Microsoft

Forecasting Financial Time Series with Deep Learning on Azure

IN-PERSON TRAINING

***Francesca Lazzeri**, PhD*
@frlazzeri

Wee Hyong Tok
@weehyong

Krishna Anumalasetty
@KrishnaAnumalas

This training course is for you because...

- You're a business analyst or a data scientist who needs to build time series forecasting models.
- You're a developer who needs to understand better how time series forecasting works and you want to learn how to operationalize your time series forecasting models.

Prerequisites:

- Experience coding in Python
- A basic understanding of machine learning and deep learning topics and terminology as well as the mathematics used for machine learning
- An Azure Notebooks account

This training course is for you because...

- You're a business analyst or a data scientist who needs to build time series forecasting models.
- You're a developer who needs to understand better how time series forecasting works and you want to learn how to operationalize your time series forecasting models.

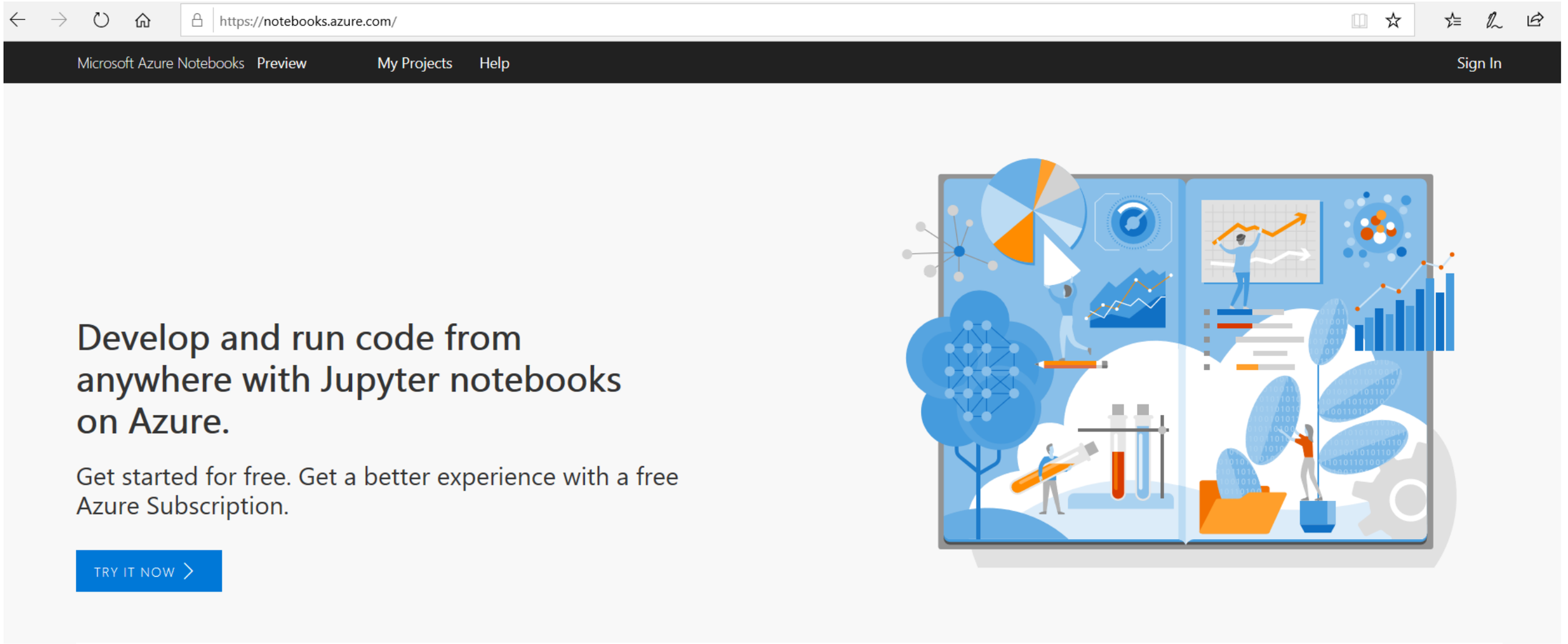
Prerequisites:

- Experience coding in Python
- A basic understanding of machine learning and deep learning topics and terminology as well as the mathematics used for machine learning
- An Azure Notebooks account

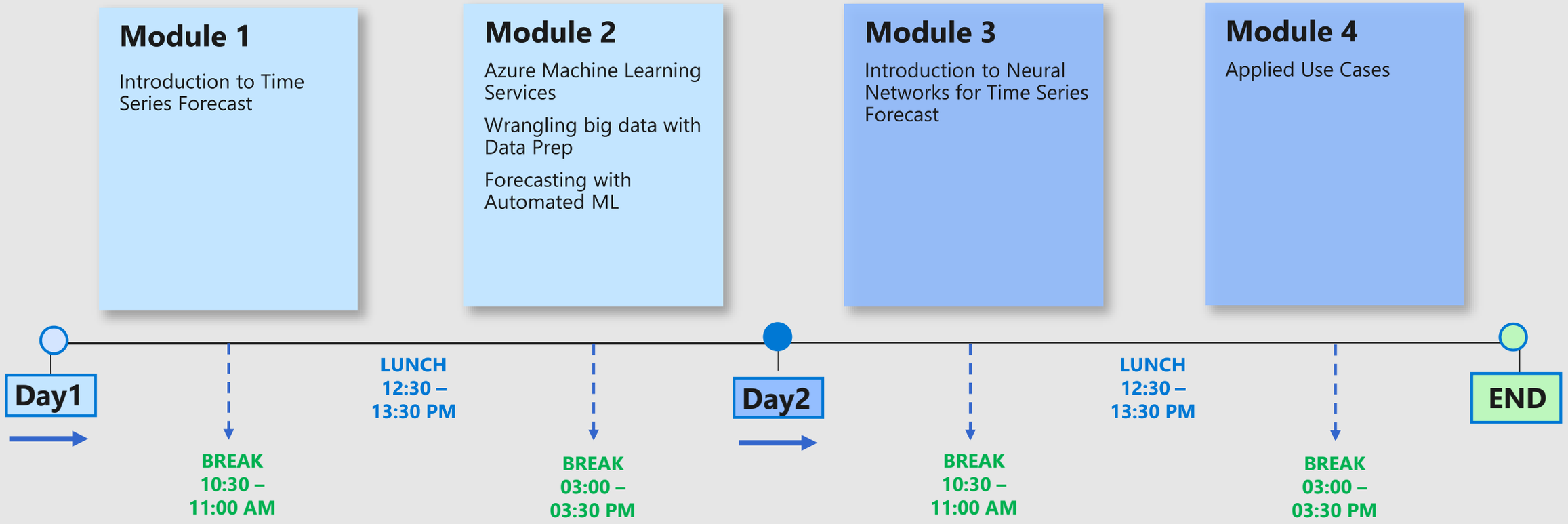
@frlazzeri

An Azure Notebooks account?

