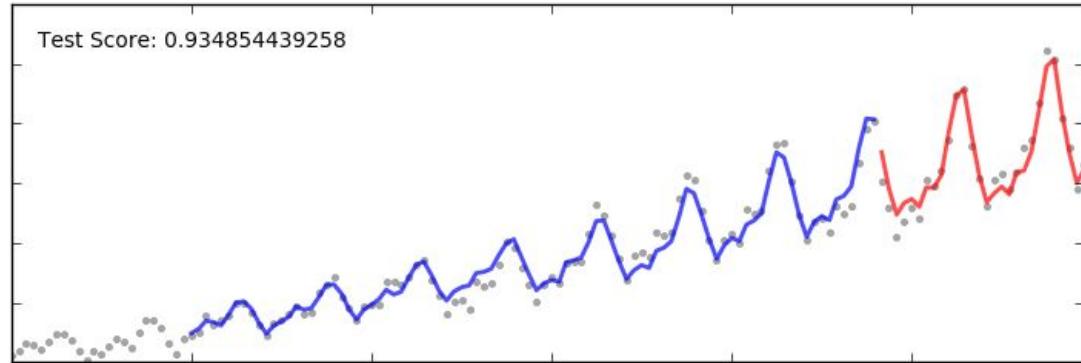


# Time Aware Machine Learning

Mark Steadman  
Viktor Kovryzhkin



# Many of the most valuable problems involve time

## Relentless Focus on ROI



**\$18M**

Healthcare system  
reduces patient stays  
and gets more  
efficient nurse staffing.



**\$200M**

Realized value by  
retailer for multiple  
SKU-level demand  
forecasting use  
cases



**\$400M**

Value to top 5 global  
bank across multiple  
use cases and models  
in production



# What is Time Aware Machine Learning?

# Machine Learning Makes Some Key Assumptions

The main one that concerns us to start: the data generating process is stable

- Training / Validation sets have the same distribution if split randomly
- Training / Prediction sets have the same distribution

But this often does not hold

- Events can change the distribution
- Data can “drift” over time as the system evolves
- Data can “drift” over time as the data evolves

# Pre-Event And Post Event Can Be Quite Different



Updated Super Bowl Odds 🏆

[go.fanduel.com/NFL.Futures](http://go.fanduel.com/NFL.Futures)

## THIS YEAR'S CHAMPION

New England Patriots	7/1	Seattle Seahawks	32/1
Kansas City Chiefs	8/1	San Francisco 49ers	36/1
New Orleans Saints	9/1	Tennessee Titans	40/1
Los Angeles Rams	11/1	Carolina Panthers	50/1
Philadelphia Eagles	14/1	Indianapolis Colts	60/1
Los Angeles Chargers	15/1	New York Jets	60/1
Cleveland Browns	15/1	Tampa Bay Buccaneers	70/1
Chicago Bears	19/1	Denver Broncos	70/1
Green Bay Packers	19/1	New York Giants	80/1
Pittsburgh Steelers	19/1	Buffalo Bills	80/1
Dallas Cowboys	25/1	Detroit Lions	80/1
Houston Texans	30/1	Oakland Raiders	100/1
Minnesota Vikings	30/1	Washington Redskins	100/1
Jacksonville Jaguars	31/1	Arizona Cardinals	110/1
Atlanta Falcons	32/1	Cincinnati Bengals	150/1
Baltimore Ravens	32/1	Miami Dolphins	170/1

21+ AND PRESENT IN NJ OR PA  
GAMBLING PROBLEM? 1-800-GAMBLER

FANDUEL  
SPORTSBOOK

### Super Bowl LIV

NFL · 2/2 Final

20 - 31

San Francisco 49ers (15-4)      Kansas City Chiefs (15-4)

Super Bowl

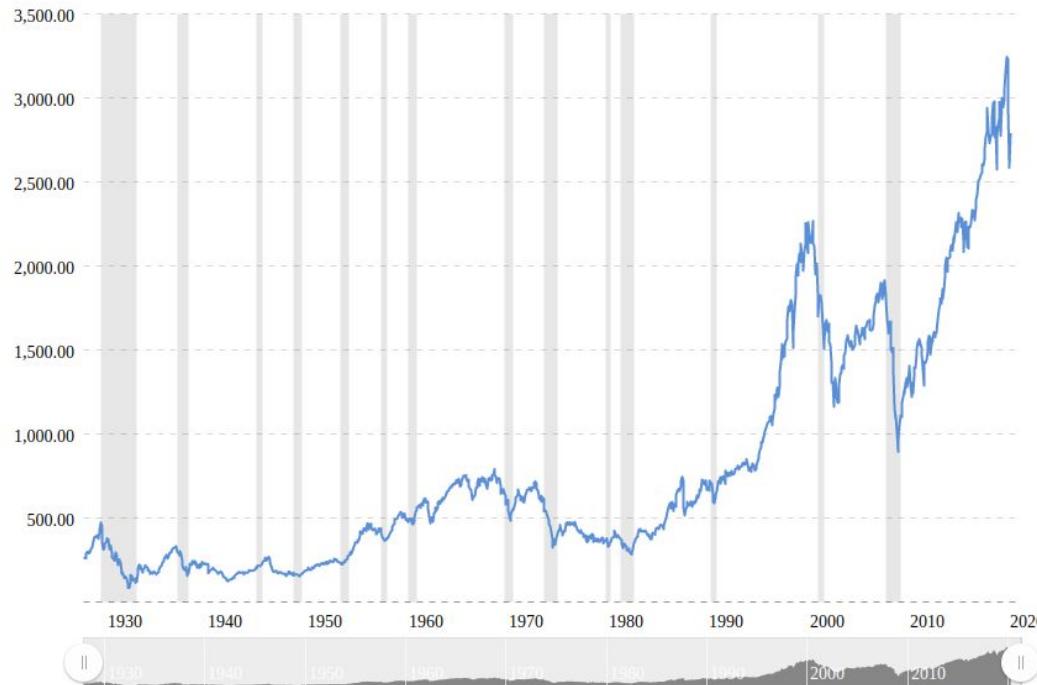
Team	1	2	3	4	T
San Francisco 49ers	3	7	10	0	20
Kansas City Chiefs	7	3	0	21	31

Game recap 13:33

Feedback

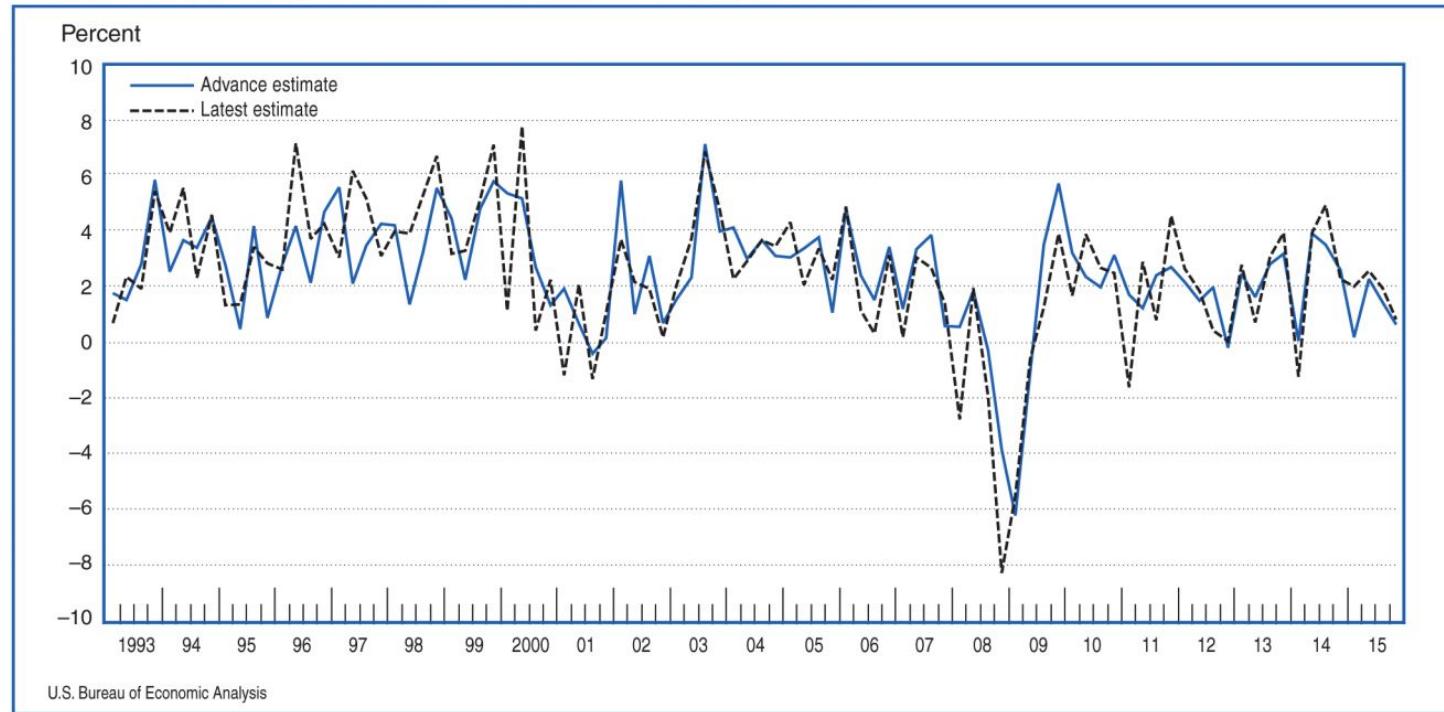
More about this game

# Your System Can Evolve Over Time



# Historical Data Can Update As Well

Chart 1. Percent Change in Real Gross Domestic Product, 1993–2015



# Dates and Times Pose Particular Challenges As Features

- Given “2020-04-01 23:09:12.004256” how to put it into your model?
  - Need to encode somehow
- They can cause overfitting even if encoded as they often behave like IDs
  - Knowing that any email that arrives at 2020-04-01 23:09:12.004256 is spam is often not helpful and doesn’t generalize
- There are a host of other consistency challenges
  - Daylight savings time?
  - What date is 04/01/2020?

# What is Time-Aware Machine Learning (TAML)

It is about performing ML in the following situations

- Date or time variables are important to the overall context of the problem
- You want to use date or time variables in your models

# Agenda

- Introduction to time series modeling
- Approach 1: Machine Learning + Feature Engineering
- Exercise
- Approach 2: Deep Learning
- Group exercise

# Time Series Exercise

---

## Prerequisites

- Python 3.6+ / 2.7 Jupyter Notebook
- numpy>=1.16, pandas>=0.24,  
scikit-learn>=0.20, matplotlib, IPython

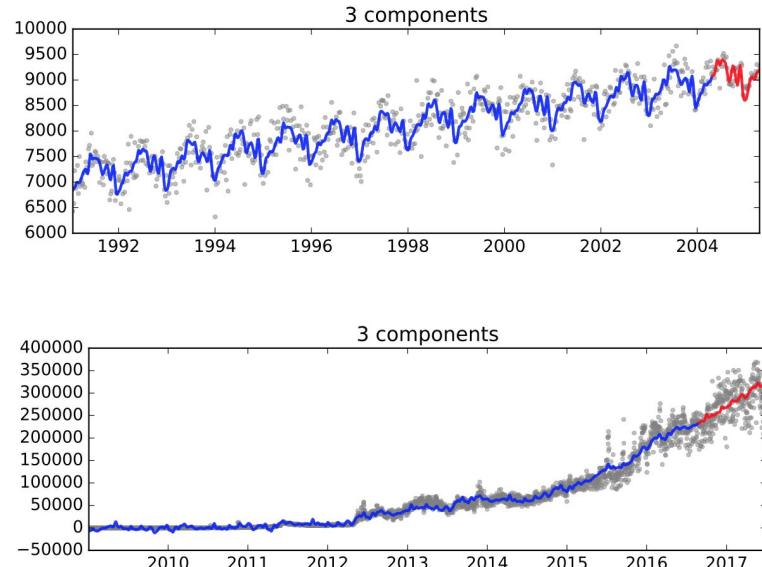
## Instructions

<https://github.com/vikua/time-series-workshop>

# What is a Time Series?

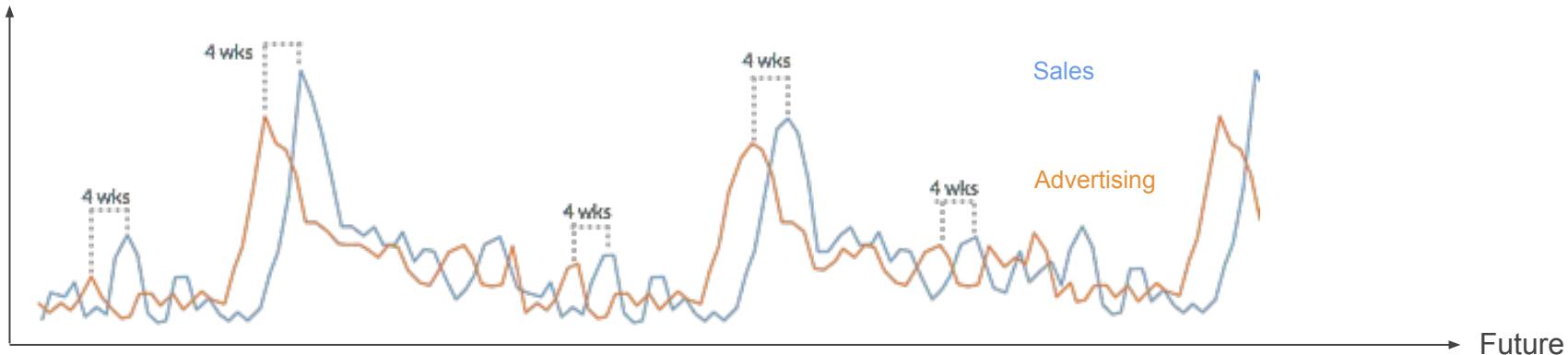
Predict future values based on past and present data

