Realtime Time Series Anomaly Detection in Production

Shreya Khurana
Data Scientist - Intuit

Intro

Intuit Identity Analytics Team:
- Derek Schwartz, Manager 3 Data and Analytics
- Aaron Walker, Principal Technical Data Analyst
- Jacob langley, Staff Technical Data Analyst

Data Science Team:
- April Liu, Manager 2 Data Science

My team

What I do

Data Scientist at Intuit
Bay Area, California, USA

I work on the anomaly detection capability that tracks authentication and business health metrics at Intuit.

Previously I was building deep learning models to improve domain name recommendations at GoDaddy.

I have a masters in statistics from UIUC.

I'm a Python enthusiast and enjoy sharing my learnings with the community

What This Talk Is About

Realtime Time Series

Anomaly Detection

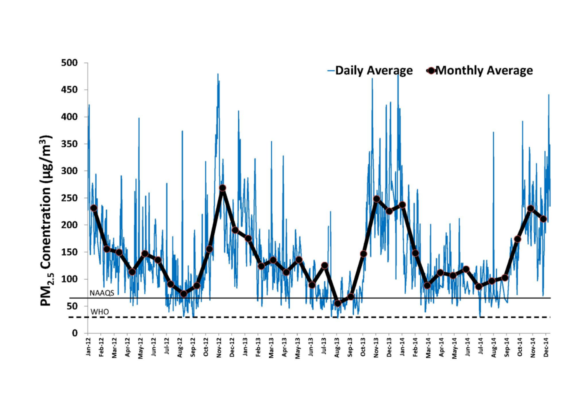
in Production

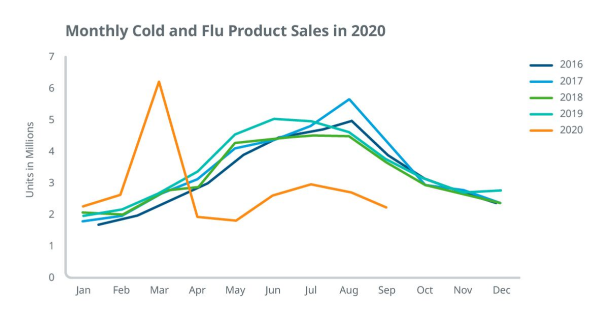
Time Series Data and Applications

Time series data: data measuring the same thing over a period of time

Exhibit some behavior like trend or seasonality that can be modeled statistically.

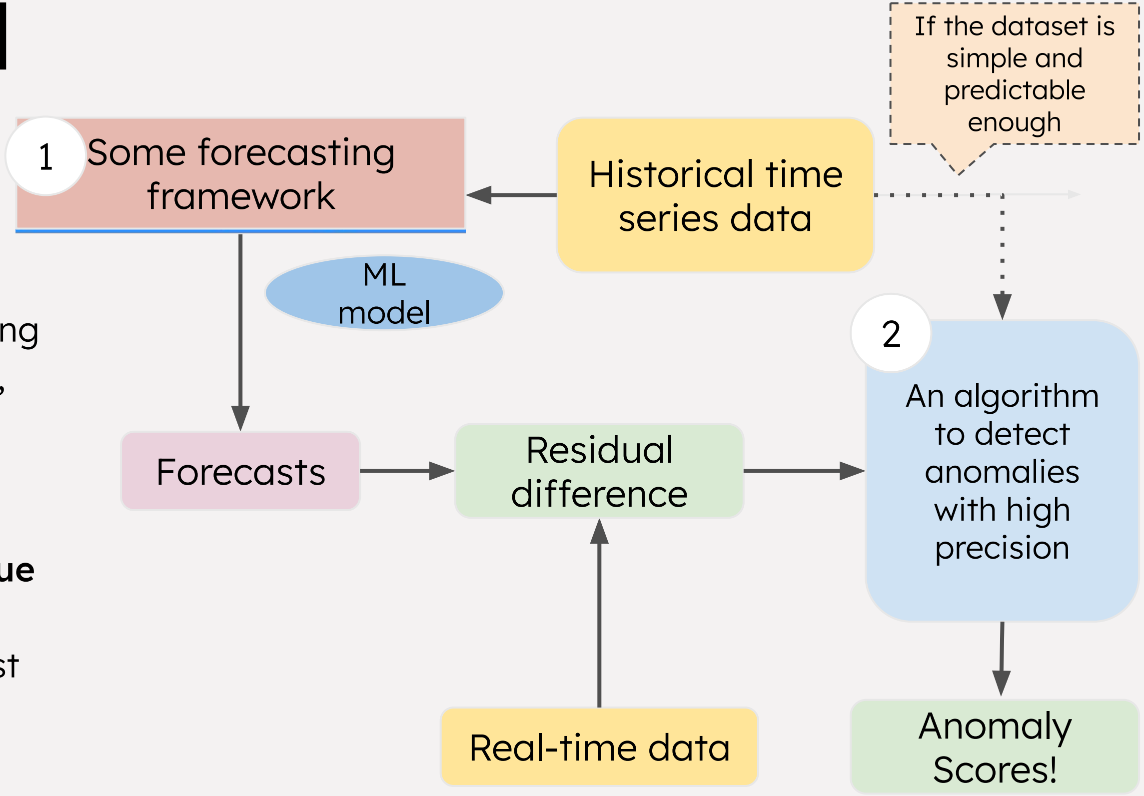
- Marketing / sales
- Weather
- KPI monitoring
- Data health checks
- Predictive maintenance
- Any monitoring system