Azure Notebooks: https://aka.ms/AzureNB



Agenda

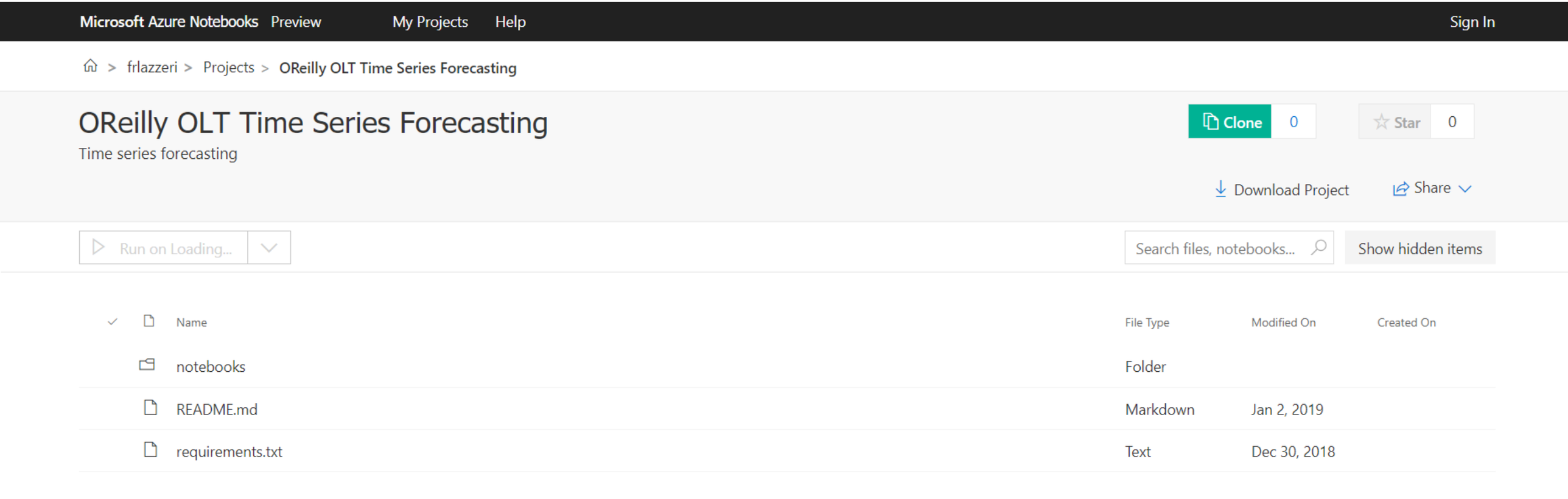

Module 1
Introduction to Time Series Forecast
Module 2
Azure Machine Learning Services
Wrangling big data with Data Prep
Forecasting with Automated ML
Module 3
Introduction to Neural Networks for Time Series Forecast
Module 4
Applied Use Cases
Day1
BREAK 10:30 – 11:00 AM
LUNCH 12:30 – 13:30 PM
BREAK 03:00 – 03:30 PM
Day2
BREAK 10:30 – 11:00 AM
LUNCH 12:30 – 13:30 PM
BREAK 03:00 – 03:30 PM
END

Materials & Exercises

@frlazzeri

- Projects hosted on Azure Notebooks: https://notebooks.azure.com/frlazzeri/projects/AIConferenceNYC

- To learn more about Azure Notebooks: aka.ms/AzureNB

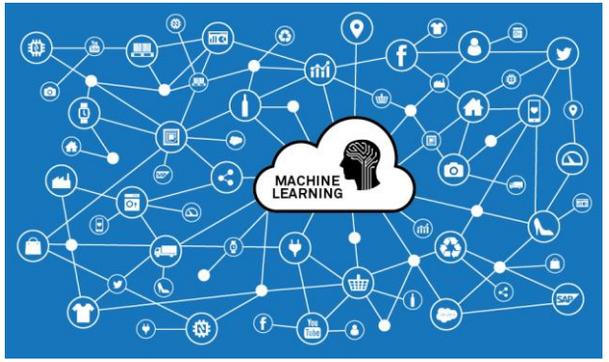


@frlazzeri

@KrishnaAnumalas

@weehyong

Microsoft



Microsoft

@frlazzeri

Module 1
Introduction to Time Series Forecasting

- What Makes Time Series Special
- How to Load and Handle Time Series in Pandas
- How to Check Stationarity of a Time Series
- How to make a Time Series Stationary

Uses of Time Series

@frlazzeri

- The most important use of studying time series is that it helps us to predict the future behavior of the variable based on previous experience
- It is helpful for business planning as it helps in comparing the actual current performance with the expected one
- From time series, we get to study the past behavior of the phenomenon or the variable under consideration
- We can compare the changes in the values of different variables at different times or places, etc.