	A	B	C	D	E	F	G
1	Week	Sales	Online_Ad_Spend	TV_Ad_Spend	Flyer_Spend	Web_Total_Visits	Web_Unique_Visitors
2	1/6/2013	122759290		459742	208609.59	139779.87	
3	1/13/2013	122837816.9		455624.07	636317.34	108928.9	481396
4	1/20/2013	122330234.3		306977.25	459119.52	107133.97	386720
5	1/27/2013	106094594.8		222976.63	332685.78	102834.1	372462
6	2/3/2013	122340940.2		393393.04	252719.83	116269.11	495301
7	2/10/2013	130370810		405611.95	586970.65	102298.72	533990
8	2/17/2013	120857254.2		273239.64	474100.01	124366.22	362208
9	2/24/2013	123750229.4		465004.55	619069.52	111549.13	464649
10	3/3/2013	119568379.1		446871.76	203441.61	125994.14	329999
11	3/10/2013	129829219.8		383115.15	187667.57	106747.38	436931
12	3/17/2013	122635050.3		485634.16	688591.96	102836.61	571140
13	3/24/2013	130689274.2		386893.47	360485.8	105834.99	377121



Sequences as opposed to individual event data in time

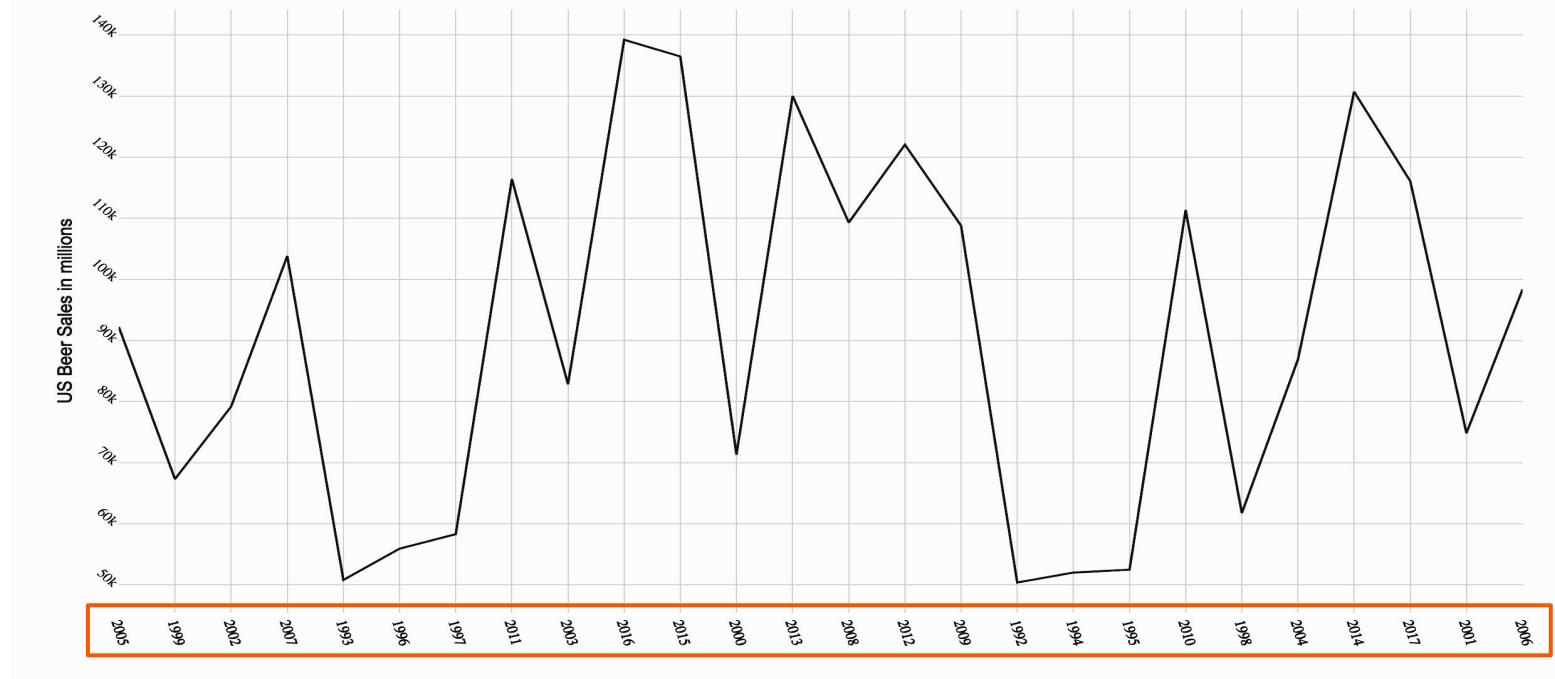
# Time Series Challenges



- Real data has delays (feature engineering required)
- Difficult to validate for future predictions
- Many different treatments (stationarity, log-transform, periodic)
- Many different algorithms (ARIMA, Exp Smoothing, Decompositions)
- Most machine learning algorithms (xgboost) today are not time-aware on their own

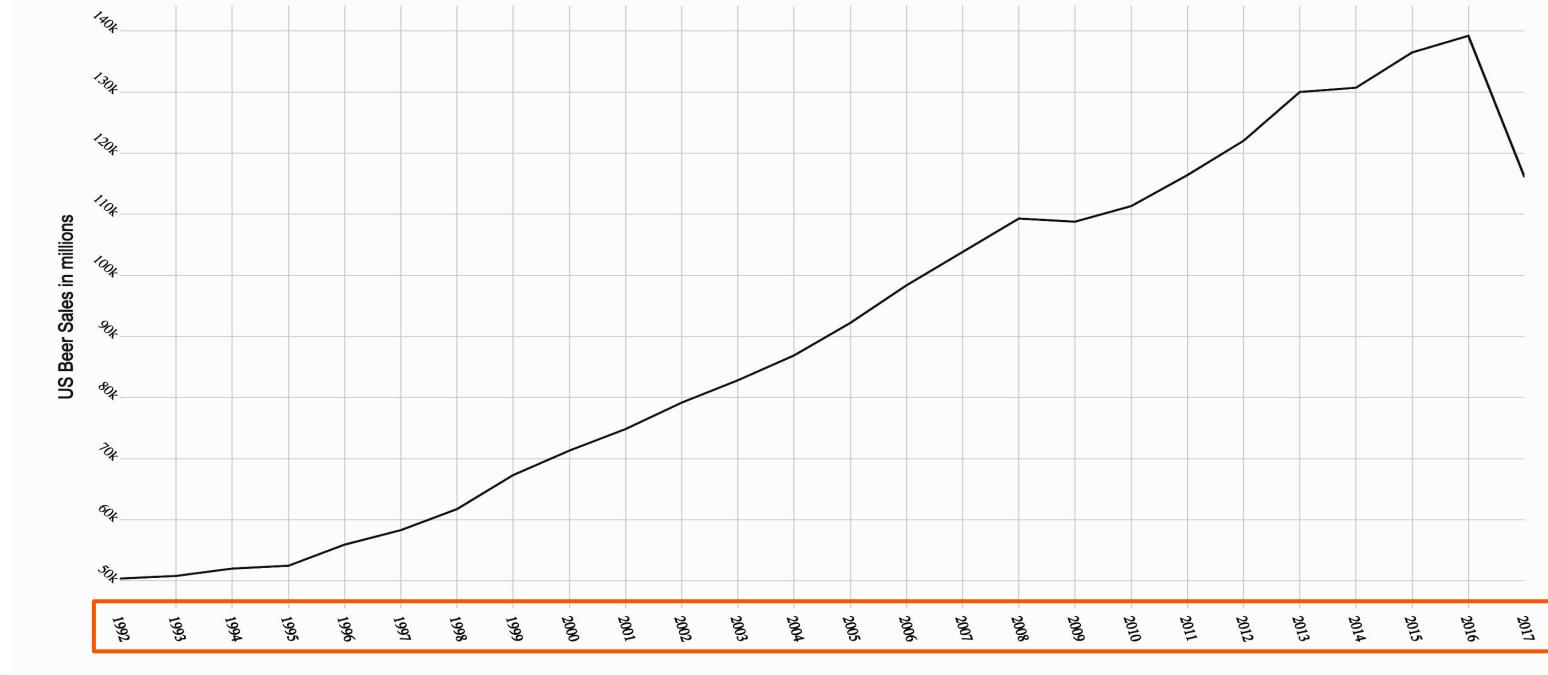
# Why are Time Series Problems Different? US Beer Sales

Common assumption #1: Order of rows doesn't matter



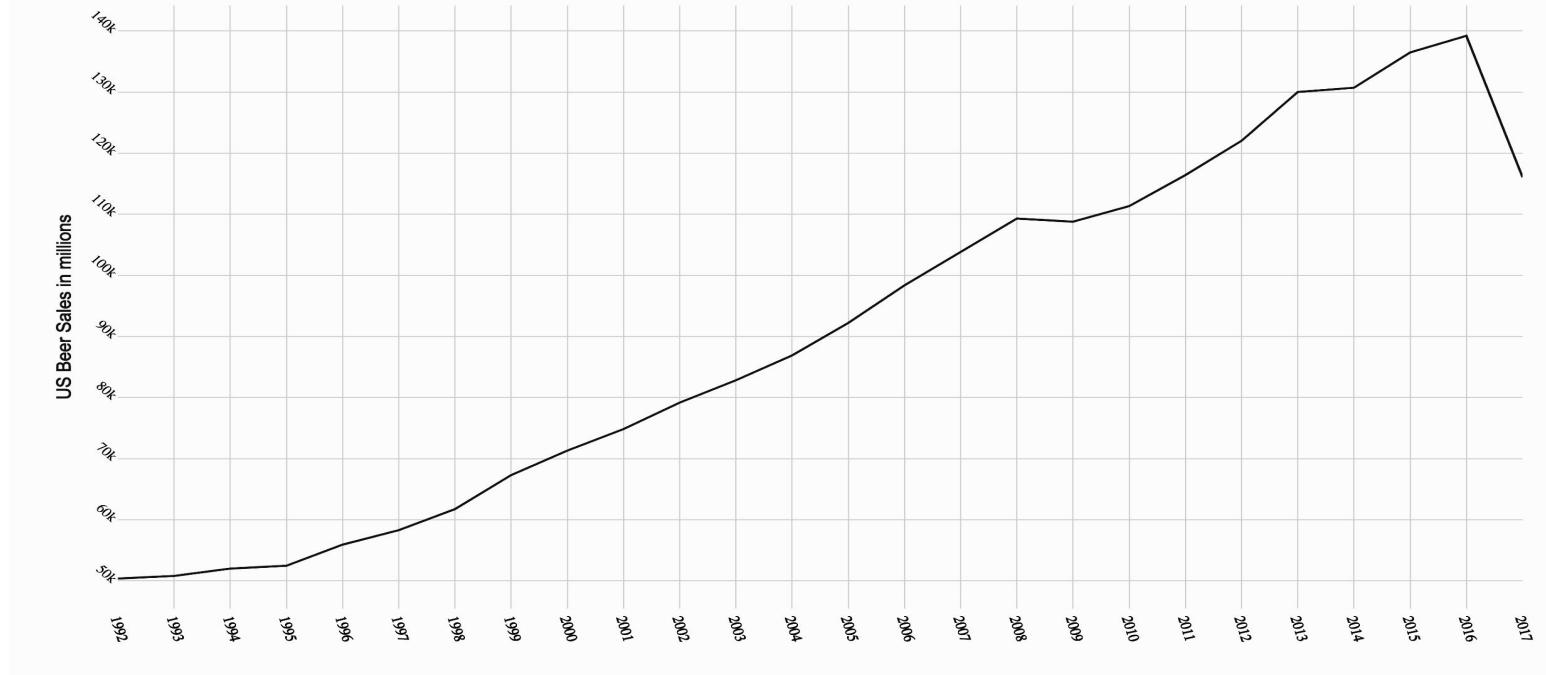
# Why are Time Series Problems Different? US Beer Sales

~~Common assumption #1: Order of rows doesn't matter~~

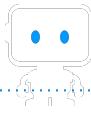


# Why are Time Series Problems Different? US Beer Sales

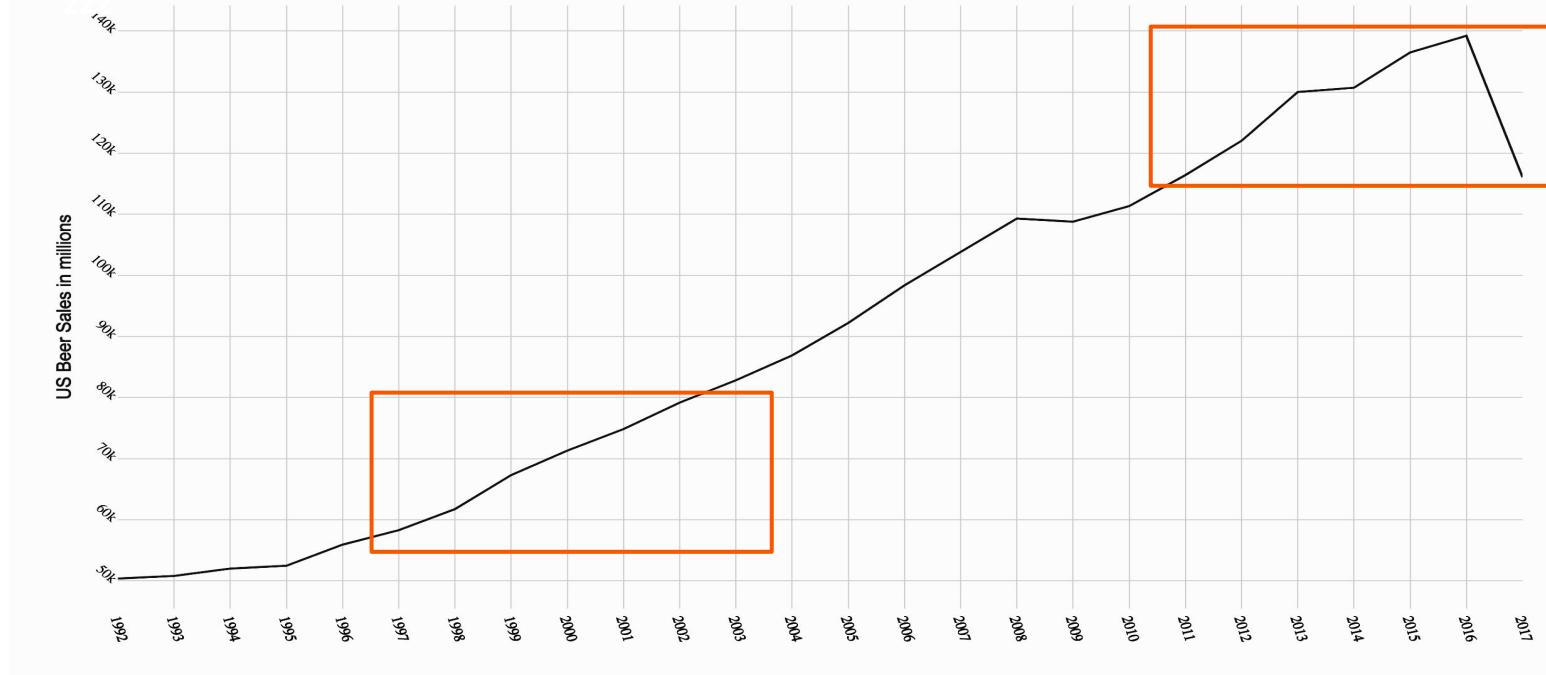
**Common assumption #2: Training and future data are similar**