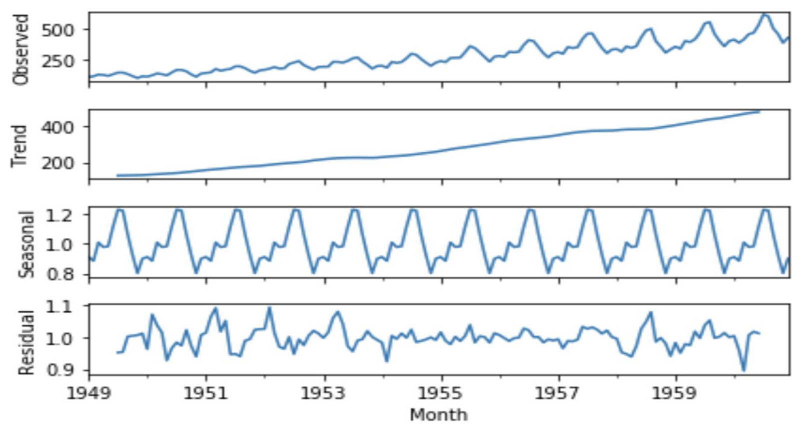


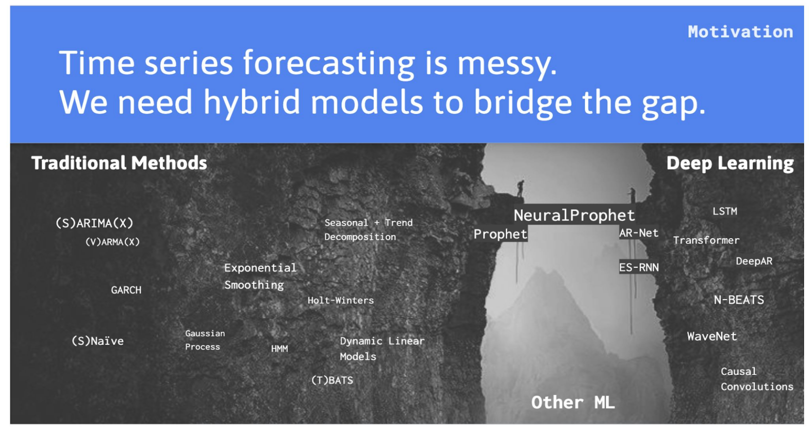
Anomaly Detection on Time Series Data

- Anomaly is when something is different that the expected or the usual
- Start simple with threshold based alerting
- If your time series is more complicated, then:
    - Assume your future data will behave *somewhat* similar to past
    - Forecast to get the **expected value**
    - Alert if the incoming data is different enough from the forecast
    - Need two core pieces



NeuralProphet: A Time Series Modeling Framework





Classical time series models had a limitation with modeling long-range dependencies

NeuralProphet is a scalable framework that uses a neural network to model autoregression

NeuralProphet Demo

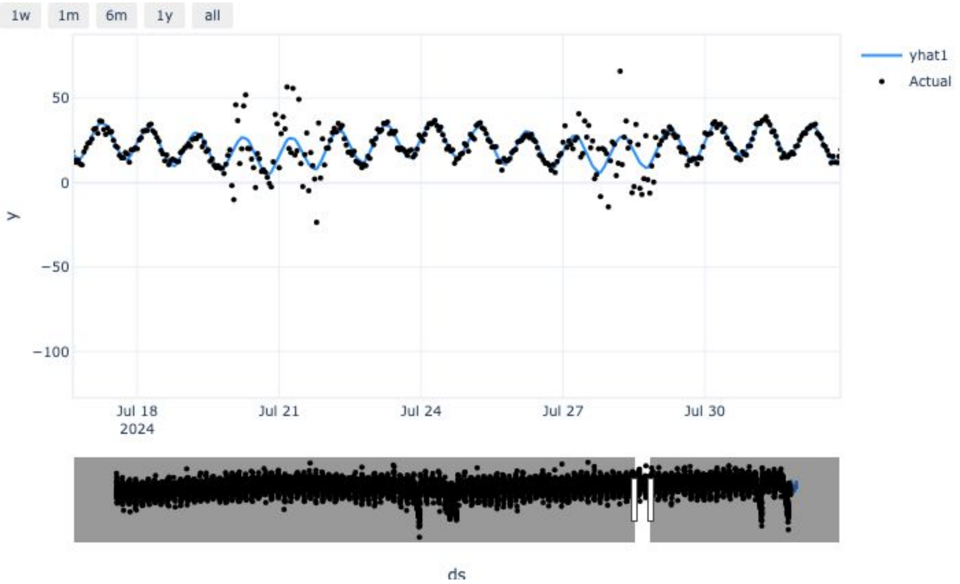
Model fitting and prediction

Component Decomposition

Modeling Custom Events

Tuning Model Parameters

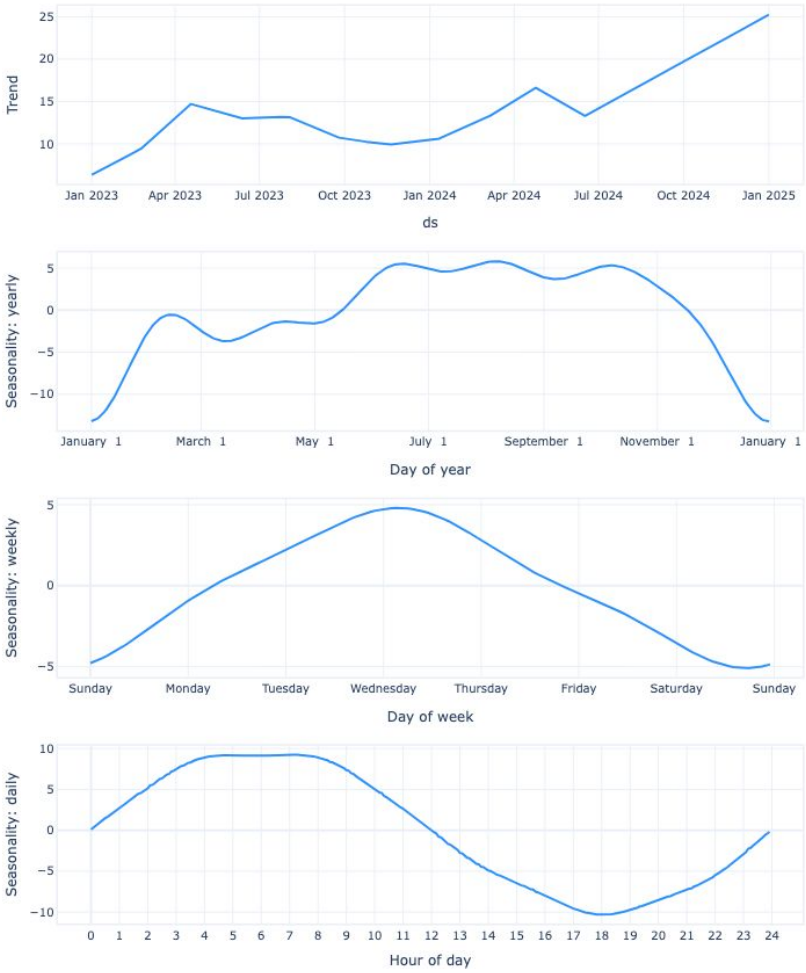
```
from neuralprophet import NeuralProphet

m = NeuralProphet()

# Fit the model on the dataset (this might take a bit)
metrics = m.fit(df)

# Create a new dataframe reaching into the future for our forecast,
n_historic_predictions also shows historic data
df_future = m.make_future_dataframe(df, n_historic_predictions=True, periods=24*7)

# Predict
forecast = m.predict(df_future)

# Visualize the forecast
fig = m.plot(forecast)
fig.show()
```



NeuralProphet Demo

Model fitting and prediction
Component Decomposition
Modeling Custom Events
Tuning Model Parameters

