	**0.537**	0.468	0.630	25.26

Table 3: Result comparison on test data

	KPI				Yahoo				Microsoft			
Model	F_1-score	*Precision*	*Recall*	*Time(s)*	F_1-score	*Precision*	*Recall*	*Time(s)*	F_1-score	*Precision*	*Recall*	*Time(s)*
SPOT	0.217	0.786	0.126	9097.85	**0.338**	0.269	0.454	2893.08	0.244	0.702	0.147	9.43
DSPOT	**0.521**	0.623	0.447	1634.41	0.316	0.241	0.458	339.62	0.190	0.394	0.125	1.37
DONUT	0.347	0.371	0.326	24248.13	0.026	0.013	0.825	2572.76	**0.323**	0.241	0.490	288.36
SR	0.622	0.647	0.598	724.02	0.563	0.451	0.747	22.71	0.440	0.814	0.301	1.55
SR-CNN	**0.771**	0.797	0.747	2724.33	**0.652**	0.816	0.542	125.37	**0.507**	0.441	0.595	16.13

Experimental Results - Generality

Table 4: Generality Comparison on Yahoo dataset

	Seasonal	Stable	Unstable	Overall	Var
FFT	**0.446**	0.370	0.301	0.364	**0.060**
Twitter-AD	0.397	**0.924**	**0.438**	**0.466**	0.268
Luminol	0.374	0.763	0.428	0.430	0.195
SPOT	0.199	0.879	0.356	0.338	0.322
DSPOT	0.211	0.485	0.379	0.316	0.120
DONUT	0.023	0.032	0.029	0.026	0.004
SR	0.558	0.601	**0.556**	0.563	**0.023**
SR-CNN	**0.716**	**0.752**	0.464	**0.652**	0.128

Var indicates the standard deviation of the overall F_1-scores for the three classes

Experimental Results (cont.)

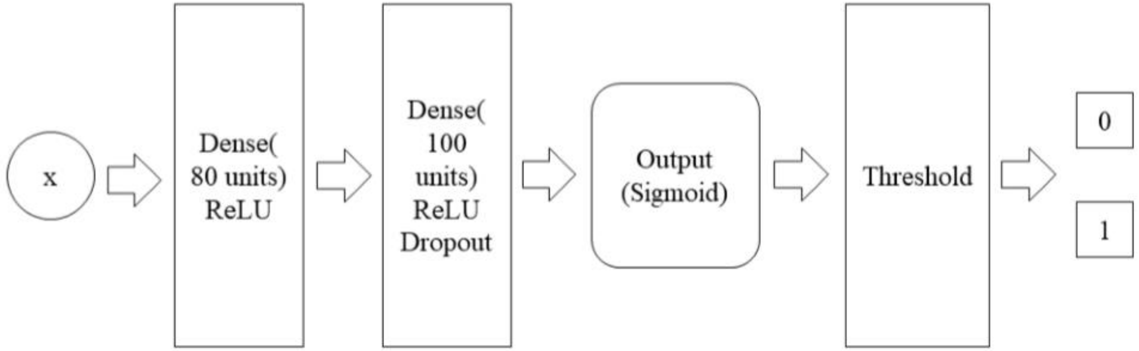
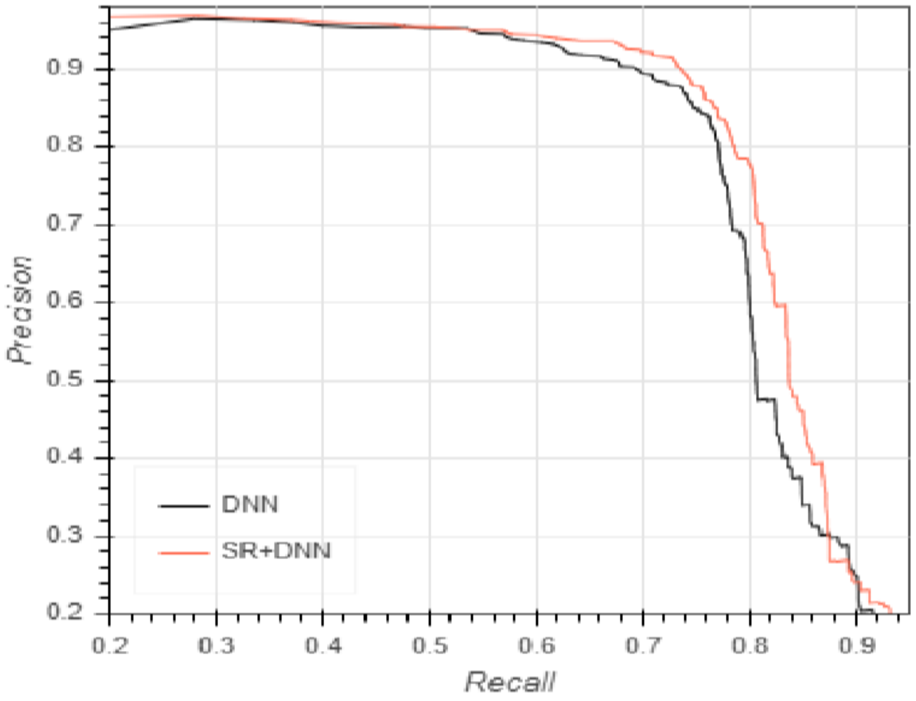