Investigating financial time series:

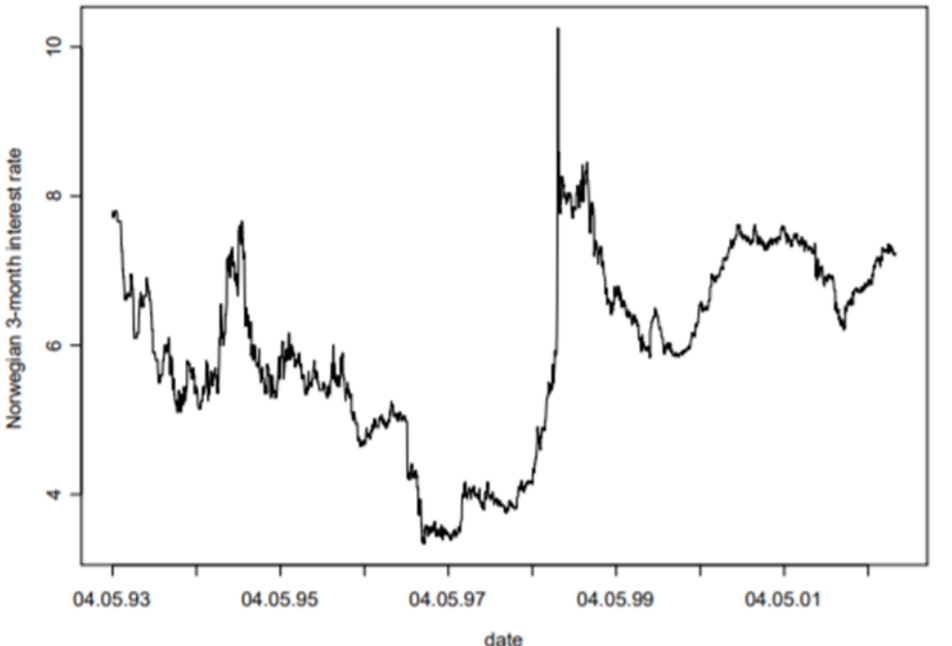
@frlazzeri

Reason #1:

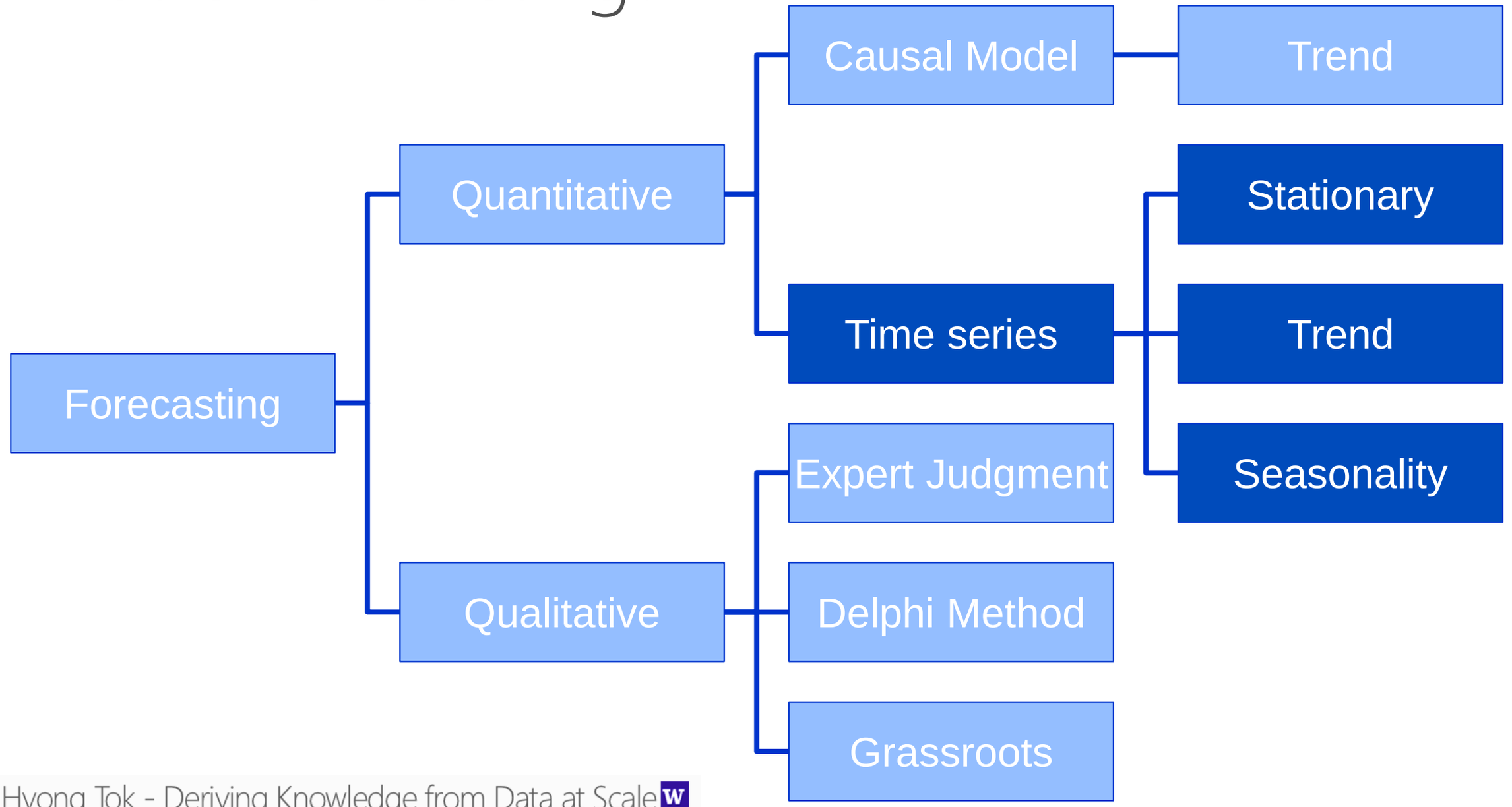
- Understand how prices behave. The variance of the time series is particularly relevant.
- Tomorrow's price is uncertain, and it must therefore be described by a probability distribution.
- This means that statistical methods are the natural way to investigate prices. Usually one builds a model, which is a detailed description of how successive prices are determined.

Reason #2:

- The second objective is to use our knowledge of price behavior to reduce risk or take better decisions.
- Time series models may for instance be used for forecasting, option pricing and risk management.



What is Forecasting?



What is Time Series?

@frlazzeri

Machine Learning Dataset →

- observation #1
- observation #2
- observation #3

Time Series Dataset →

- Time #1, observation
- Time #2, observation
- Time #3, observation

What is Time Series?

@frlazzeri

Machine Learning Dataset →

- observation #1
- observation #2
- observation #3

Time Series Dataset →

- Time #1, observation
- Time #2, observation
- Time #3, observation

What makes Time Series Special?

	Date	Observation
History	2018-06-04	60
	2018-06-05	64
	2018-06-06	66
	2018-06-07	65
	2018-06-08	67
	2018-06-09	68
	2018-06-10	70
	2018-06-11	69
	2018-06-12	72
Future	2018-06-13	?
	2018-06-14	?
	2018-06-15	?