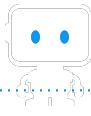
# Why are Time Series Problems Different? US Beer Sales



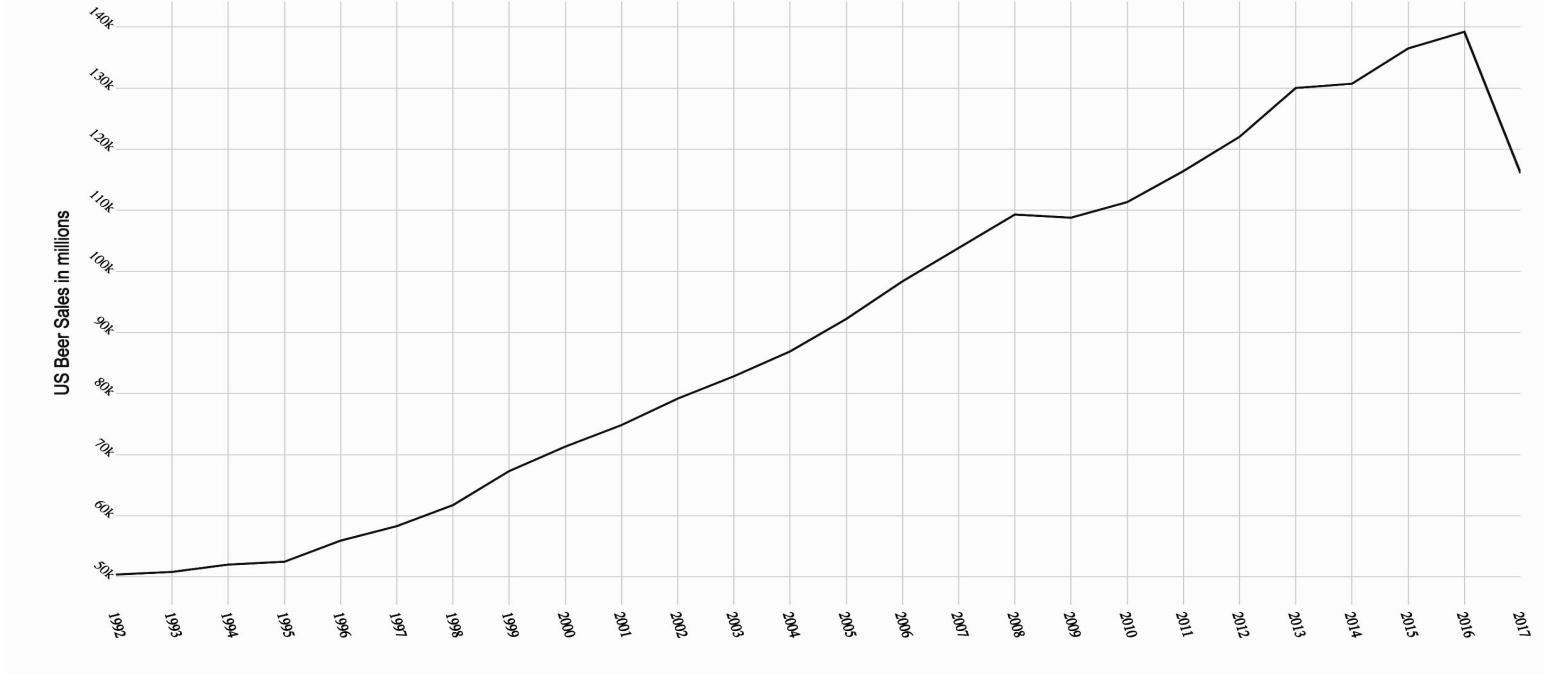
~~Common assumption #2: Training and future data are similar~~



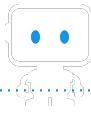
# Why are Time Series Problems Different? US Beer Sales



**Common assumption #3: Models can be evaluated with a single row of data**



# Why are Time Series Problems Different? US Beer Sales



~~Common assumption #3: Models can be evaluated with a single row of data~~



# Time Series Machine Learning

## Feature Engineering

- Automated lag selection
- Automated window statistics
- Rolling numeric, categorical, text information

## Target Transforms

- Stationarity, exponential, periodicity detection
- Automated differencing, offsets, link functions, log transformations
- Additive or multiplicative models

## Model Backtesting

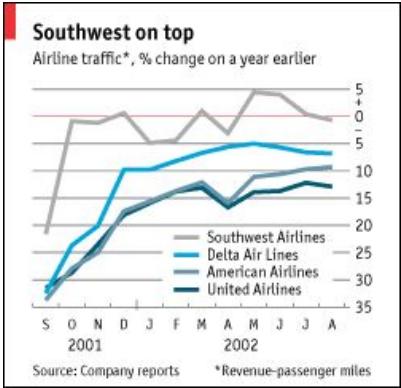
- Time-aware data partitioning and validation
- Automated or configurable backtesting strategies
- Refit on most recent data

## Modeling

- Classical time series models (ARIMA, ETS, decompositions)
- Time-aware xgboost, distance modeling, and other algorithms
- Deploy to dedicated prediction service

# Types of Time Series Problems

## Strategic



## Operational

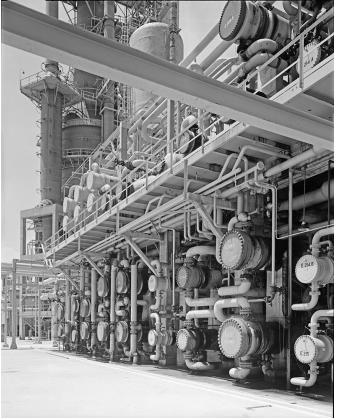


- Few series
- Traditional approaches often due well
- Sufficient History
- Longer refresh periods

- Many series
- Insufficient history → routes start and stop
- Shorter refresh periods → need a forecast every day

# Two Types of Complexity in Machine Learning

## The Refinery



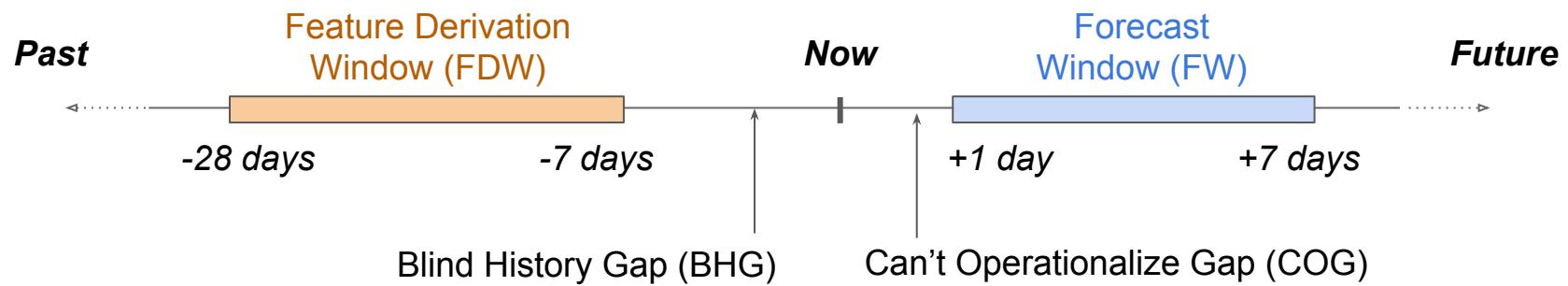
- Complexity is in the pipeline
- Each step is understandable but how to put them together?
- Parallelizable but infrastructure heavy

## The Plan

$$\begin{aligned}
 & \delta\mu + \left[ (\mu + p) v^a + (\delta\mu + \delta p) v^a + \Pi_\beta^\alpha v^\beta \right] v_\alpha - 2(\mu + p) C_{ab} v^a v^b \\
 &= \sum_j \left\{ \delta\mu_{(j)} + \left[ (\mu_{(j)} + p_{(j)}) v_{(j)}^a + (\delta\mu_{(j)} + \delta p_{(j)}) v_{(j)}^a + \Pi_{(j)\beta}^\alpha v_{(j)}^\beta \right] v_{(j)a} \right. \\
 &\quad \left. - 2(\mu_{(j)} + p_{(j)}) C_{ab} v_{(j)}^\alpha v_{(j)}^\beta \right\}, \\
 & \delta p + \frac{1}{3} \left[ (\mu + p) v^a + (\delta\mu + \delta p) v^a + \Pi_\beta^\alpha v^\beta \right] v_\alpha - \frac{2}{3}(\mu + p) C_{ab} v^a v^b \\
 &= \sum_j \left\{ \delta p_{(j)} + \frac{1}{3} \left[ (\mu_{(j)} + p_{(j)}) v_{(j)}^a + (\delta\mu_{(j)} + \delta p_{(j)}) v_{(j)}^a + \Pi_{(j)\beta}^\alpha v_{(j)}^\beta \right] v_{(j)a} \right. \\
 &\quad \left. - \frac{2}{3}(\mu_{(j)} + p_{(j)}) C_{ab} v_{(j)}^\alpha v_{(j)}^\beta \right\}, \\
 & (\mu + p) v_\alpha + [\delta\mu + \delta p + \frac{2}{3}(\mu + p) v^a v_\beta] v_\alpha + [\Pi_\beta^\alpha + (\mu + p)(v_\alpha v^\beta - \frac{1}{3}\delta_\alpha^\beta v^\gamma v_\gamma)] v_\beta \\
 &= \sum_j \left\{ (\mu_{(j)} + p_{(j)}) v_{(j)}^a + [\delta\mu_{(j)} + \delta p_{(j)} + \frac{2}{3}(\mu_{(j)} + p_{(j)}) v^a v_\beta] v_{(j)}^a + \right. \\
 &\quad \left. [\Pi_{(j)\beta}^\alpha + (\mu_{(j)} + p_{(j)})(v_{(j)\alpha} v_{(j)}^\beta - \frac{1}{3}\delta_{\alpha\beta}^\gamma v_{(j)}^\gamma v_{(j)\gamma})] v_{(j)\beta} \right\}, \\
 & (\mu + p) v_\alpha + \delta\mu v_\beta - 2\Pi_{ab} C^{b\gamma} v_\gamma = \sum_j \left\{ (\mu_{(j)} + p_{(j)}) v_{(j)\alpha} \right. \\
 &\quad + [\delta\mu_{(j)} + \delta p_{(j)} + \frac{2}{3}(\mu_{(j)} + p_{(j)}) v_{(j)}^\beta v_{(j)\beta} - 2\Pi_{ab} C^{b\gamma} v_{(j)\gamma}] v_{(j)\alpha} \\
 &\quad + [\Pi_{(j)\beta}^\alpha + (\mu_{(j)} + p_{(j)})(v_{(j)\alpha} v_{(j)}^\beta - \frac{1}{3}\delta_{\alpha\beta}^\gamma v_{(j)}^\gamma v_{(j)\gamma})] v_{(j)\beta} \\
 &\quad \left. - \frac{2}{3}(\mu_{(j)} + p_{(j)}) v_{(j)\alpha} v_{(j)\beta} v_{(j)\beta} - 2\Pi_{(j)\alpha\beta} C^{b\gamma} v_{(j)\gamma} \right\}, \\
 & \Pi_\beta^\alpha + \left[ (\mu + p) v^a + (\delta\mu + \delta p) v^a + \Pi_\gamma^\alpha v^\gamma \right] v_\beta - \frac{2}{3}(\mu + p) C_{ab}^\alpha v^a v_\gamma - \Pi_{ab}^\alpha v_\beta v_\gamma \\
 &\quad - \frac{2}{3}\delta_\beta^\alpha \{[(\mu + p) v^a + (\delta\mu + \delta p) v^a + \Pi_\gamma^\alpha v^\gamma] v_\gamma - 2(\mu + p) C_{ab} v^a v^b\} \\
 &= \sum_j \left\{ \Pi_{(j)\beta}^\alpha + \left[ (\mu_{(j)} + p_{(j)}) v_{(j)}^a + (\delta\mu_{(j)} + \delta p_{(j)}) v_{(j)}^a + \Pi_{(j)\gamma}^\alpha v_{(j)\gamma} \right] v_{(j)\beta} \right. \\
 &\quad \left. - \frac{2}{3}(\mu_{(j)} + p_{(j)}) C_{ab}^\alpha v_{(j)\beta} v_{(j)\gamma} - \Pi_{(j)\beta}^\alpha v_{(j)\beta} v_{(j)\gamma} - \frac{2}{3}\delta_\beta^\alpha \{[(\mu_{(j)} + p_{(j)}) v_{(j)}^\gamma \right. \\
 &\quad \left. + (\delta\mu_{(j)} + \delta p_{(j)}) v_{(j)}^\gamma + \Pi_{(j)\delta}^\alpha v_{(j)}^\delta] v_{(j)\gamma} - 2(\mu_{(j)} + p_{(j)}) C_{ab}^\alpha v_{(j)}^\delta\} \right\}. \tag{31}
 \end{aligned}$$

- Complexity is in the model
- How to understand and control what the model is doing?
- How to train?