Table 7: Supervised results on KPI dataset

Model	F_1-score	Precision	Recall
DNN	0.798	0.849	0.753
SR+DNN	**0.811**	0.915	0.728



Production Impact

- 10% of online traffic has been run by SR and gained 37.9% F1-score improvement on labeled online DSAT
- In May, SR has been added in ML.net
- In June, SR has been enabled in Cognitive Services
- SR-CNN open-sourced on GitHub

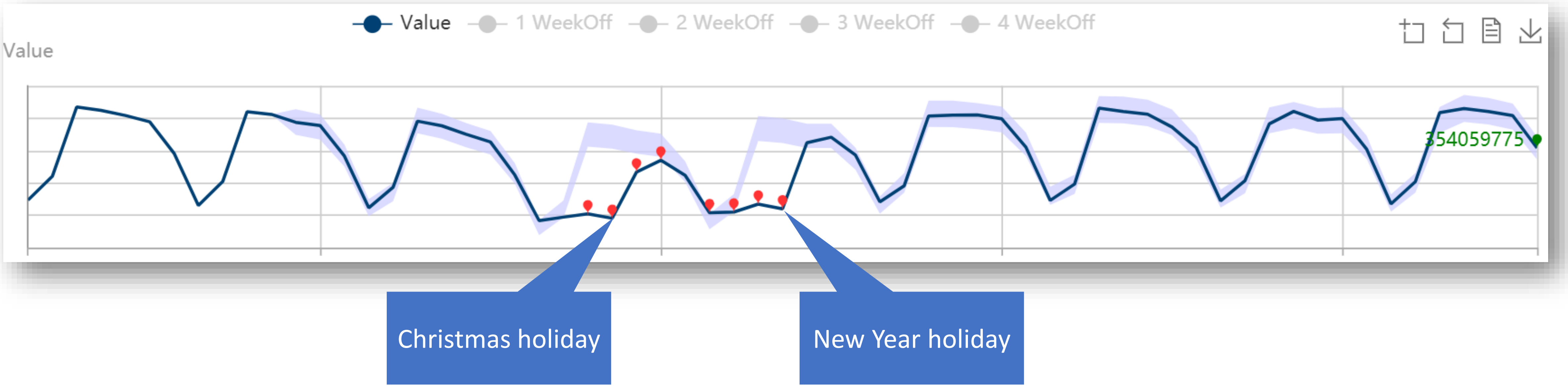
Smart alert

Anomaly ≠ Alert

Anomaly is objective by metrics history pattern and algorithm

Alert is complicated and subjective to multiple variates(criteria, holiday…)

Example of anomaly vs. alert



Real anomalies?

Yes

Need to fire alerts?