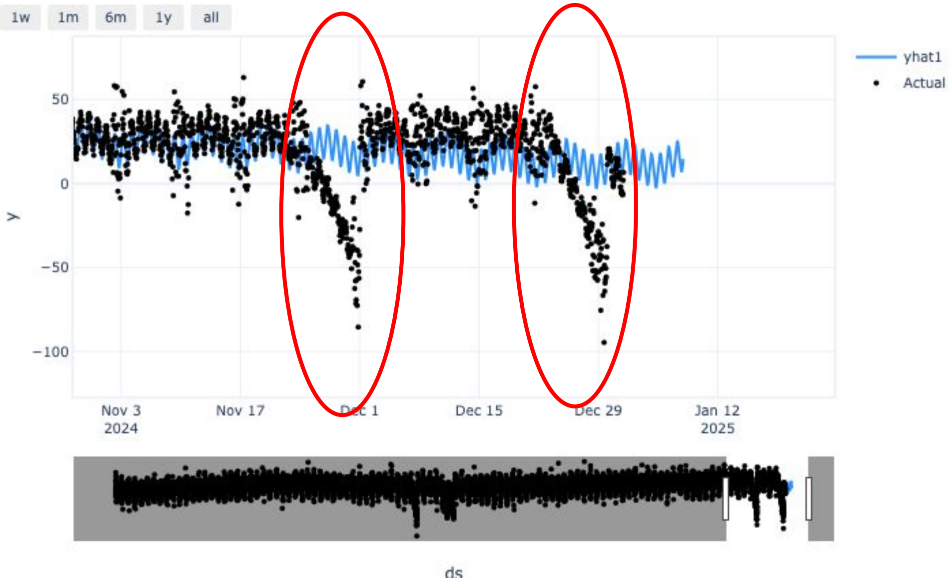
```
fig = m.plot_parameters()
fig.show()
```



NeuralProphet Demo

Model fitting and prediction
Component Decomposition
Modeling Custom Events
Tuning Model Parameters

Before:



Observe the model doesn't fit very well during holiday periods by default.

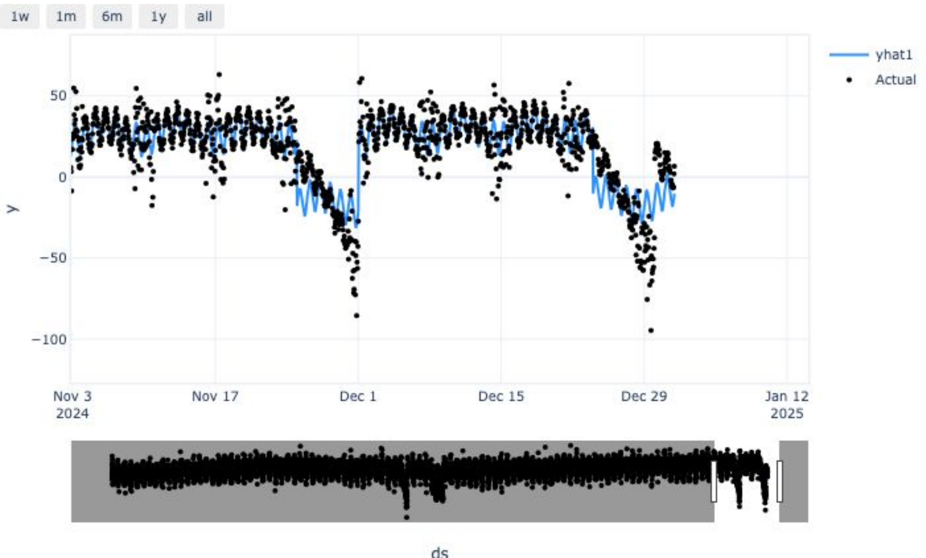
We need to add this signal to the model explicitly.

NeuralProphet Demo

Model fitting and prediction
Component Decomposition
Modeling Custom Events
Tuning Model Parameters

```
holiday_weeks = {
  "Thanksgiving": ["2023-11-20/2023-11-26", "2024-11-25/2024-12-01"],
  "Christmas": ["2023-12-23/2023-12-29", "2024-12-24/2024-12-30"],
  "NewYear": ["2023-12-30/2024-01-05", "2024-12-30/2025-01-05"]
}

df_events = pd.DataFrame()
for holiday, periods in holiday_weeks.items():
  for period in periods:
    start_date, end_date = (pd.to_datetime(p) for p in period.split('/'))
    df_events = pd.concat([df_events, pd.DataFrame({"event": "holiday", "ds":
pd.date_range(start_date, end_date, freq="h")})])

m.add_events("holiday")

df_all = m.create_df_with_events(df, df_events)

metrics = m.fit(df_all)
forecast = m.predict(df_all)
```

After:

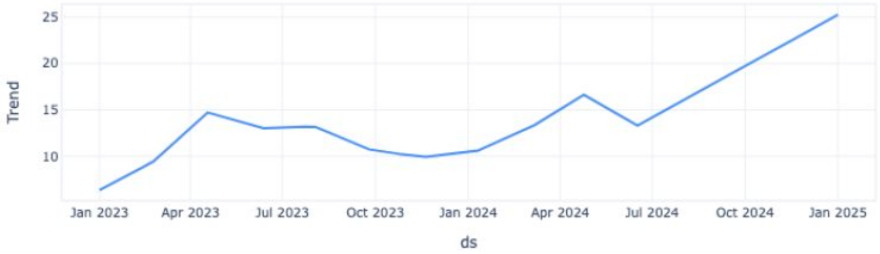


NeuralProphet Demo

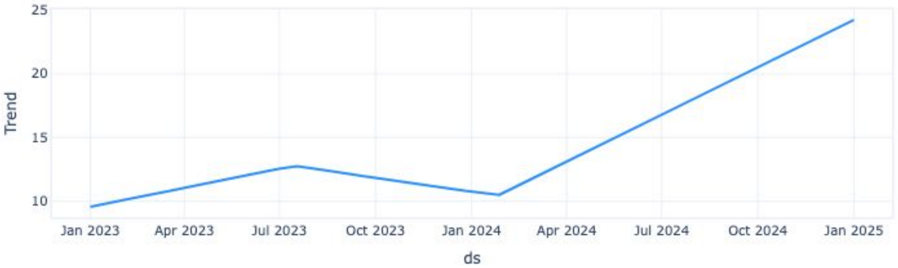
Model fitting and prediction
Component Decomposition
Modeling Custom Events
Tuning Model Parameters

Smart defaults for parameters and hyperparameters, but can be manually overridden