- **Time Series Analysis**

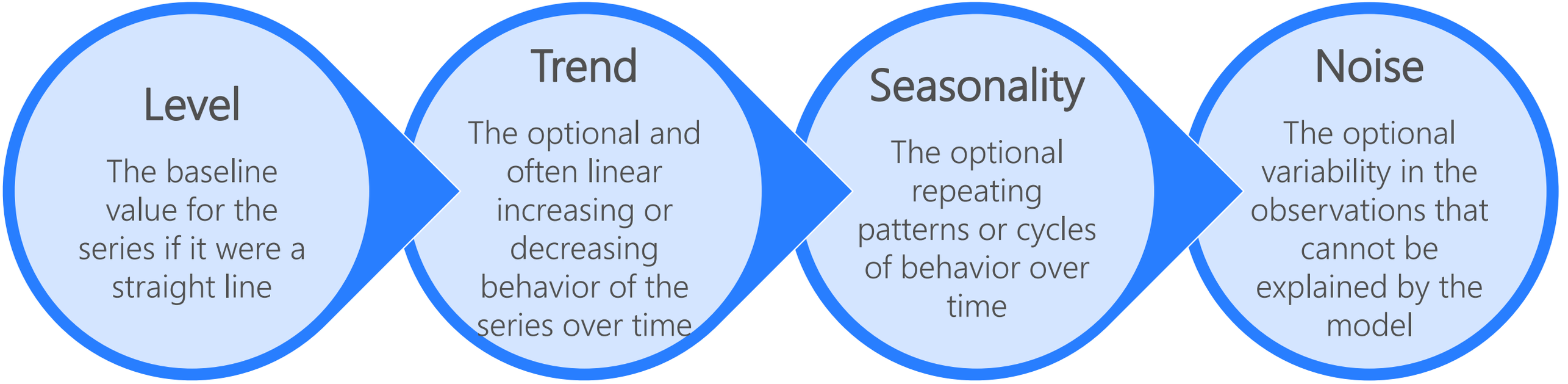
Involves developing models that best capture or describe an observed time series in order to understand the underlying causes

- **Time Series Forecasting**

Making predictions about the future is called extrapolation in the classical statistical handling of time series data

What makes Time Series Special?

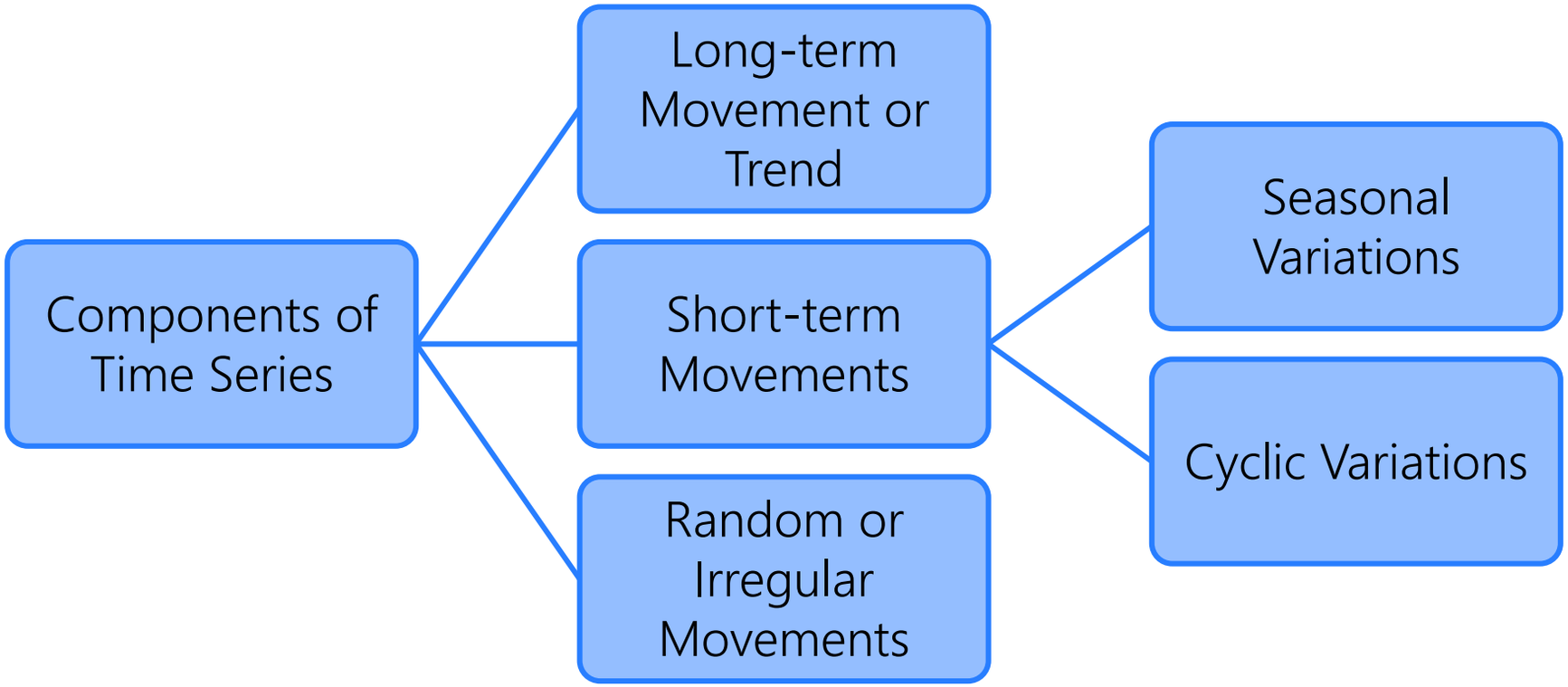
Components of Time Series:



What makes Time Series Special?