# General Time Series Framework



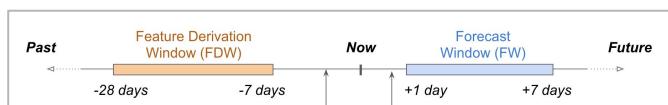
# Simple Lags

Date	Sales	Sales 3 days ago	Number of Employees	Ad Spending	Inventory Rate	Weather
11/09/17	\$432,897					
11/10/17	\$474,306					
11/11/17	\$415,434					
11/12/17	\$289,290	\$432,897	16	\$3,000	47%	Rain
11/13/17	\$355,786	\$474,306	19	\$2,100	46%	Heavy Wind
11/14/17	\$375,284	\$415,434	17	\$1,200	41%	Cloudy
...	...		...	...	...	...
tomorrow	11/15/17	?	\$289,290	20	\$1,800	31%

# Candidate Time Series Features

Time	Sales	Employees	Weather
1/1/17	\$4,438,925	16	Rain
1/2/17	\$4,703,801	17	Heavy Winds
1/3/17	\$4,646,682	17	Heavy Winds
1/4/17	\$4,162,316	16	Partly Cloudy
1/5/17	\$2,811,162	16	Partly Cloudy
1/6/17	\$3,394,591	16	Frost
1/7/17	\$3,328,435	15	Heavy Winds
1/8/17	\$4,227,790	15	Frost
1/9/17	\$5,426,552	16	Heavy Winds
1/10/17	\$4,472,594	16	Frost

+2 days



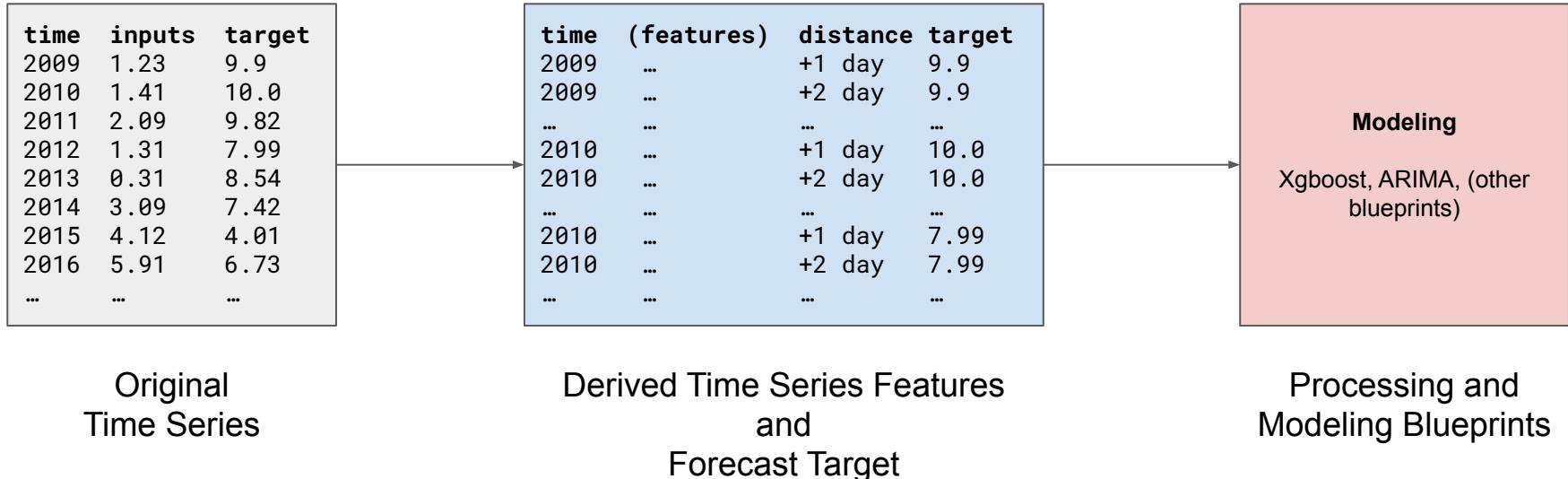
## Window Features

- Rolling min, max, mean
- Rolling majority
- Rolling entropy
- Rolling text statistics
- Bollinger bands

## Target Feature

- +1 day ahead
- +2 days ahead
- etc.

# Time Series Feature Engineering



Original  
Time Series

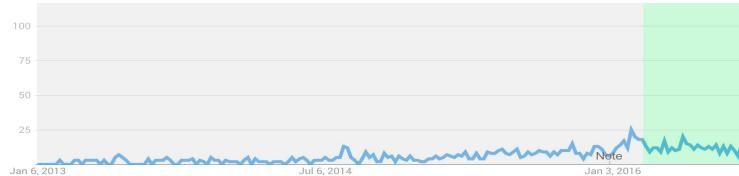
Derived Time Series Features  
and  
Forecast Target

Processing and  
Modeling Blueprints

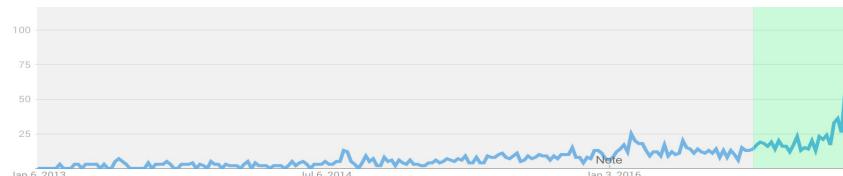
# Out-of-Time Validation

Walk-forward a set of backtests

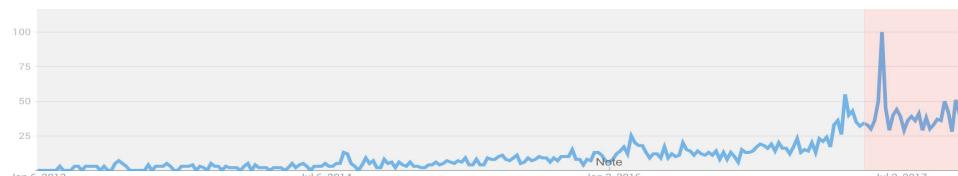
Backtest 2



Backtest 1



Holdout



(Total validation score) = (validation score #1) + (validation score #2) + (validation score #3) + ...

# Tutorial

## Walkthrough in Python

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** Shows the URL "localhost:8889/notebooks/Walkthrough.ipynb", the Jupyter logo, the title "jupyter Walkthrough", and a "Logout" button.
- Toolbar:** Includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, and various cell type icons (Code, Markdown, etc.).
- Cell Content:** A cell titled "Time Series Workshop – Walkthrough" containing the following text:

ODSC East – April 15, 2020  
Mark Steadman Viktor Kovryzhkin  
<https://github.com/vikua/ts-workshop>
- Code Cell:** An input cell labeled "In [1]" with the following Python code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import display
from sklearn.linear_model import ElasticNetCV
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import r2_score
from sklearn.base import BaseEstimator, RegressorMixin
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
```
- Section Header:** A section titled "Example Time Series Data Set".

# Time Series Modeling Families

## Integrated Models



ARIMA, ETS, RNNs, etc.

## Per Forecast Distance Models



Xgboost, elastic-net, etc.

## Trends and Decomposition Models



Fourier models, linear trends,  
decompositions, etc.

# Target Transformations

## Log Transform

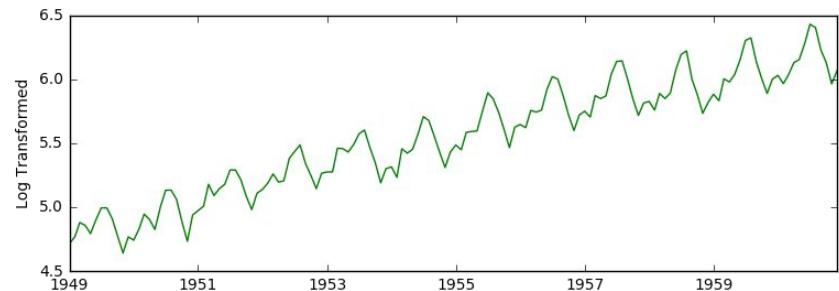
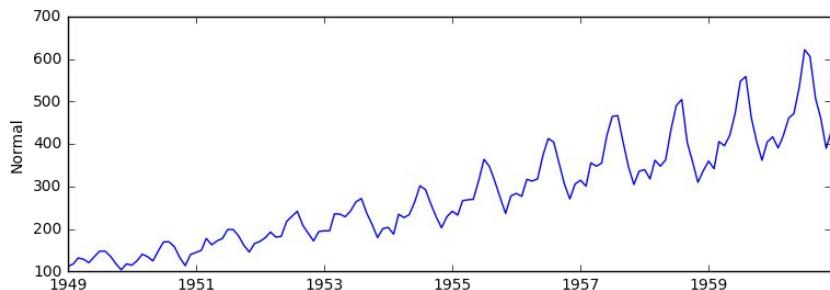
$$\log(y_t) = \log(y_{t-1}) + \log(y_{t-2}) + \dots$$

$$y_t = \exp(\log(y_{t-1}) + \log(y_{t-2}) + \dots)$$

$$y_t = \exp(\log(y_{t-1})) * \exp(\log(y_{t-2})) * \dots$$

$$y_t = y_{t-1} * y_{t-2} * \dots$$

- Results in a multiplicative model (instead of additive)
- Removes exponential growth and variance

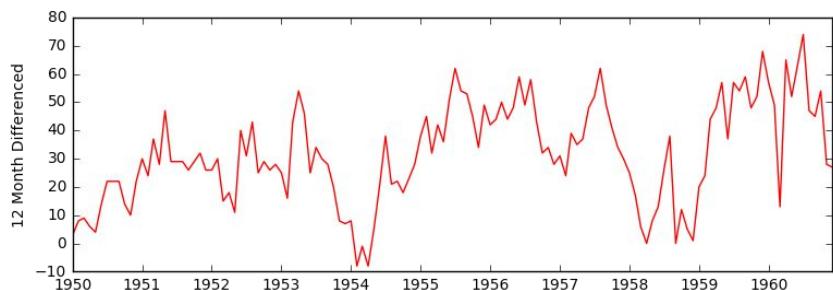
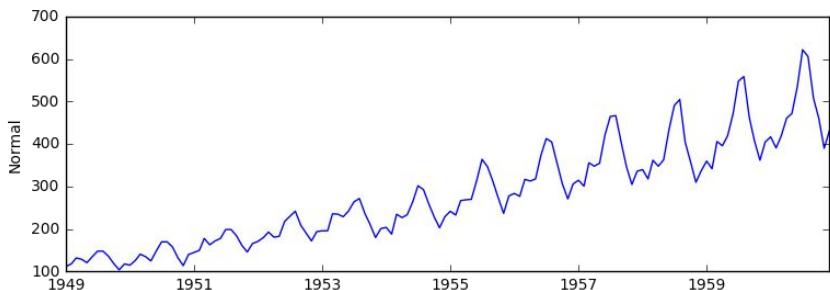


# Target Transformations

## Differencing

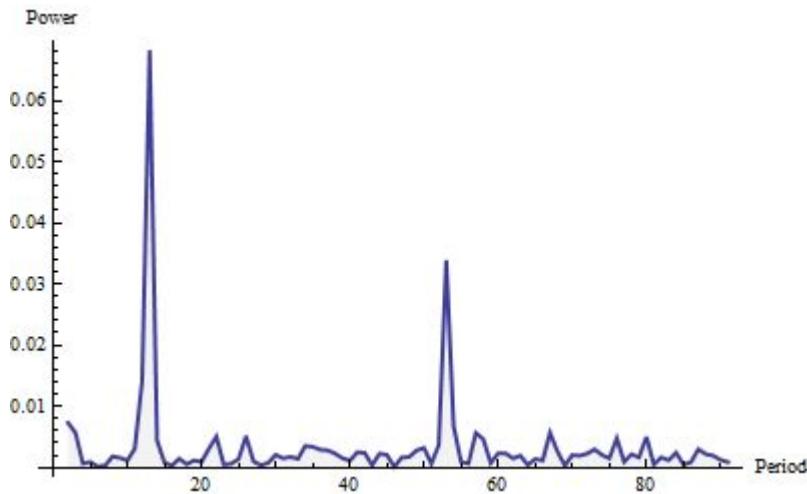
$$y_t - y_{t-p} = y_{t-1} + y_{t-2} + \dots$$

- Removes trend and periodicity / creates a more stationary signal



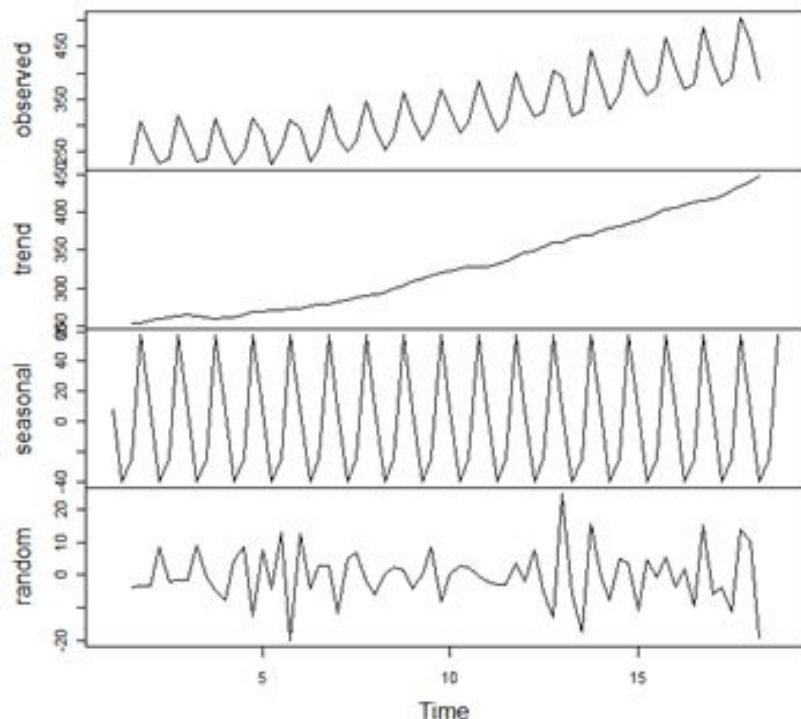
# Periodicity / Periodogram

Periodic pattern detection



# Time Series Decomposition

Decomposition of additive time series



Observed data

=

Trend component

+

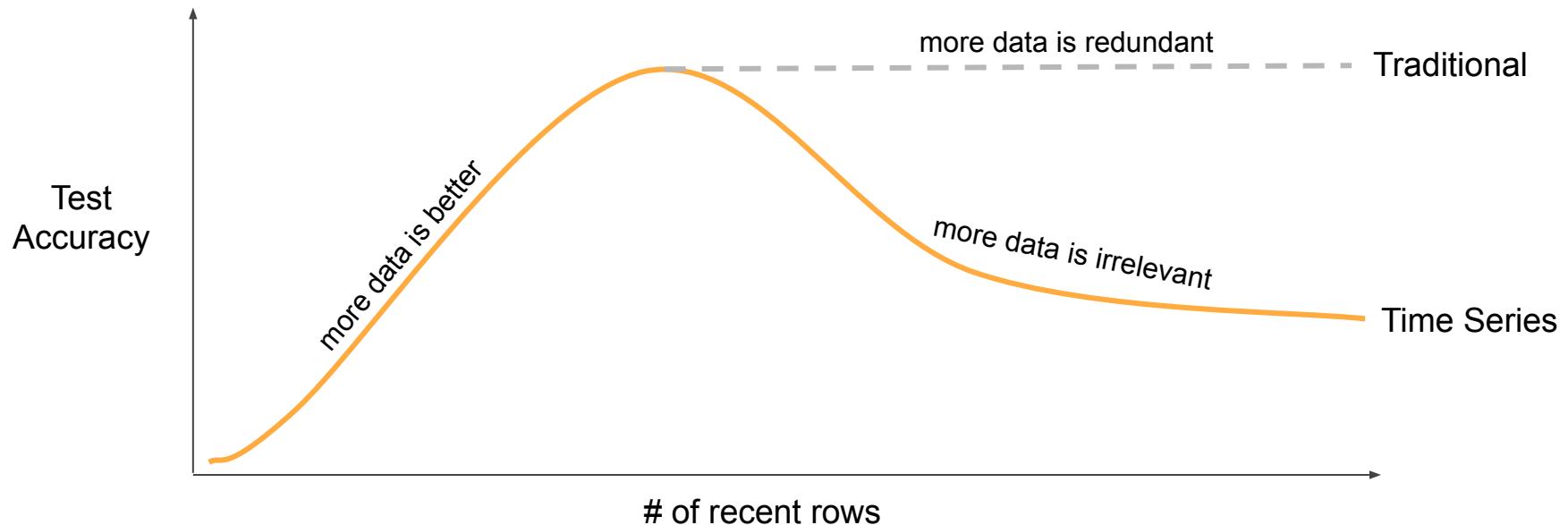
Annual seasonal component