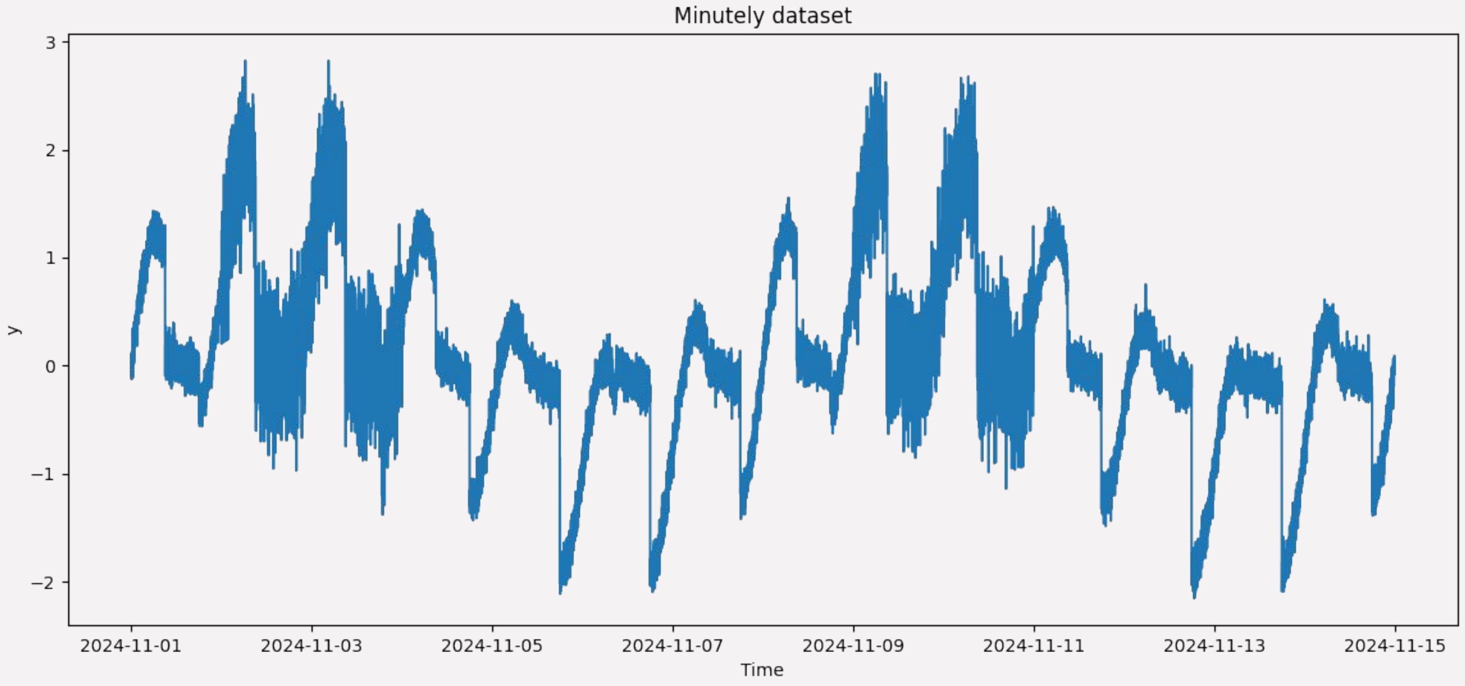
```
fig = m.plot_parameters(components=["trend"])
fig.show()
```




```
m = NeuralProphet(n_changepoints=2)
m.fit(df)
fig = m.plot_parameters(components=["trend"])
fig.show()
```



Anomaly Detection



Z-scores

Use z-scores to determine the extent of an anomaly i.e. how far a data point is from the average.

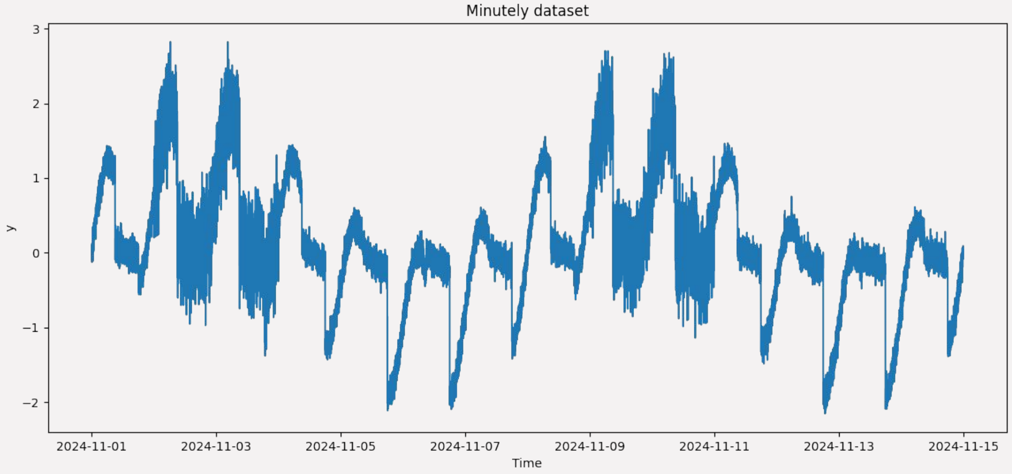
Mean based	
Mean of the distribution from some period	$$z = \frac{X - \mu}{\sigma}$$
Measure spread of the distribution through **standard deviation**	

Lookback Period

Period which you "look back upon" to find how the non-anomalous distribution should look like

Calculate stats like averages and spread of the distribution

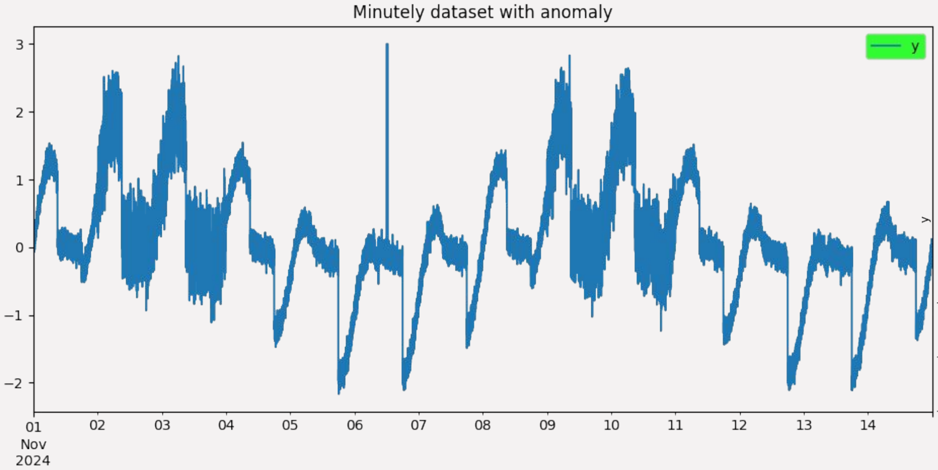
- What makes most sense?
- Experiment experiment experiment!
- Try multiple windows and domain knowledge
- Latency!
- Example: think about seasonality and hour of day and day of week