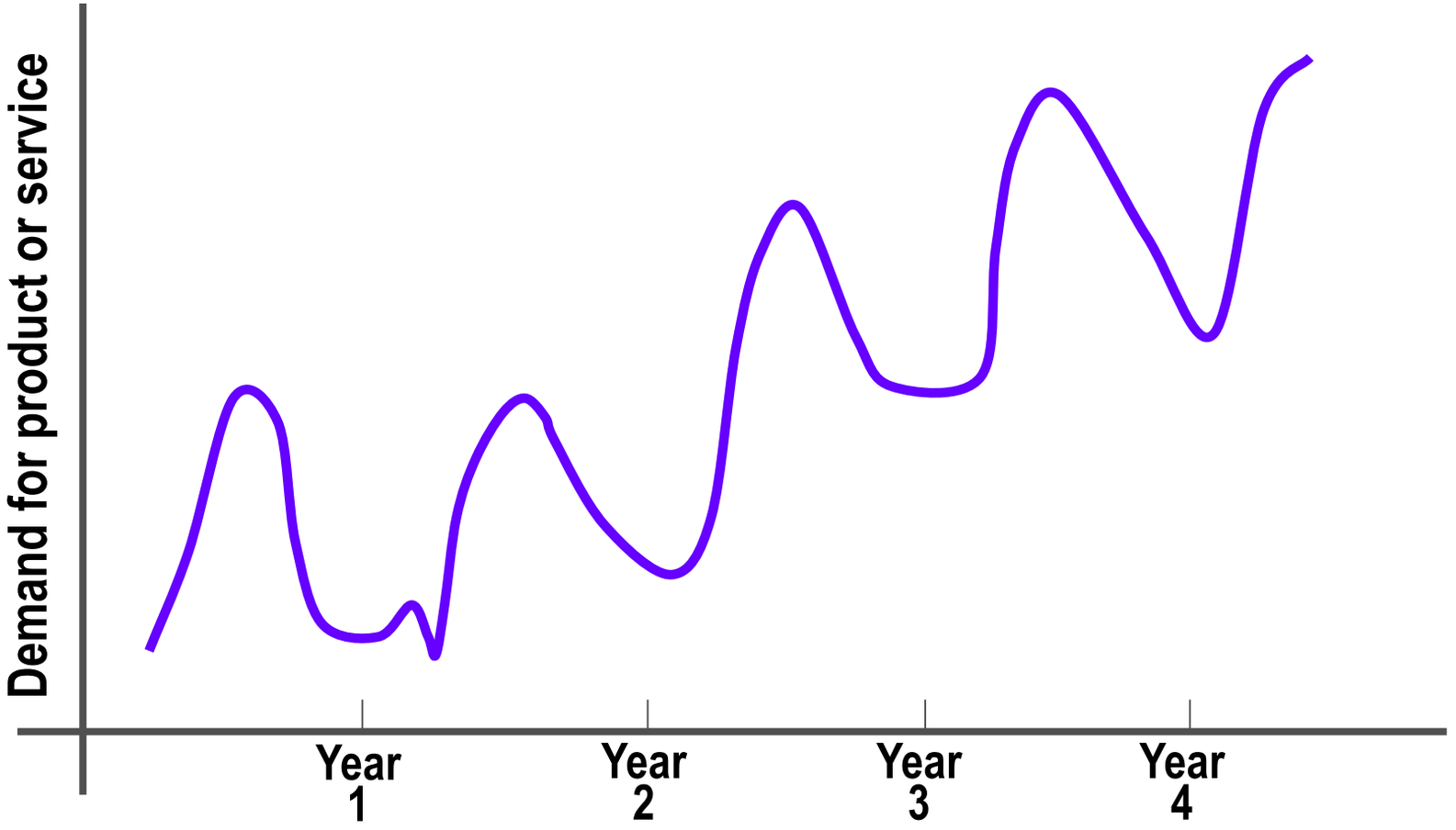
Components of Time Series:

@frlazzeri

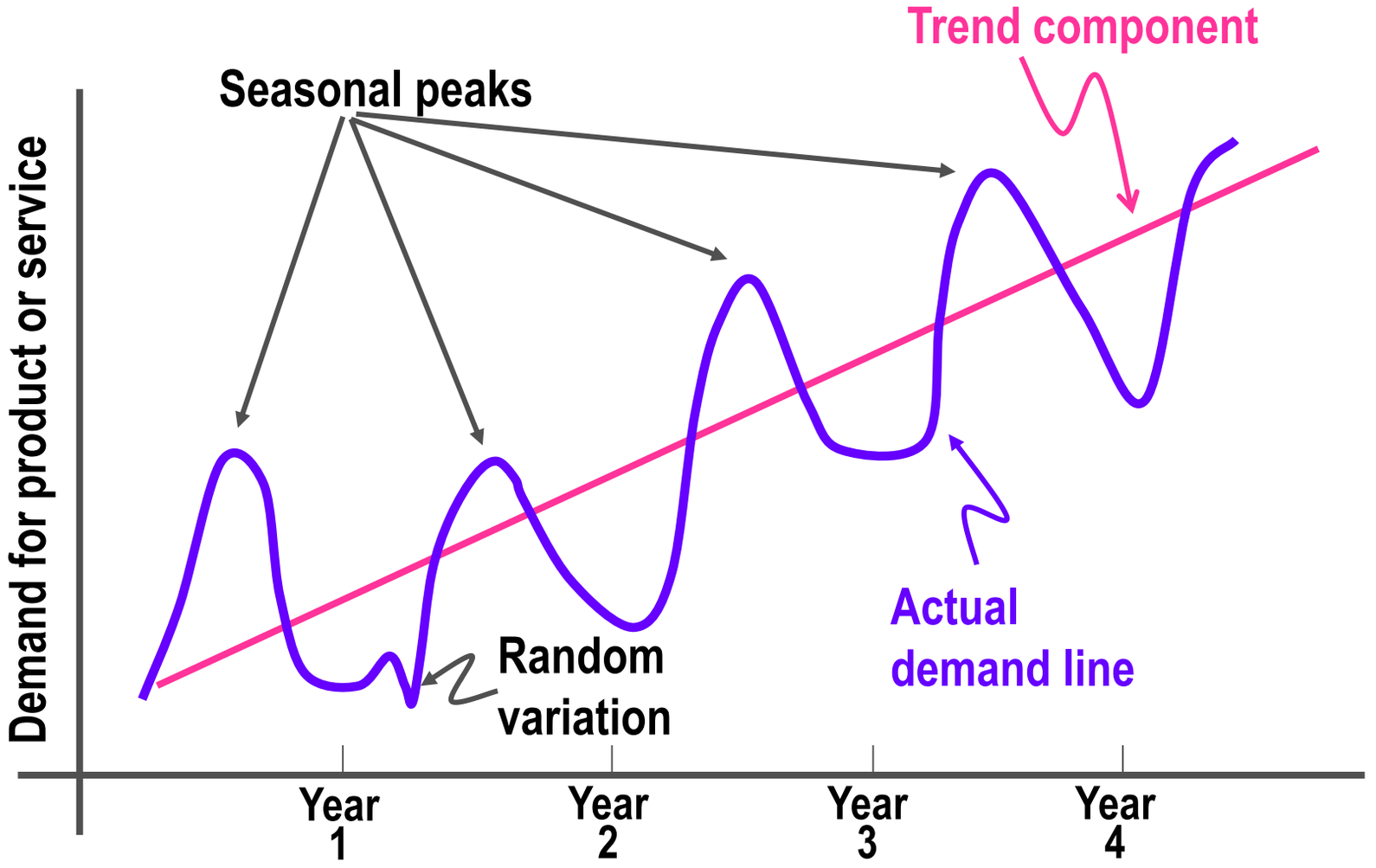


Example: Product Demand over Time

@frlazzeri



Example: Product Demand over Time



Example: Cyclical Variation

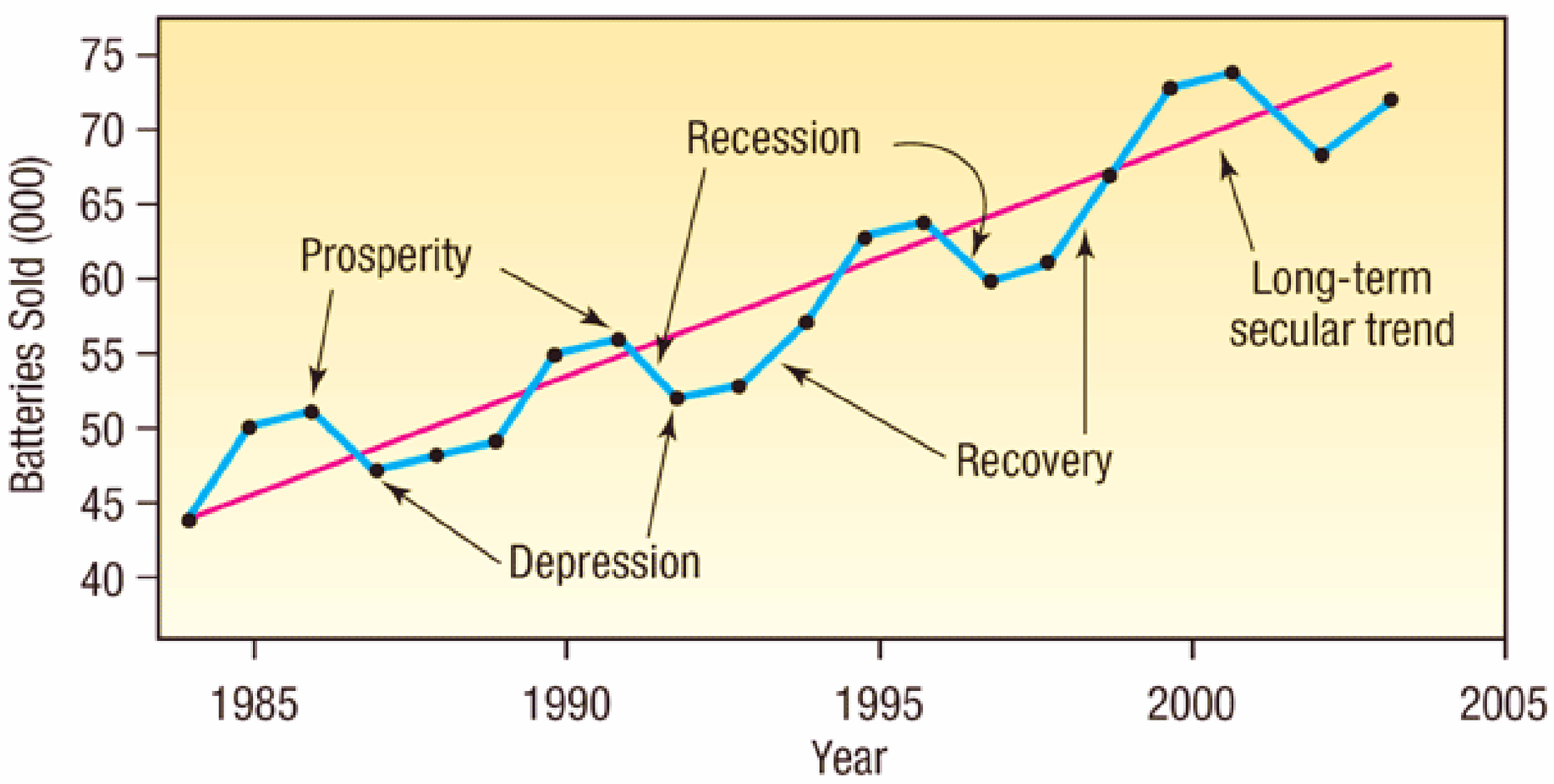


CHART 16–1 Batteries Sold by National Battery Retailers, Inc., from 1984 to 2004

Example: Seasonal Variation

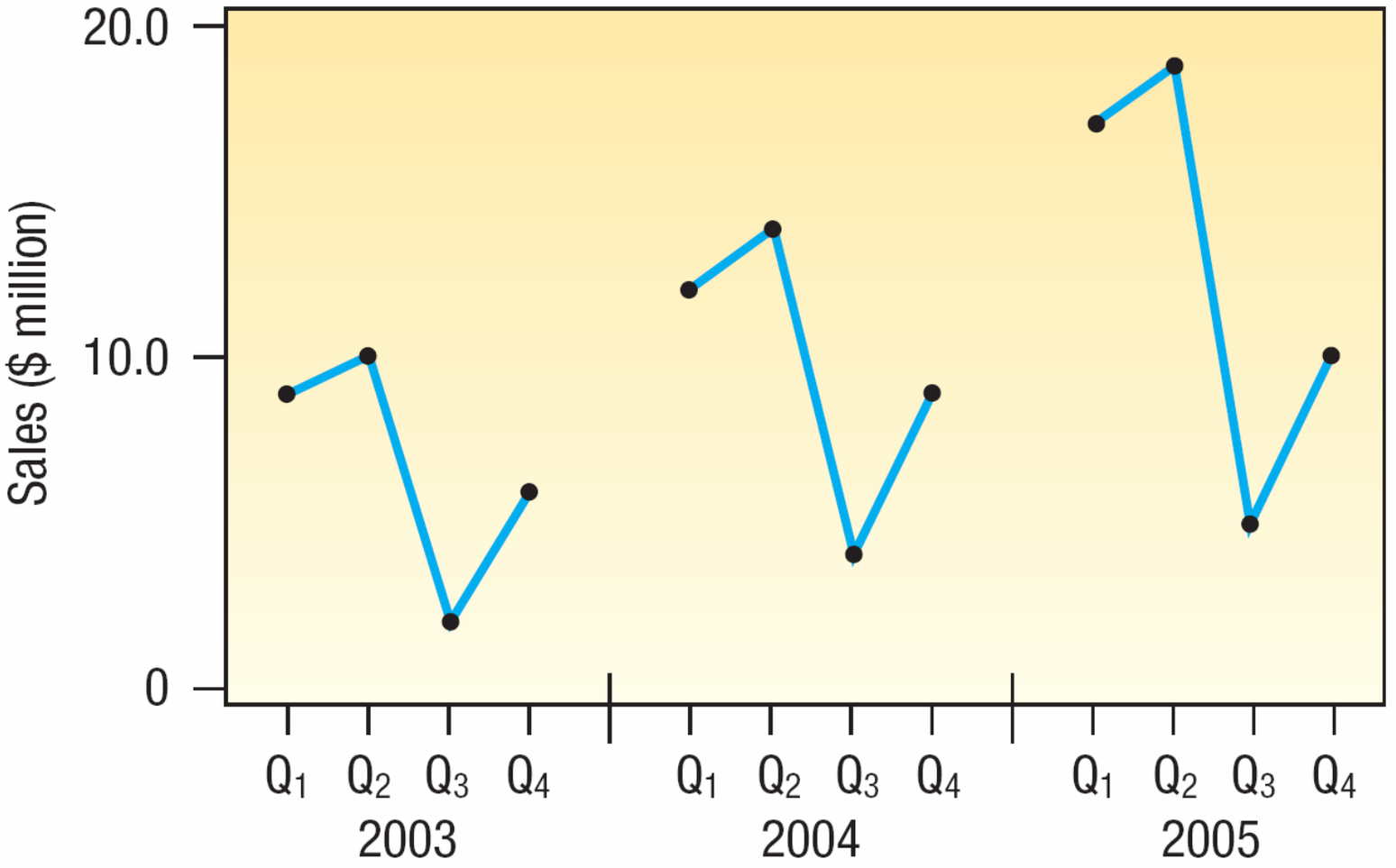


CHART 16–2 Sales of Baseball and Softball Equipment, Hercher Sporting Goods, 2003–2005 by Quarter

What Makes Time Series Special

Concerns of Forecasting:

- How much **data** do you have available and are you able to gather it all together?
- What is the **time horizon** of predictions that is required? Short, medium or long term?
- Can forecasts be updated frequently over time or must they be made once and remain **static**?
- At what **temporal frequency** are forecasts required?

Mathematical Model for Time Series

@frlazzeri

$y_t = f(t)$

$t: t_1, t_2, t_3, \ldots, t_n$

$y_t: y_{t1}, y_{t2}, y_{t3}, \ldots, y_{tn}$

Additive Model for Time Series Analysis

$$y_t = T_t + S_t + C_t + R_t.$$

Multiplicative Model for Time Series Analysis

$$y_t = T_t \times S_t \times C_t \times R_t$$

Mixed models

$$y_t = T_t + S_t \times C_t \times R_t$$

or $y_t = T_t \times C_t + S_t \times R_t$ etc.

How to Shape Time Series as Supervised Learning Problem: Sliding Window

Time Series Dataset

Time	Value
1	100
2	110
3	108
4	115
5	120

Time Series As Supervised Learning Problem

X	y
?	100
100	110
110	108