+

Annual seasonal component

# How much data

## Time series learning curves

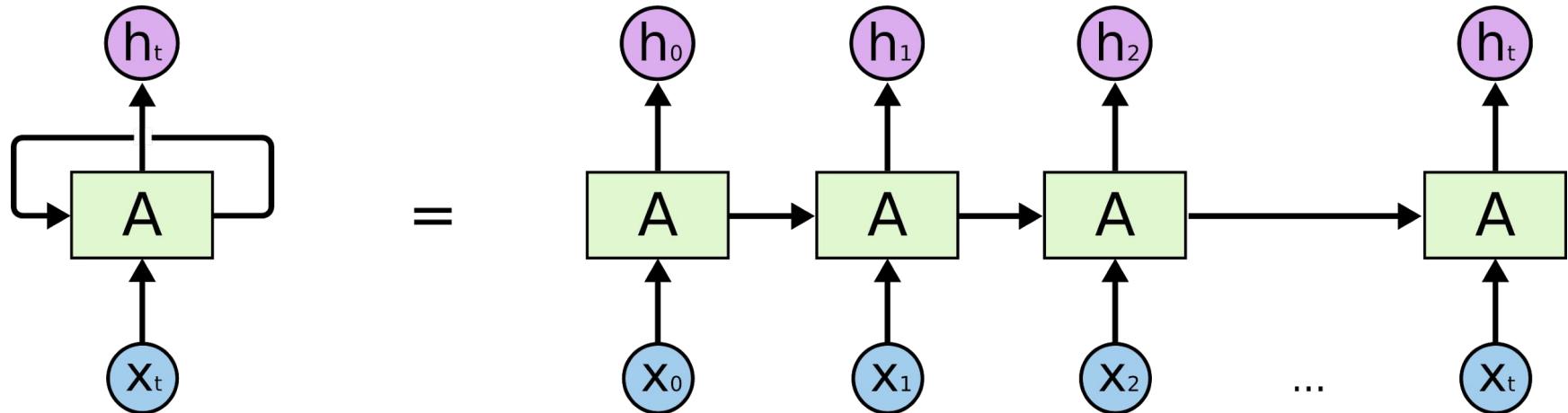


If you have a large big dataset and you  
train a very big neural network, then  
success is guaranteed!

Ilya Sutskever

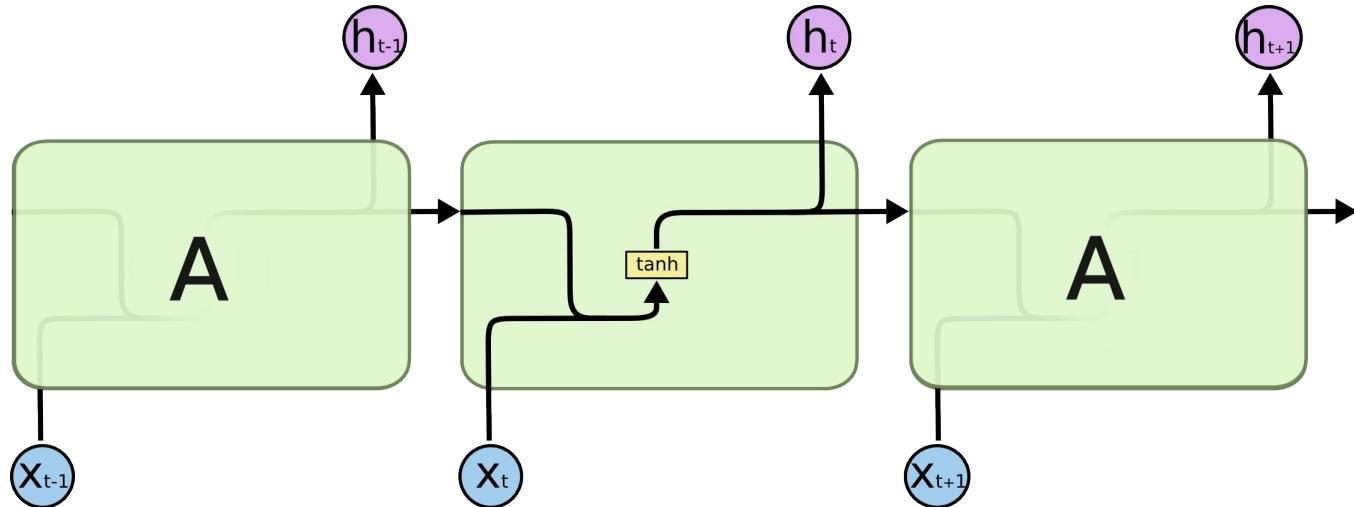
Deep learning - success is  
guaranteed.

# RNN refresher



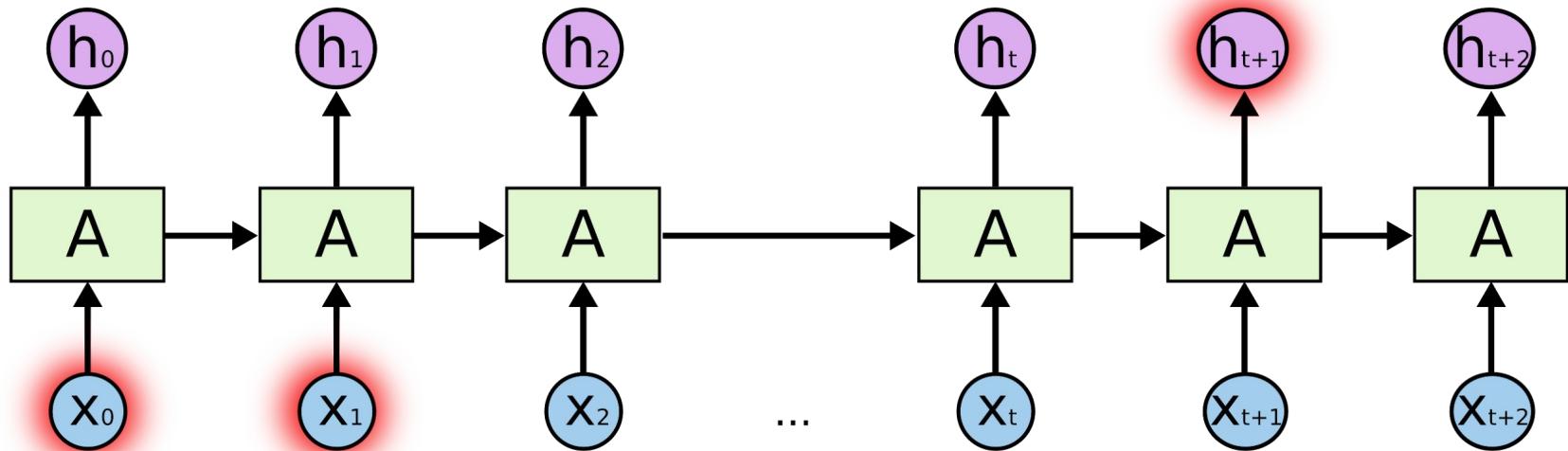
<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

# RNN refresher



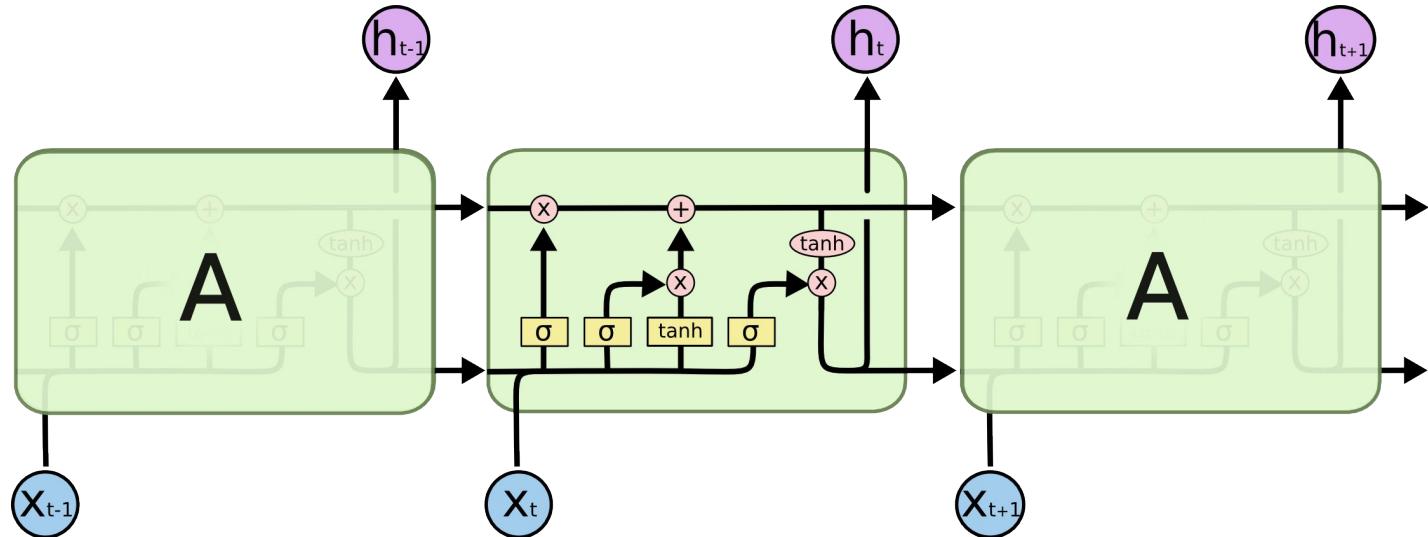
<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

# RNN refresher



<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

# LSTM refresher



<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

# Why RNNs for time series

---

1. Autocorrelated process.
2. Huge success in NLP which is also sequential problem.
3. *Maybe* can do automatic feature extraction and learn hidden patterns in data.
4. *Maybe* can learn trend and seasonality automatically, without special feature engineering.

# Preprocessing

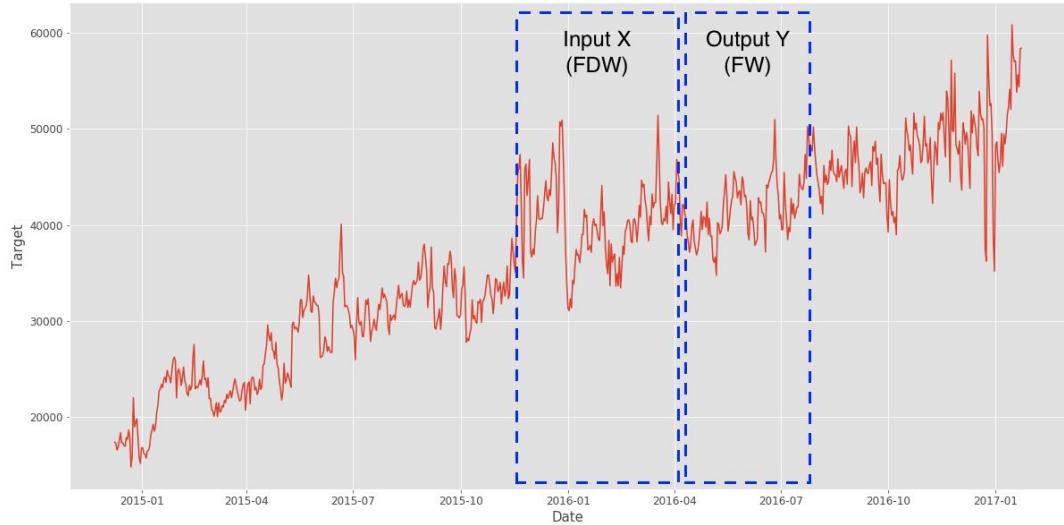
Generate sequences of historical data points and sequences of targets.

$$[X_1, X_2, X_3, X_4] \rightarrow [y_5, y_6, y_7]$$

$$[X_2, X_3, X_4, X_5] \rightarrow [y_6, y_7, y_8]$$

...

$$[X_t, X_{t+1}, X_{t+2}, X_{t+3}] \rightarrow [y_{t+4}, y_{t+5}, y_{t+6}]$$



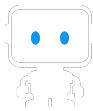
# Normalization

---

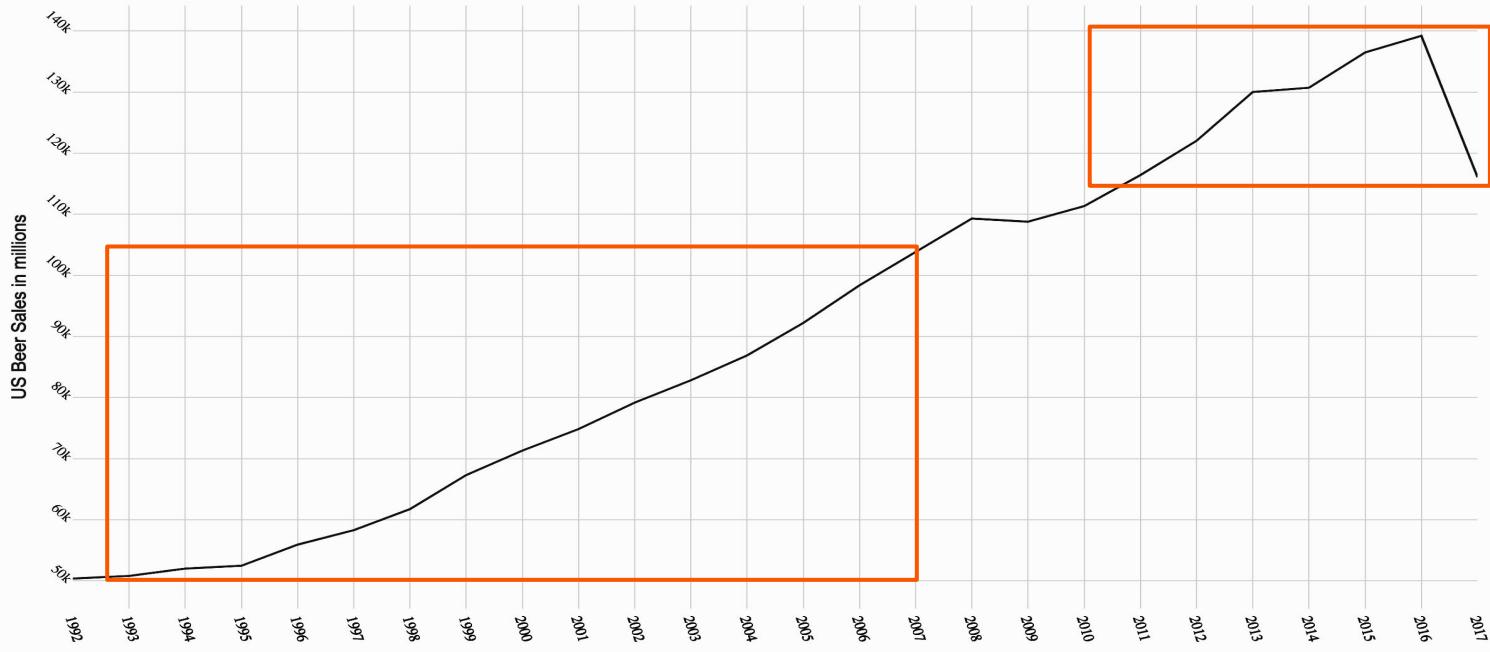
Normalization is very important step.

Wrong normalization technique or mistakes can ruin NN performance no matter how sophisticated architecture is.

It can be done across all time series, but usually better to do it locally, e.g. divide each sliding window by some value like mean or median. c

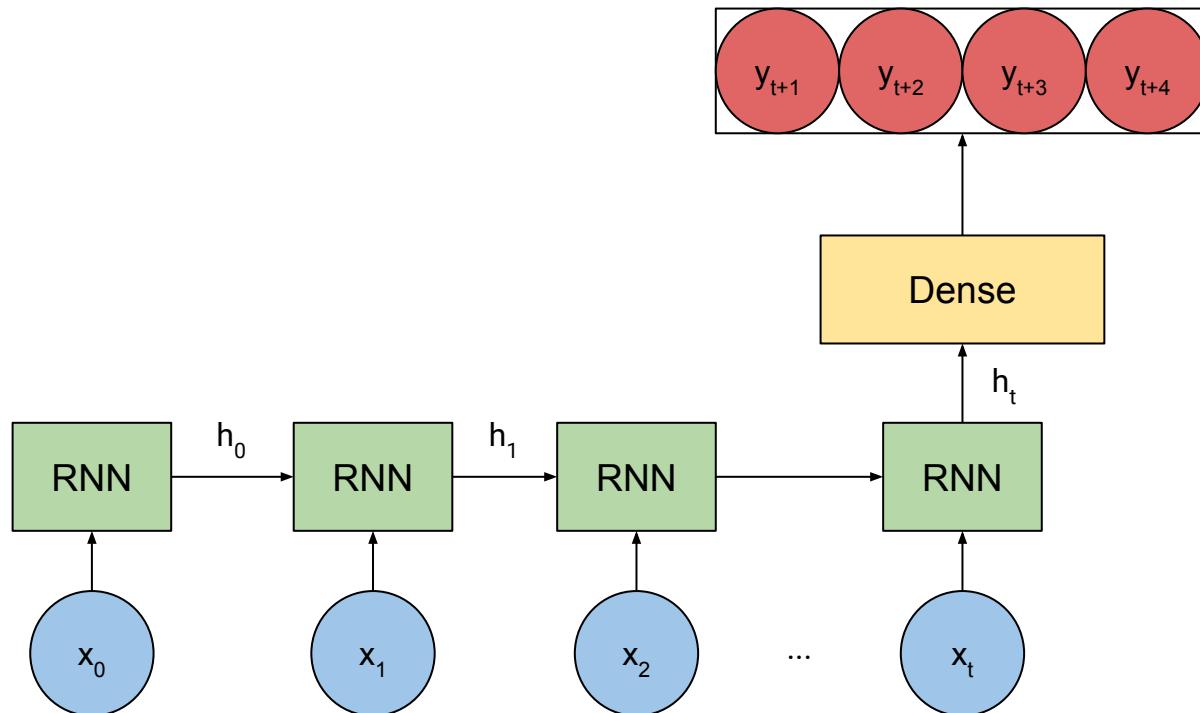