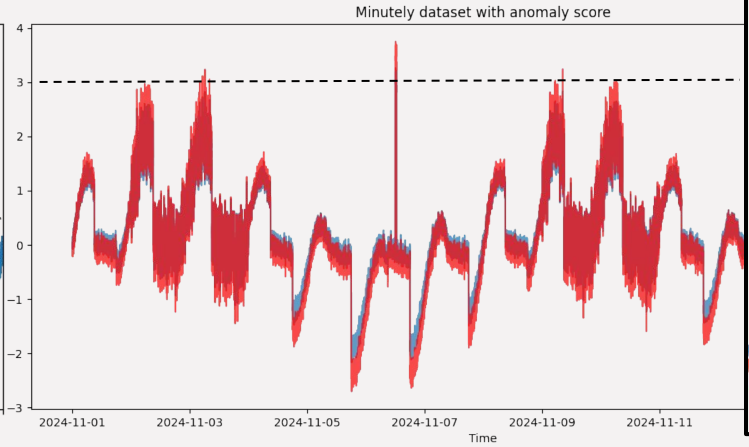


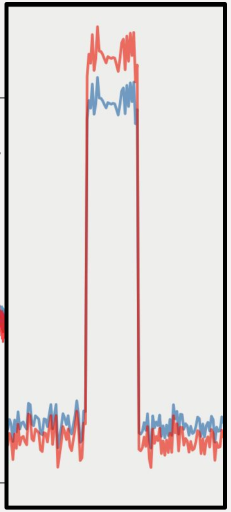
Lookback Period

Using the entire history

```
mean = minutely_df['y'].mean()
std = minutely_df['y'].std()
minutely_df['z_score'] = (minutely_df['y'] - mean) / std
```



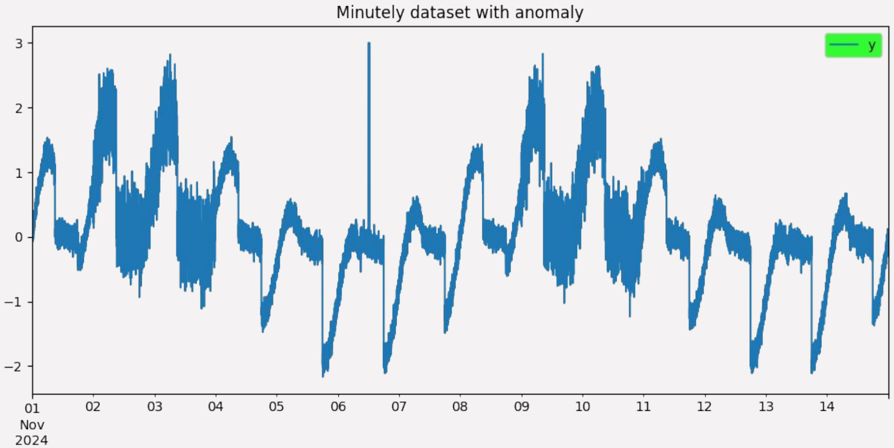


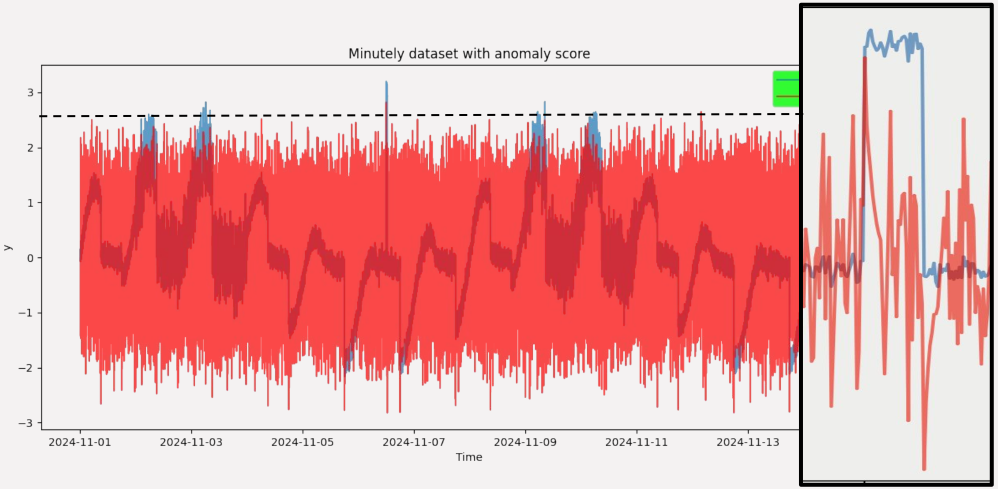


Lookback Period

Using last 10 minutes

```
minutely_df['rolling_mean'] =
minutely_df['y'].rolling(window=10).mean()

minutely_df['rolling_std'] =
minutely_df['y'].rolling(window=10).std()

minutely_df['z_score'] = (minutely_df['y'] -
minutely_df['rolling_mean']) / minutely_df['rolling_std']
```