108	115
115	120
120	?

A key function to help transform time series data into a supervised learning problem is the ***Pandas shift() function****.*

Given a DataFrame, the shift() function can be used to create copies of columns that are pushed forward (rows of NaN values added to the front) or pulled back (rows of NaN values added to the end).

@frlazzeri

How to Shape Time Series as Supervised Learning Problem: Multivariate Time Series

Multivariate Time Series Dataset

Time	Value 1	Value 2
1	0.2	88
2	0.5	89
3	0.7	87
4	0.4	88
5	1.0	90

Multivariate Time Series As Supervised Learning Problem

X1	X2	X3	y
?	?	0.2	88
0.2	88	0.5	89
0.5	89	0.7	87
0.7	87	0.4	88
0.4	88	1.0	90
1.0	90	?	?

How to Shape Time Series as Supervised Learning Problem: Sliding Window With Multiple Steps

Univariate Time Series Dataset

Time	Value
1	100
2	110
3	108
4	115
5	120

Univariate Time Series As Multi-Step Supervised Learning

X1	y1	y2
?	100	110
100	110	108
110	108	115
108	115	120
115	120	?
120	?	?

How to perform feature engineering for time series data

Time Series →

- Time #1, Value 1
- Time #2, Value 2
- Time #3, Value 3

Time Series as a Supervised Learning Problem →

- Input #1, Output #1
- Input #2, Output #2
- Input #3, Output #3

- **Date Time Features**: these are components of the time step itself for each observation.
- **Lag Features**: these are values at prior time steps.
- **Window Features**: these are a summary of values over a fixed window of prior time steps.