# Normalization

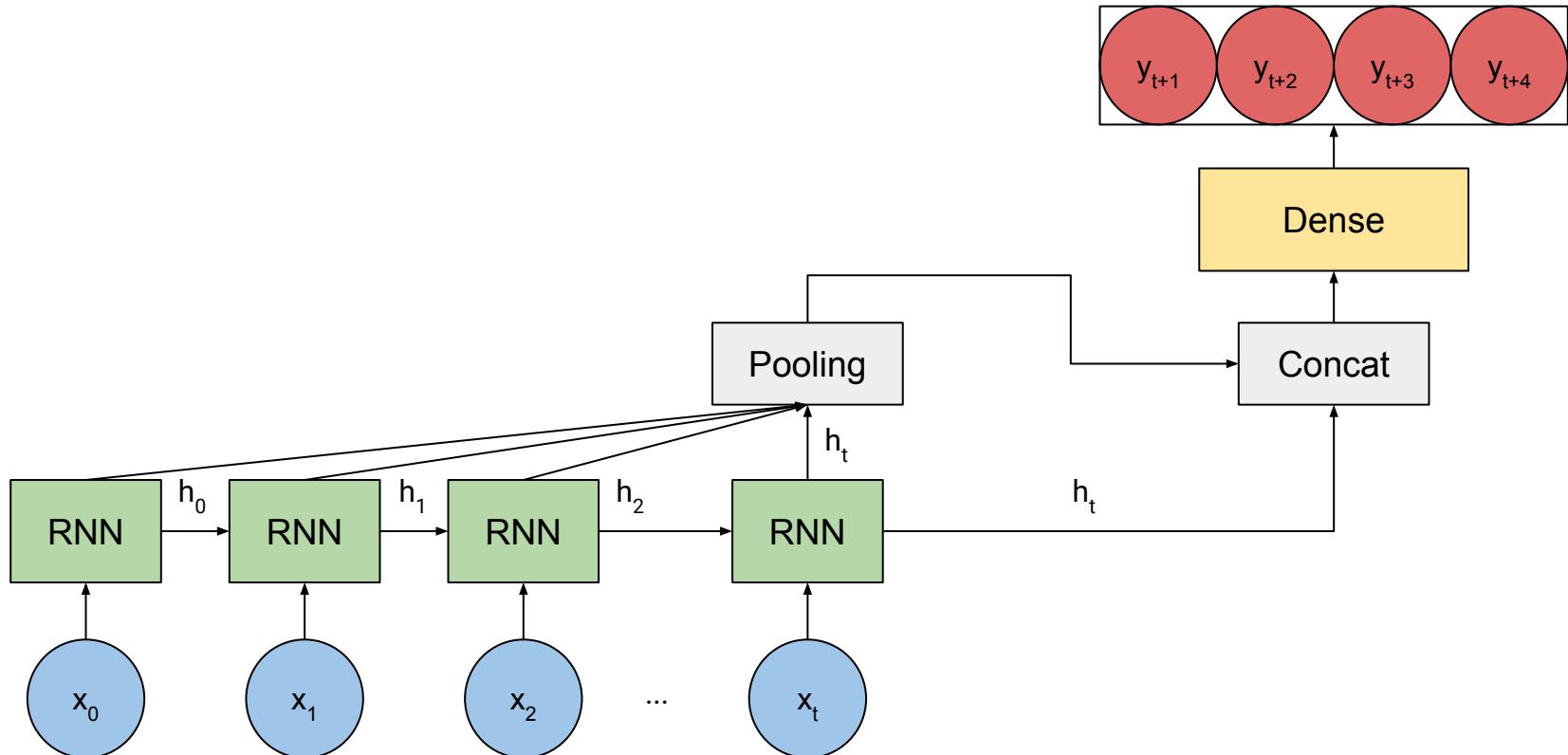


Now let's talk about models

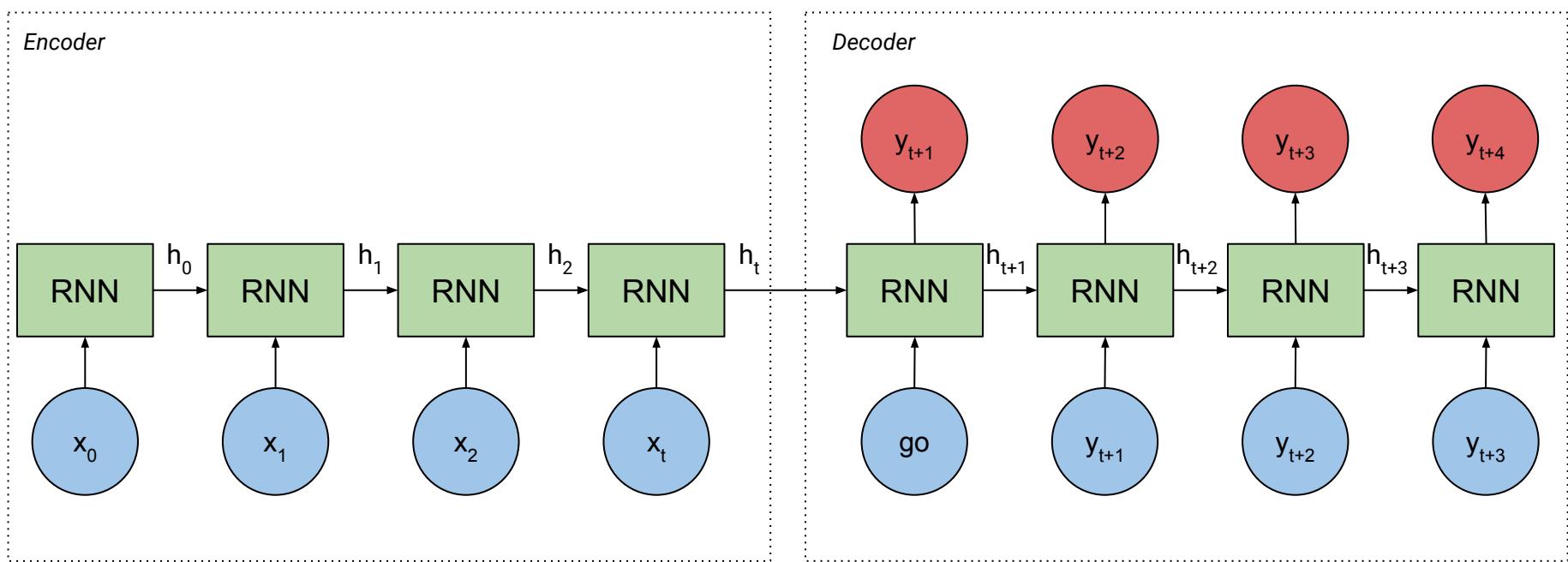
# Simple LSTM architecture



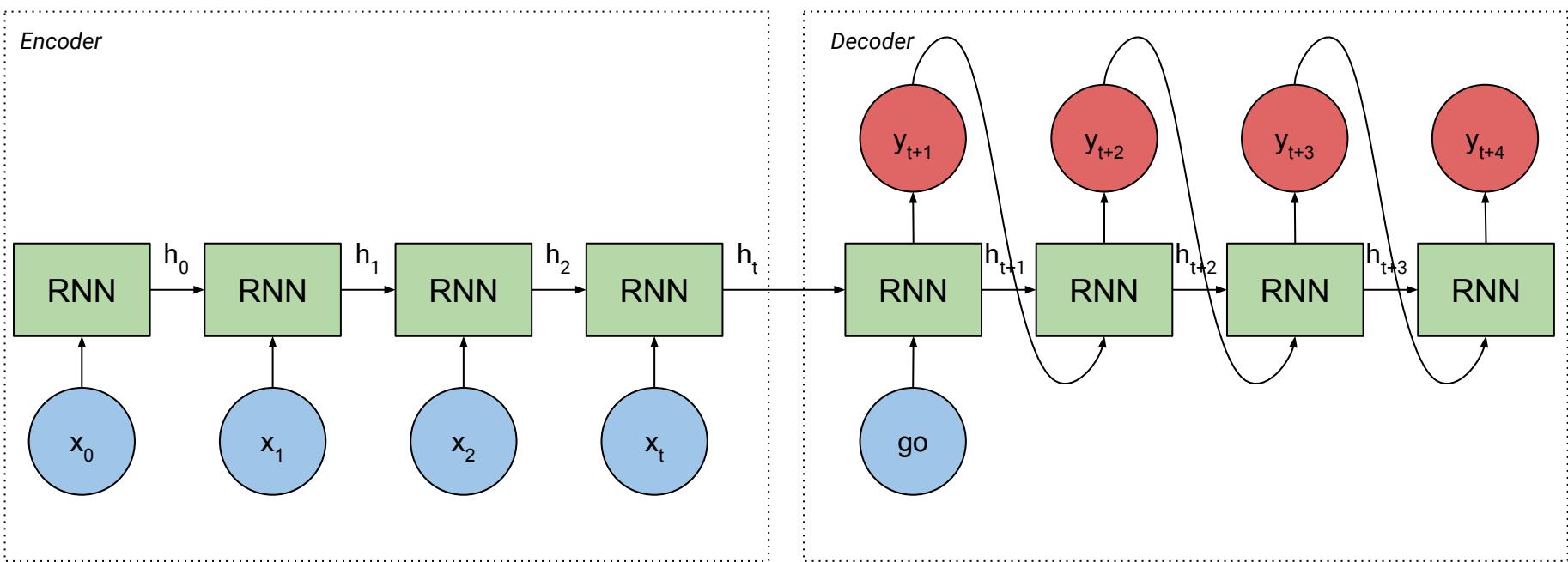
# Improvement: use all hidden states

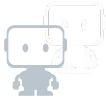


# Sequence to sequence



# Sequence to sequence (prediction)





# Feature preprocessing

## Numerical

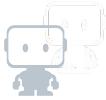
Missing values imputation & normalization

## Categorical & Series ID

Categorical embedding or one-hot encoding

## A priori

According to type, but also feed into  
network separately

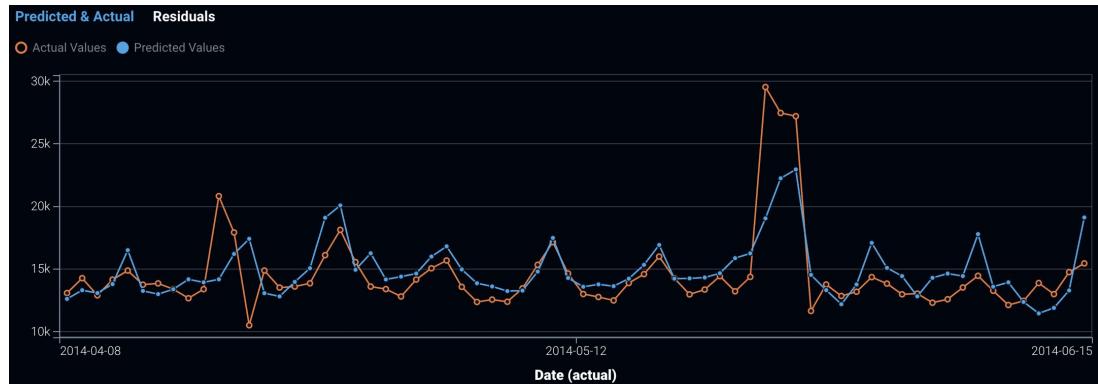


# A priori features

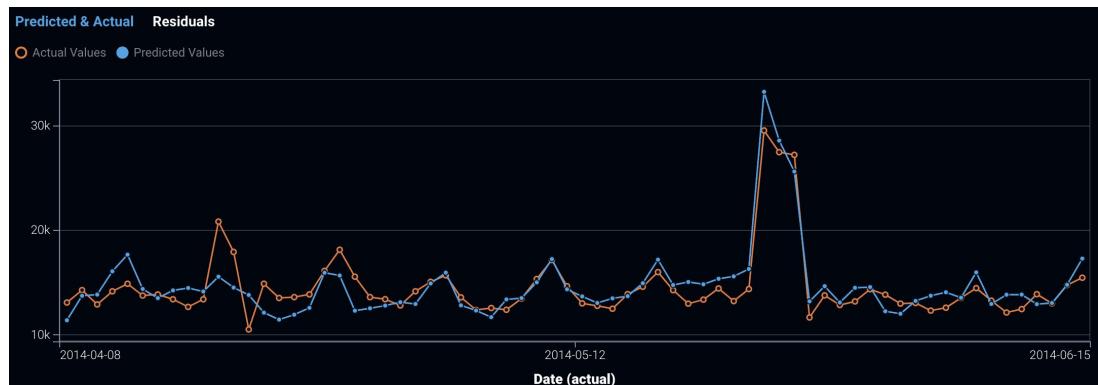
Features, which are known in advance, at the time of prediction.

For example, whether prediction date is a Holiday is known upfront, or whether store is going to do marketing campaign for an item.

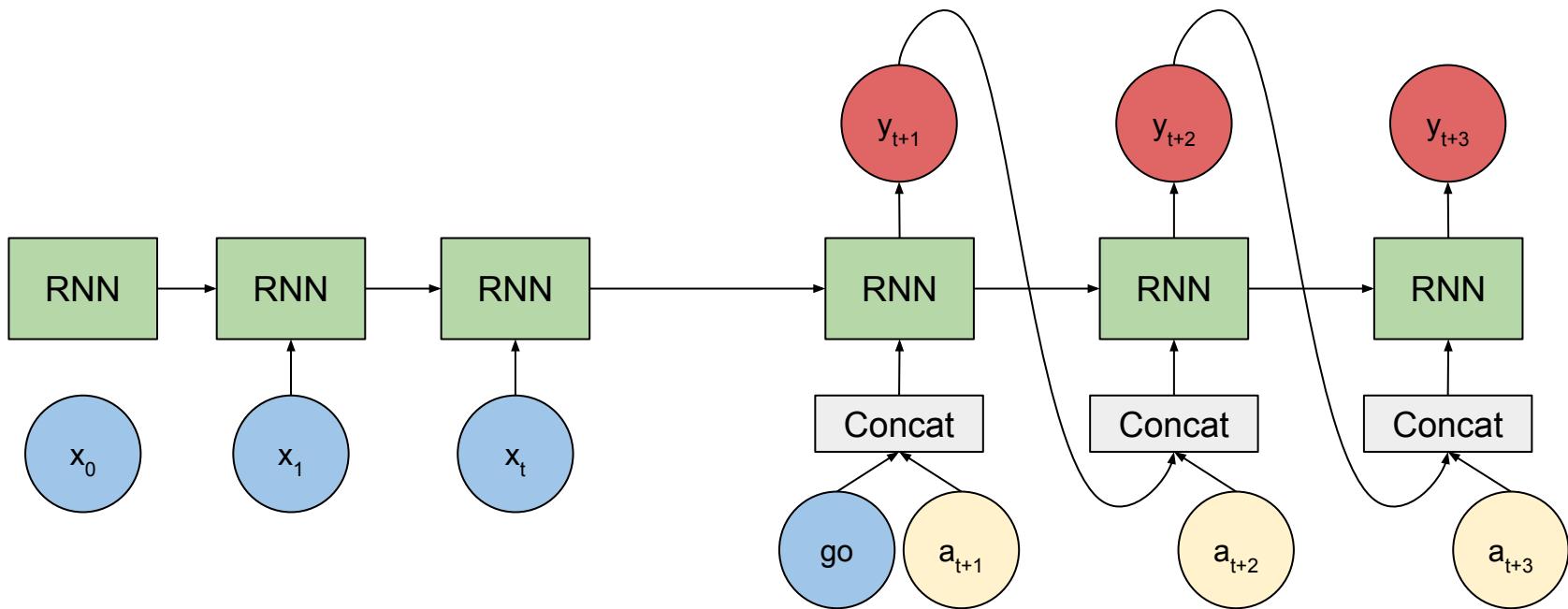
## Without a priori features



## With a priori features



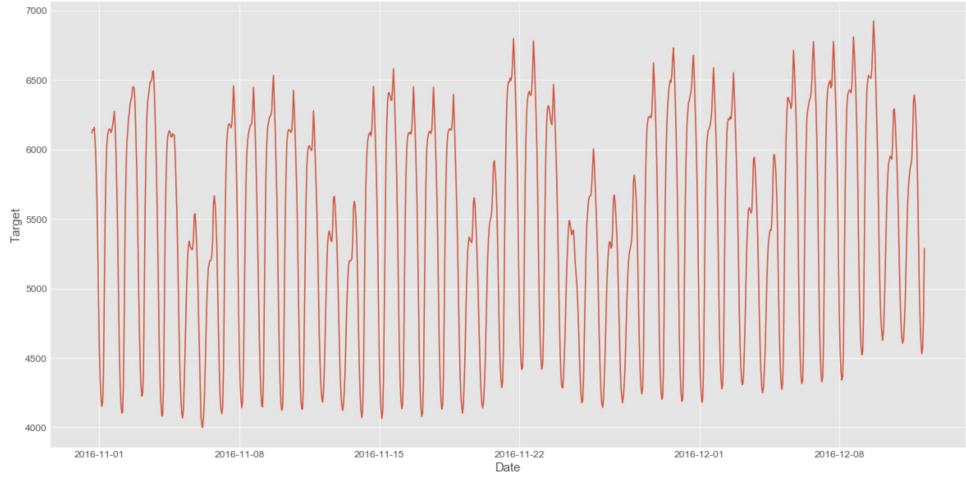
# Sequence to sequence with a priori features



# Exercise

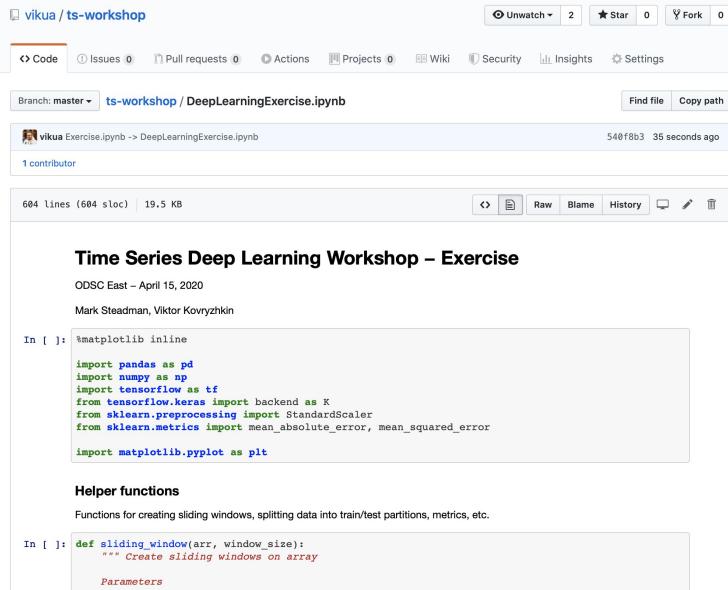
Setup:

- Hourly dataset of energy consumption on NYC
- Multiple seasonalities - intraday and intraweek
- FDW (X): **168 hours (7 days)**
- FW (Y): **24 hours**
- Model: LSTM seq2seq



# Exercise

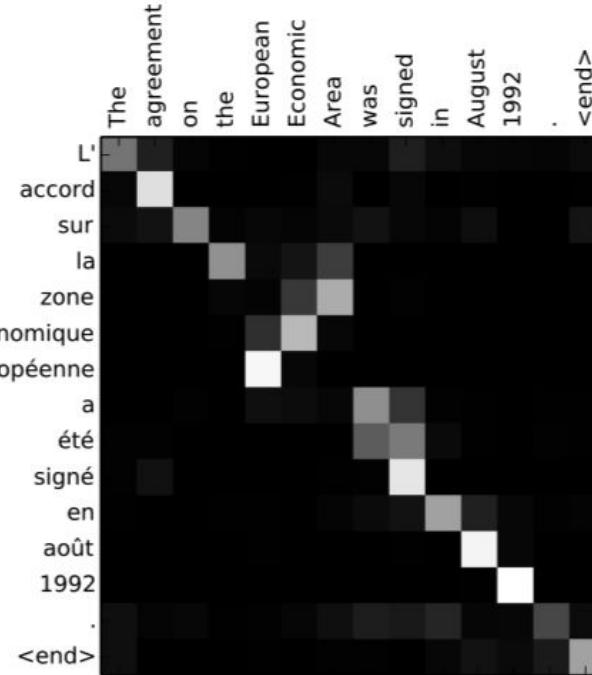
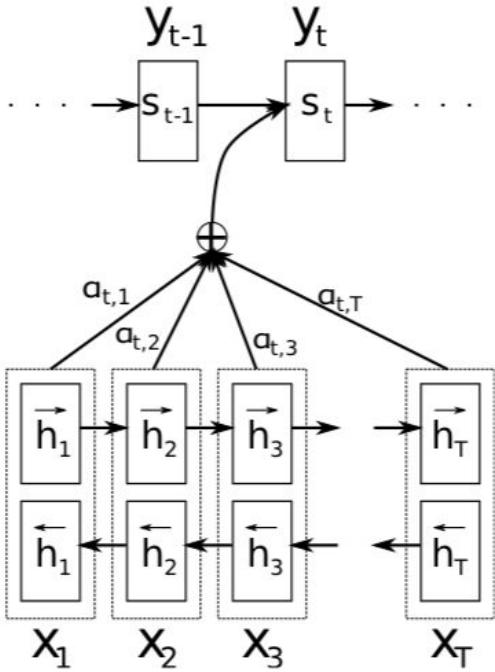
## Walkthrough in Python



The screenshot shows a GitHub repository page for 'vikua / ts-workshop'. The repository has 2 stars and 0 forks. The 'Code' tab is selected, showing a file named 'DeepLearningExercise.ipynb' with a branch of 'master'. The file was last updated 35 seconds ago by 'vikua'. It contains 604 lines and is 