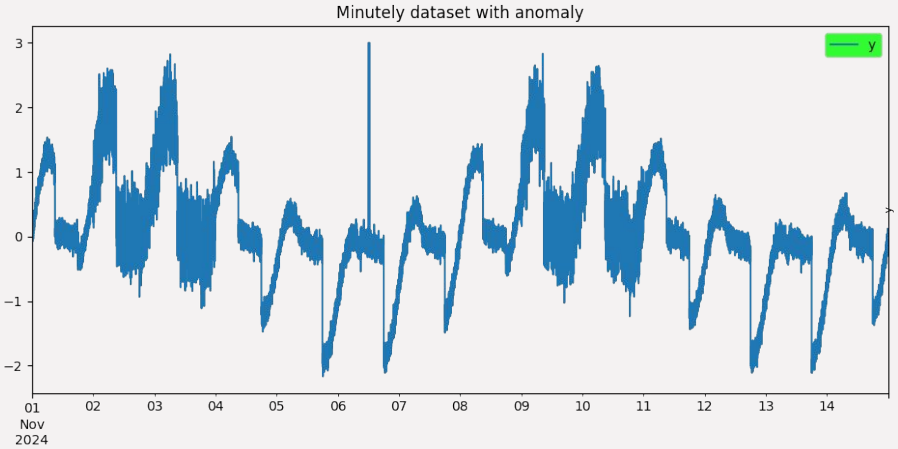


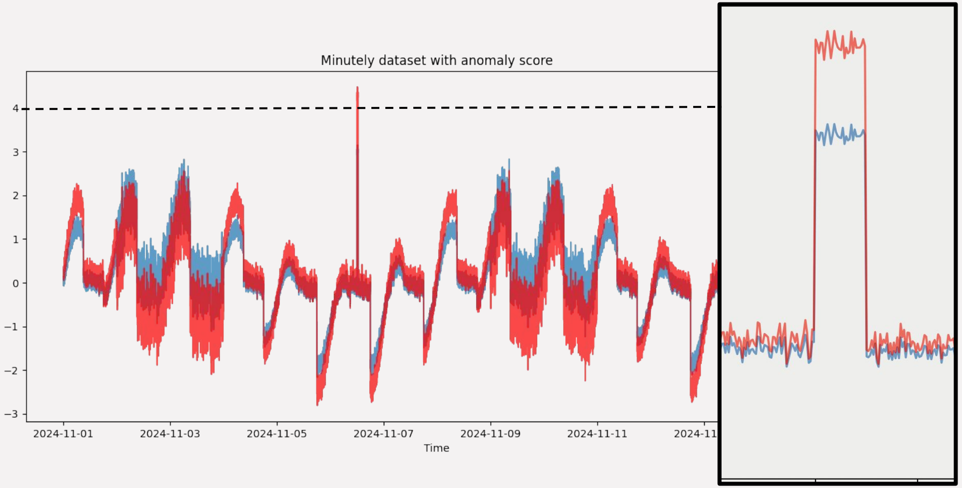


Lookback Period

Partitioning weekdays and weekends

```
minutely_df['weekend'] = minutely_df['ds'].dt.dayofweek >= 5

weekend_mean = minutely_df[minutely_df['weekend']]['y'].mean()
weekend_std = minutely_df[minutely_df['weekend']]['y'].std()
weekday_mean = minutely_df[~minutely_df['weekend']]['y'].mean()
weekday_std = minutely_df[~minutely_df['weekend']]['y'].std()

minutely_df['z_score'] = np.where(minutely_df['weekend'],
                        (minutely_df['y'] - weekend_mean) / weekend_std,
                        (minutely_df['y'] - weekday_mean) / weekday_std)
```





Z-scores

Use z-scores to determine how far a data point is from the average.

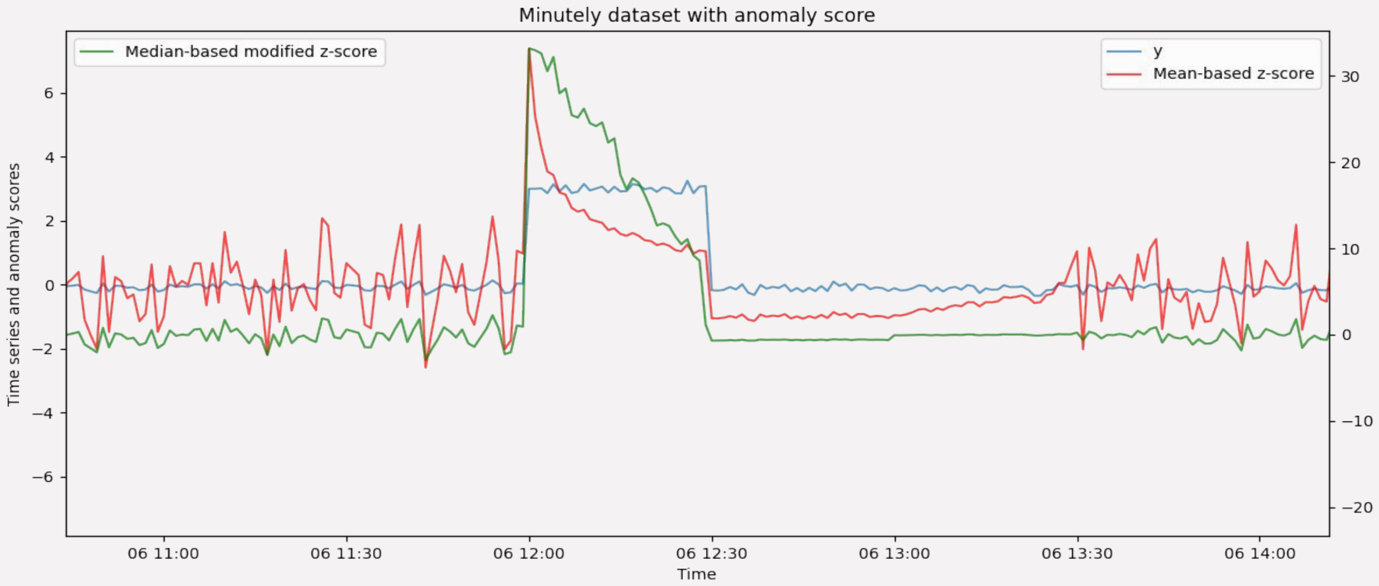
- **An “average” can be defined in multiple ways**
- **We also need a way to measure spread of the data, to put this distance in context**

Mean based	Median based
Mean of the distribution from some period	**Median** of the distribution from some period
Measure spread of the distribution through **standard deviation**	Measure spread of the distribution through **median absolute deviation**

Z-scores

Use z-scores to determine how far a data point is from the average.

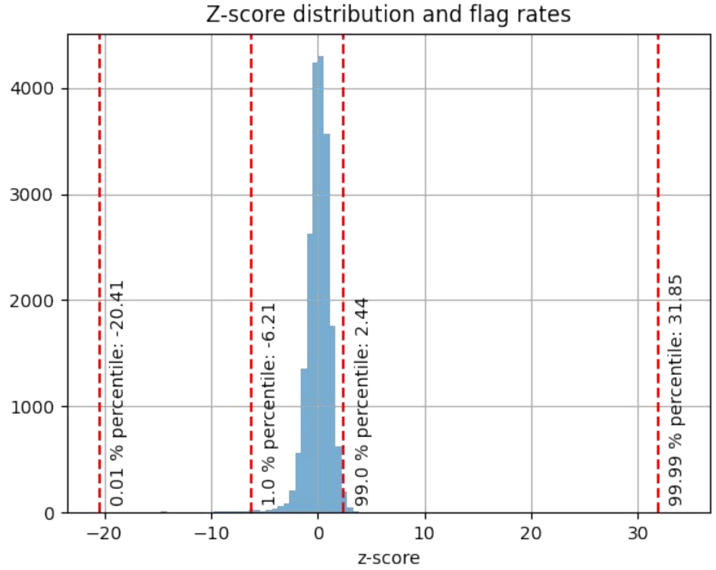
- **An "average" can be defined in multiple ways**
- **We also need a way to measure spread of the data, to put this distance in context**



Tuning anomaly scores

How do we know if the data is “far enough” to call it anomalous?