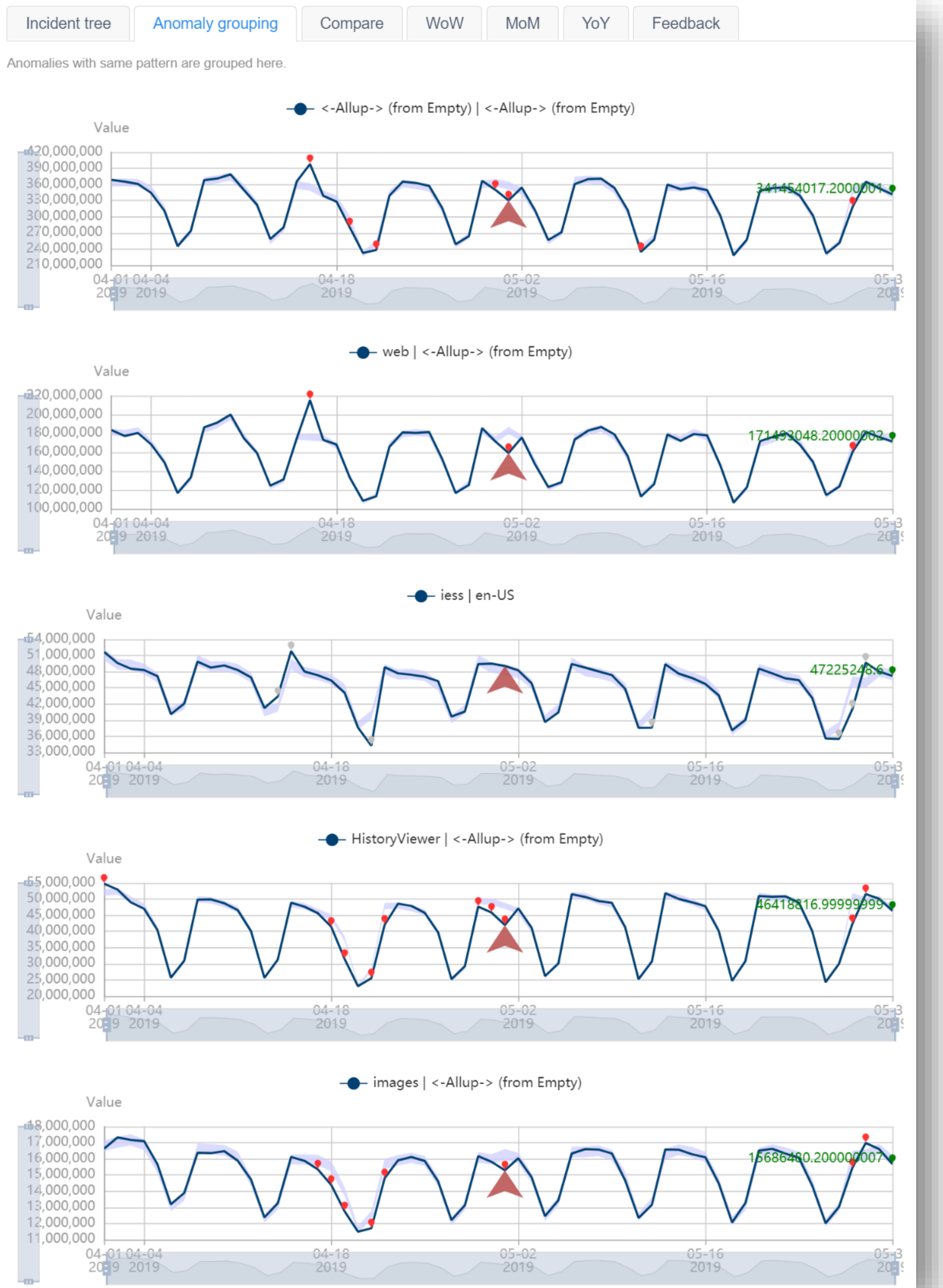
???

Rich diagnostic (not part of the service – just things you can do with the output results)

Diagnose with insights

Similar anomalies clustering

- Using **clustering** to correlate similar anomalies
- Integrate **K-means** and **hierarchical clustering** for both scalability and accuracy



References

Check out the overview and documentations of the API service

- Anomaly Detector: https://aka.ms/anomalydetector
- Technical docs: https://aka.ms/addoc
- Best practices: https://aka.ms/adbest
- Algorithm video https://www.youtube.com/embed/ERTaAnwCarM

Open-source code

- KDD 2019: https://www.kdd.org/kdd2019/accepted-papers/view/time-series-anomaly-detection-service-at-microsoft
- SR in ML.NET
- SR in Python: https://github.com/microsoft/anomalydetector

Try out the service

- Azure Notebook: https://aka.ms/adnotebook
- Create Anomaly Detector resource: https://aka.ms/adnew

Contact us:

anomalydetector@microsoft.com

https://techcommunity.microsoft.com/t5/AI-Customer-Engineering-Team/Introducing-Azure-Anomaly-Detector-API/ba-p/490162

Thanks

Q&A

jim.williams@microsoft.com