19.5 KB in size. The notebook content includes a title 'Time Series Deep Learning Workshop – Exercise', a subtitle 'ODSC East – April 15, 2020', and author information 'Mark Steadman, Viktor Kovyrzhkin'. Below this is a code cell starting with '%matplotlib inline' and importing various Python libraries like pandas, numpy, tensorflow, keras, and sklearn. A section titled 'Helper functions' follows, with a code cell defining a 'sliding\_window' function that creates sliding windows on an array. The parameters for this function are listed as 'Parameters'.

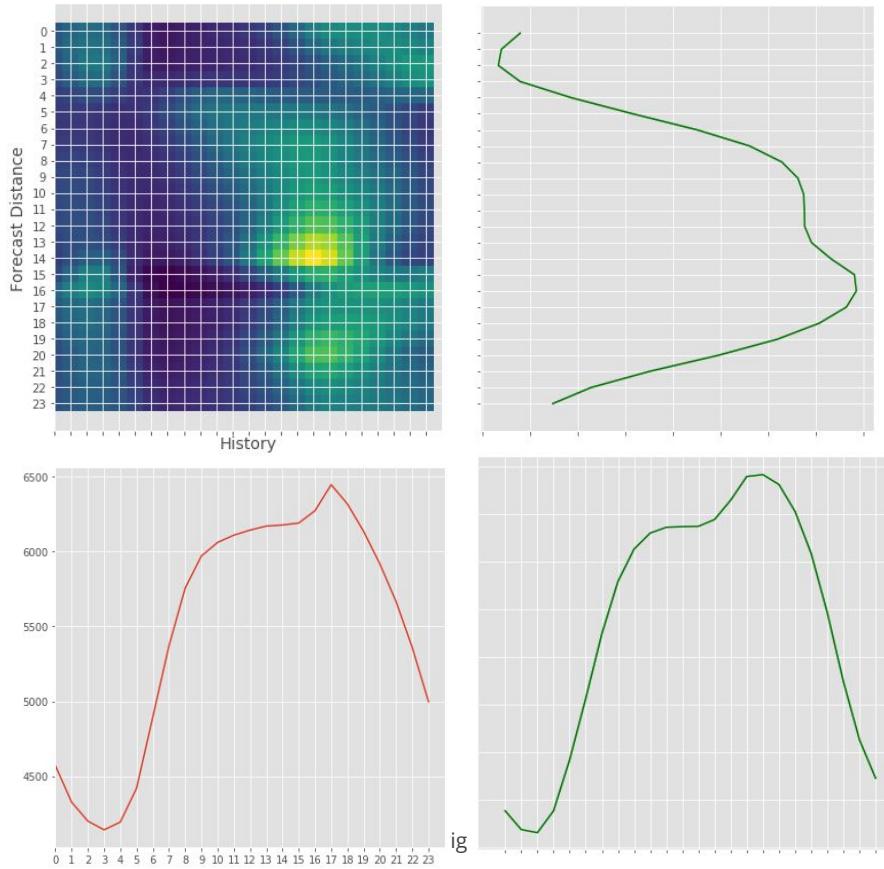
<https://github.com/vikua/ts-workshop>

# Attention



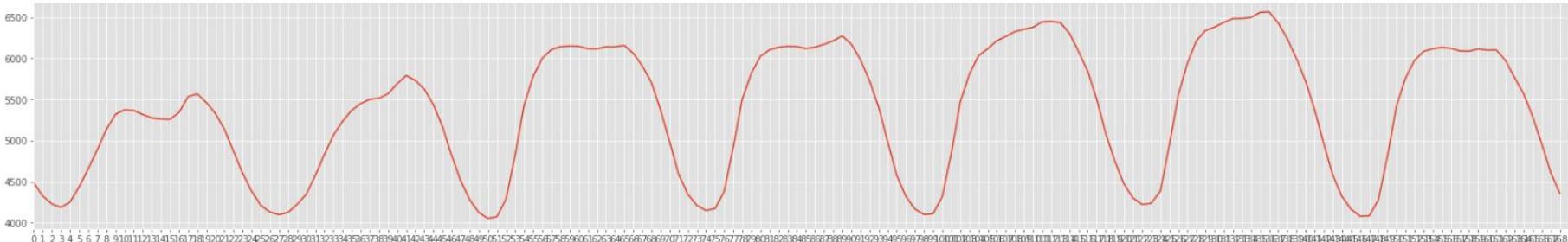
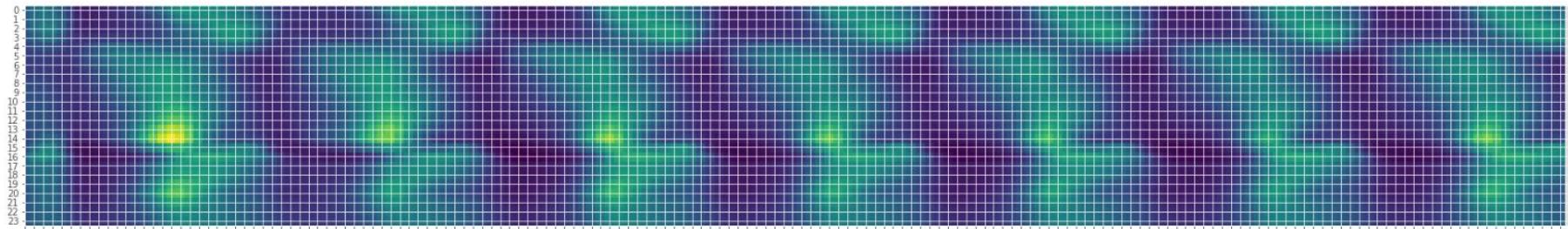
<https://arxiv.org/abs/1409.0473>

# Attention

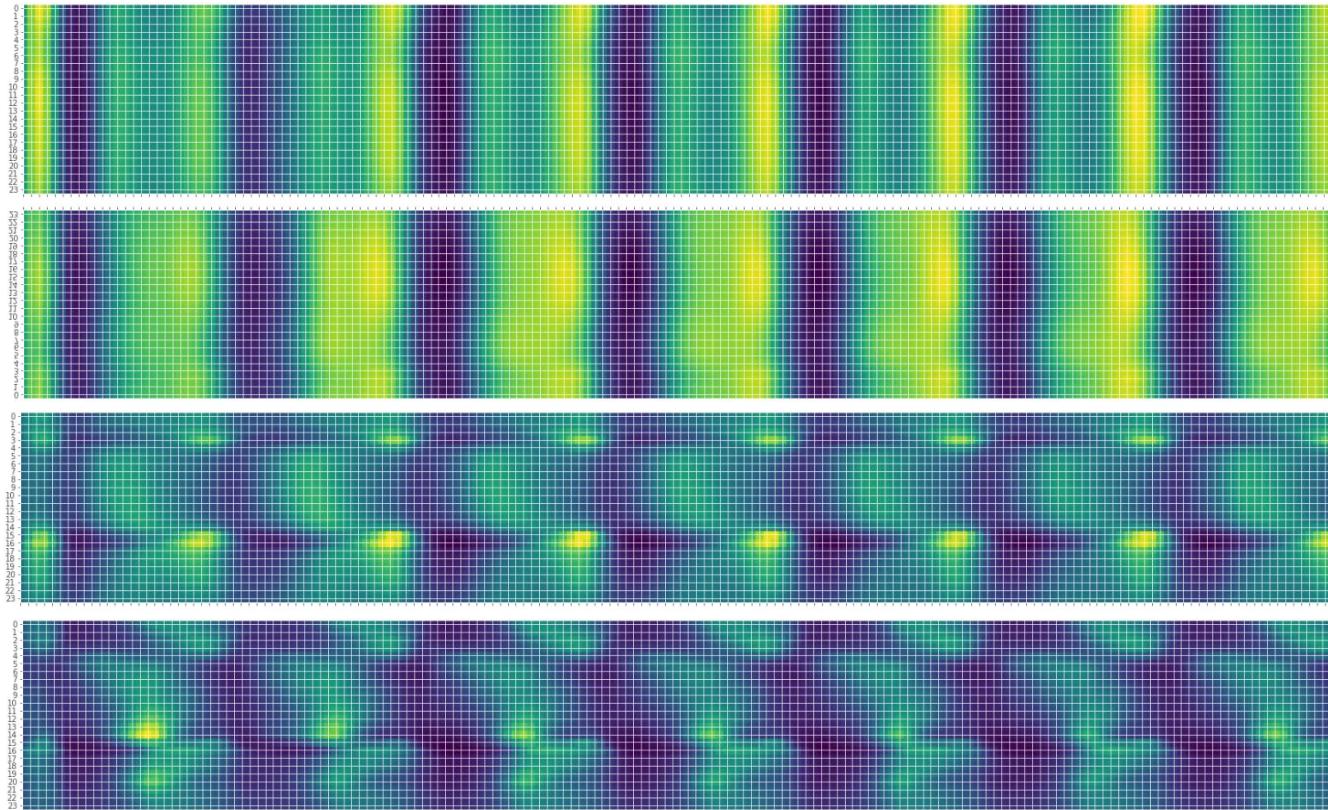


<https://arxiv.org/abs/1409.0473>

# Attention weight (univariate model)



# Attention weight (univariate model)



5 epochs

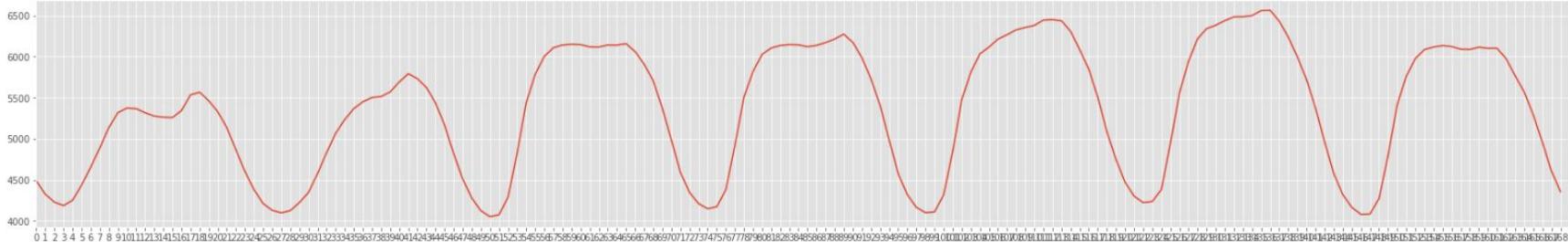
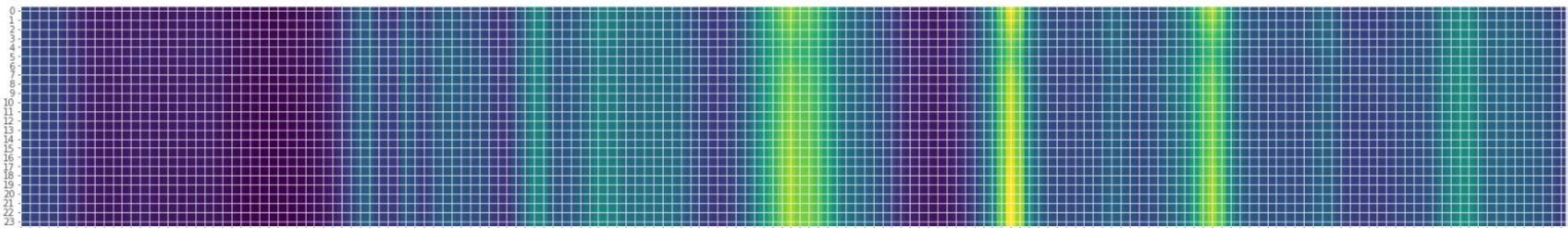
10 epochs

20 epochs

50 epochs

# Attention weight (multivariate model)

**With more predictive features attention doesn't make that much of a sense**



# Other architectures

---

ES-RNN

First place winning solution in M4 competition (by Uber)

DeepAR

<https://docs.aws.amazon.com/sagemaker/latest/dg/deepar.html>

Deep state space models

<https://papers.nips.cc/paper/8004-deep-state-space-models-for-time-series-forecasting>

Auto-encoders for feature extraction

<https://eng.uber.com/neural-networks/>

# Automation

---

## Key requirements to automate this process

- Time series data and framework
- Target transformations
- Feature generation
- Feature reduction
- Standard data preparation
- Model generation and validation