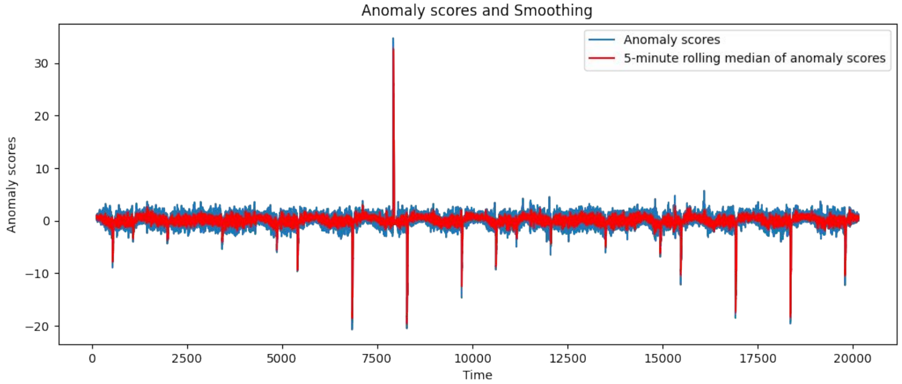
- If you have ground truth of anomalies, amazing!
- If not:
  - Visual inspection
  - Flag a given percentile (say, 0.001% of the data)
  - Flag and run for some time
  - Measure precision/recall

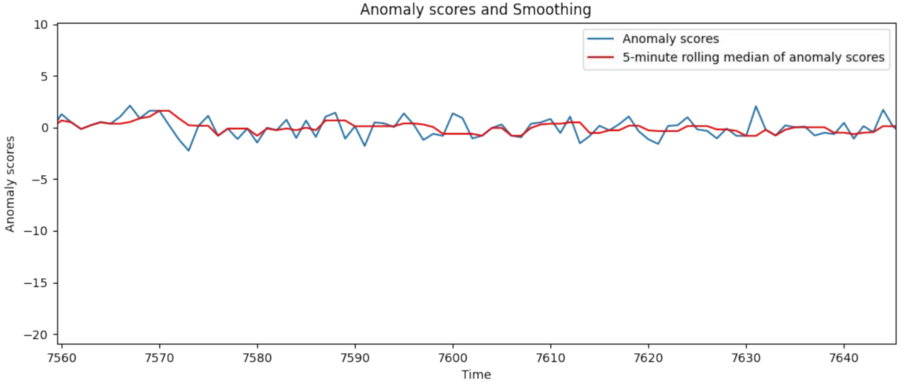


Tuning anomaly scores

How do we set thresholds in case of a volatile time series and volatile z-scores?

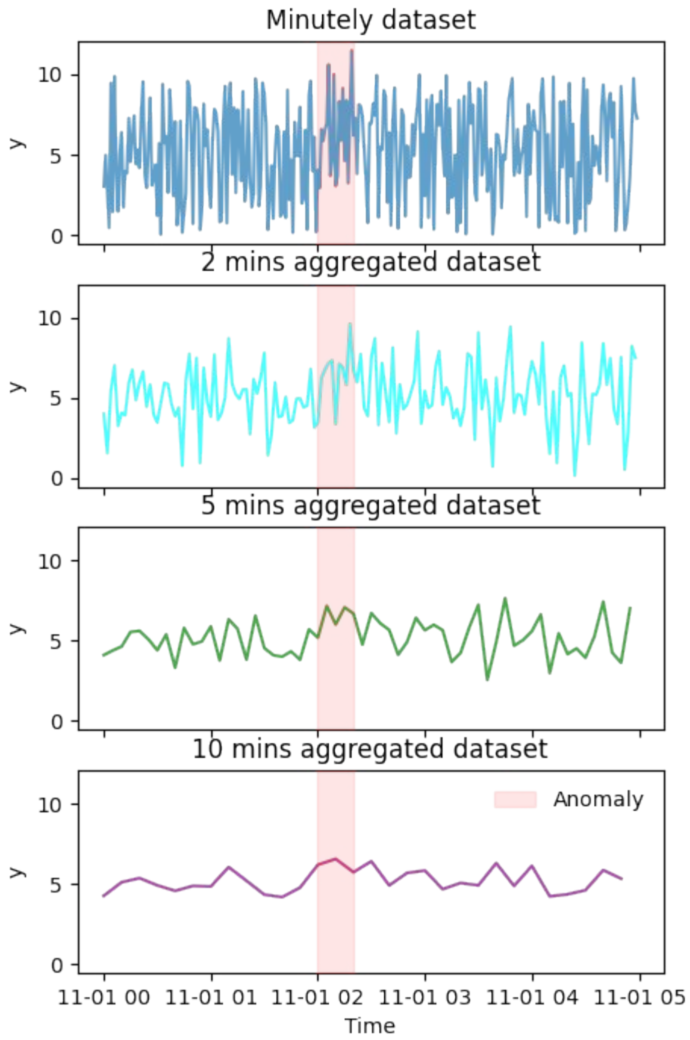
- Smoothing of scores
  - Average anomaly scores over a period of time
- Accuracy ve time-to-detect latency





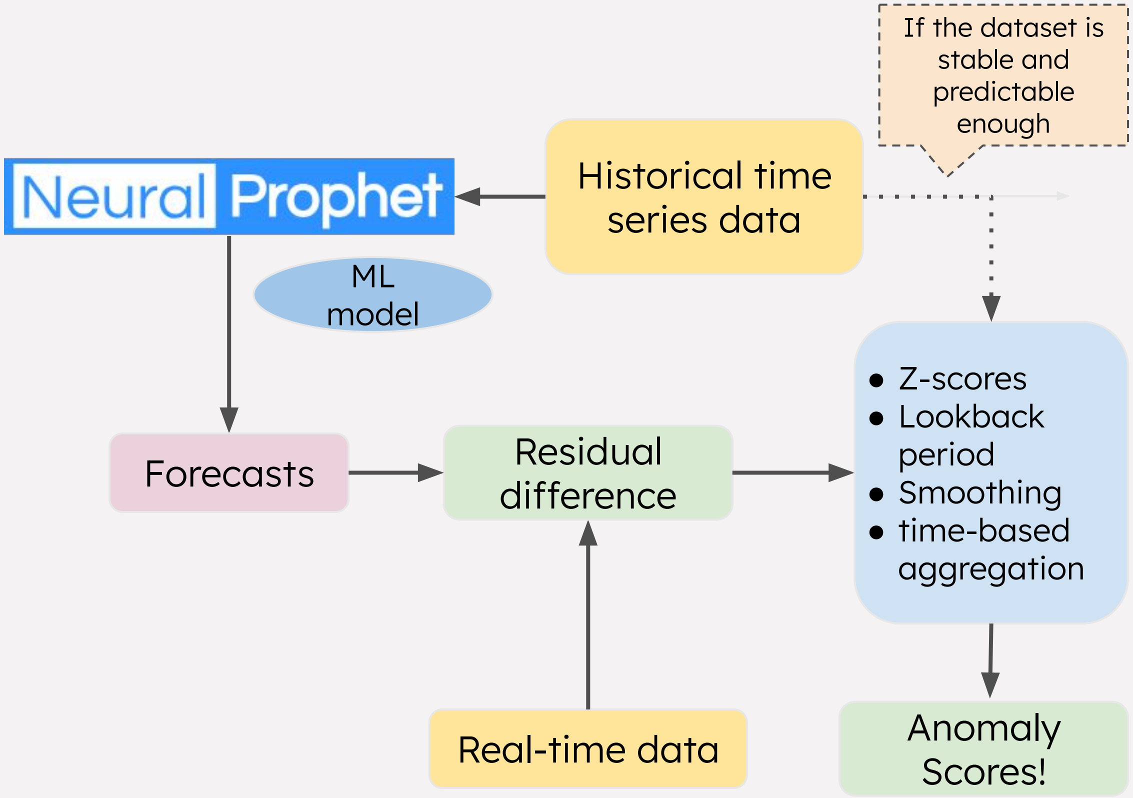
Considerations for sparse or volatile time series

- Impute values if possible
- Aggregate into time windows
- Discrete or rolling time bucketing
- Trade-offs with latency



Bringing it all together

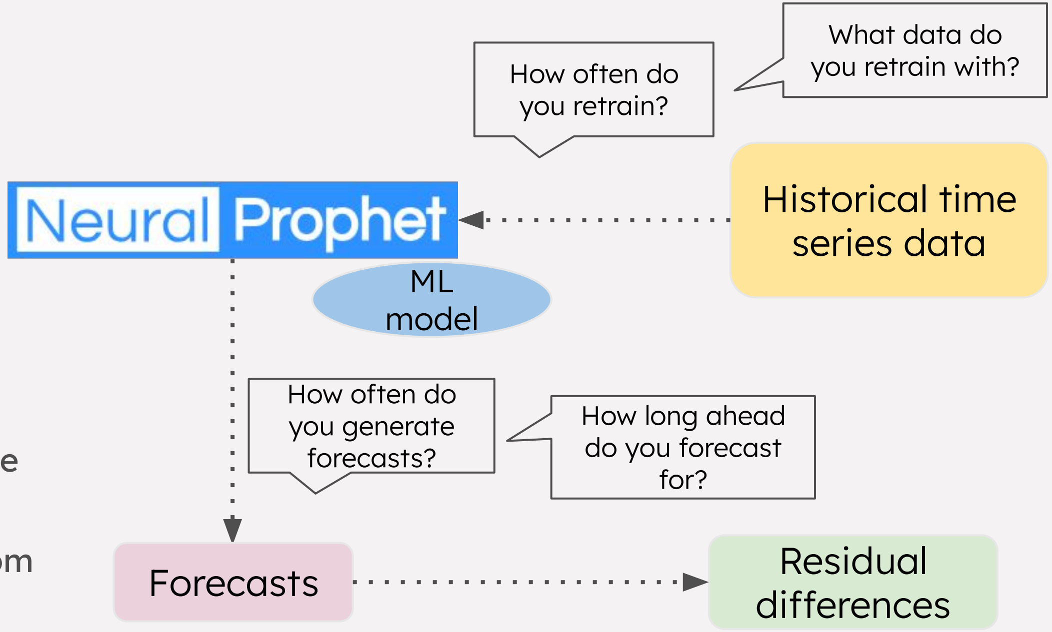
- Start simple using basic z-scores and lookback period on the time series data
- If your time series is more complicated, then use forecasts to find the expected value
- Instead of modeling the time series data, model residuals
- And do all the statistical experimentation with residuals



Building a Realtime Pipeline

- Model development life cycle
- Realtime data is continuously populating your systems

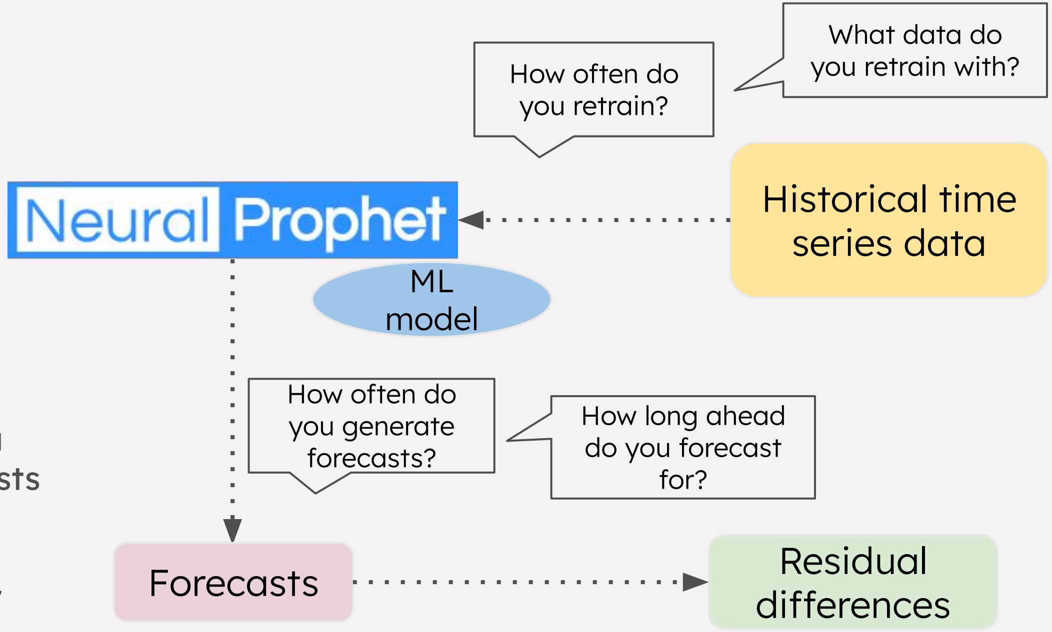
1. **Save your data from anomalies**
    - Anomalous data WILL creep in hence, use robust metrics
    - Label and filter out anomalous period from the data to retrain ASAP
    - OR remove confidently predicted anomalies



Building a Realtime Pipeline

- Model development life cycle
- Realtime data is continuously populating your systems

2. Forecast reasonably to avoid bias
  - NeuralProphet will use ALL historical data and the trained model to generate forecasts
  - Forecast frequently to incorporate recent behavior
  - But try to avoid using anomalous data for prediction
  - OR remove confidently predicted anomalies from retraining/forecasting



Thank you!

Questions?

Talk contents:
https://github.com/ShreyaKhurana/pydata-global-2024

References:
https://neuralprophet.com/contents.html
https://www.statology.org/modified-z-score/

Email

shreyakhurana11235@gmail.com

LinkedIn

https://www.linkedin.com/in/shreya-khurana/

GitHub

@ShreyaKhurana