How to Load and Handle Time Series in Pandas

- How to load your time series dataset from a CSV file
- How to peek at the loaded data and query using date-times
- How to calculate and review summary statistics
- How to perform feature engineering for time series

@frlazzeri

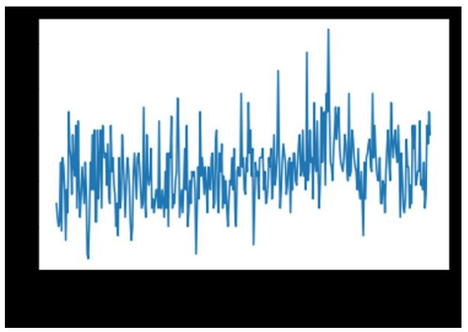
How to Load and Handle Time Series in Pandas

Concept	Scalar Class	Array Class	Pandas Data Type	Primary Creation Method
Date times	Timestamp	DatetimeIndex	datetime64[ns] or datetime64[ns, tz]	to_datetime or date_range
Time deltas	Timedelta	TimedeltaIndex	timedelta64[ns]	to_timedelta or timedelta_range
Time spans	Period	PeriodIndex	period[freq]	Period or period_range
Date offsets	DateOffset	None	None	DateOffset

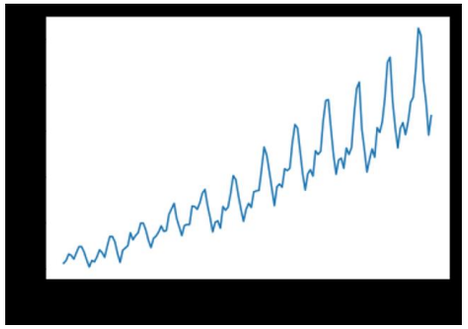
http://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html

Stationarity of a Time Series

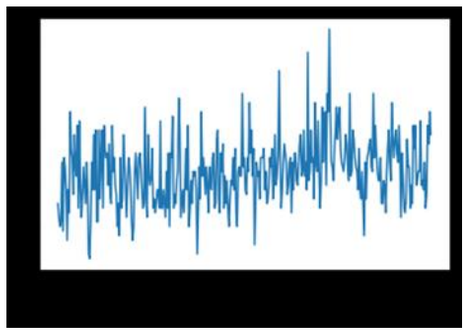
The observations in a stationary time series are not dependent on time



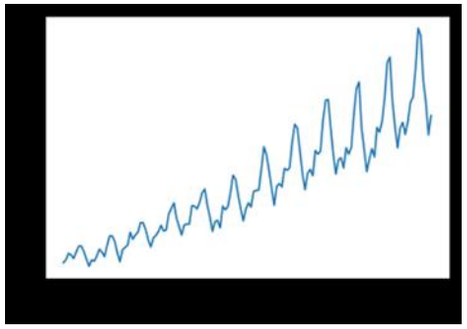
Time series are stationary if they do not have trend or seasonal effects



Summary statistics calculated on the time series are consistent over time, like the mean or the variance of the observations



When a time series is stationary, it can be easier to model



How to Check Stationarity of a Time Series

Look at Plots

- You can review a time series plot of your data and visually check if there are any obvious trends or seasonality

Summary Statistics

- You can review the summary statistics for your data for seasons or random partitions and check for obvious or significant differences

Statistical Tests

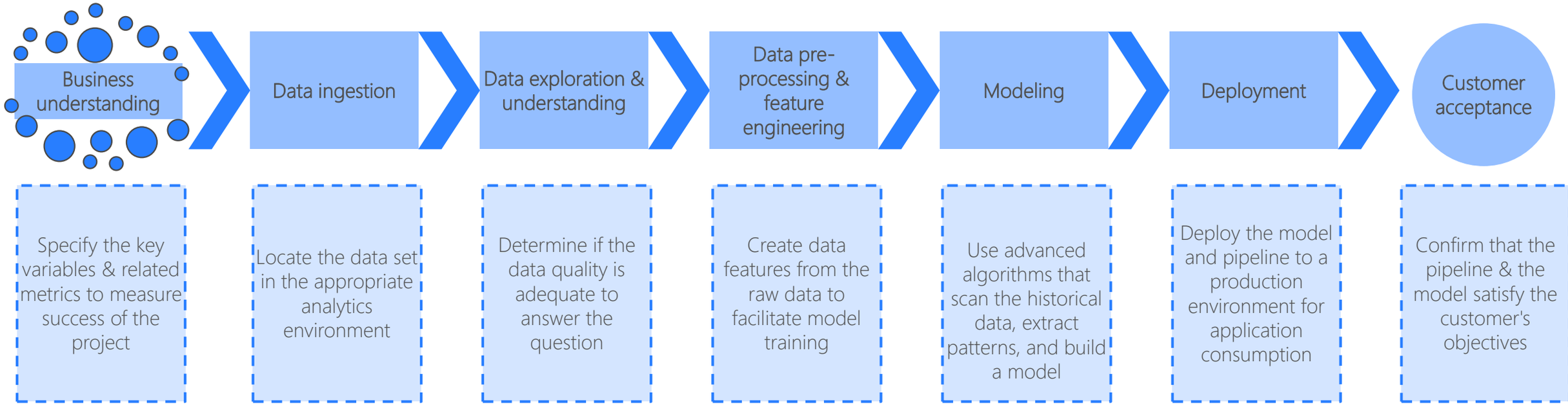
- You can use statistical tests to check if the expectations of stationarity are met or have been violated

How to Make a Time Series Stationary

- Difference Transform
  - difference(t) = observation(t) - observation(t-1)
  - Lag Difference
  - Difference Order
- Differencing to Remove Trends
  - difference() function
- Differencing to Remove Seasonality
  - Time of day, daily, weekly, monthly, annually, etc. etc.

@frlazzeri

Time Series Forecasting Framework



Exercise

@frlazzeri

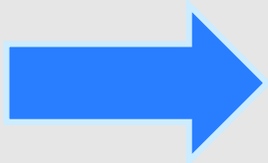
1. Go to: https://notebooks.azure.com/frlazzeri/projects/AIConferenceNYC

2. Once you are logged in to Azure Notebooks, on the top right, select '**Clone**' tab. Then type in any name you prefer for '**Project Name**' and '**Project ID**'. Once you have filled in all boxes, click '**Clone**'. Wait till you see a list of files cloned.

3. On the top left, select '**Free Compute**' as compute target option from the dropdown menu.

4. Make sure you see '**Python 3.6**' kernel on the top right for your notebook environment. If not, you can select '**Kernel**', then '**Change kernel**' to make changes.

5. Run each cell in the notebook by click '**Run**' on top. This notebook will download sample data to your environment and visualize the data. Wait and make sure you can see all the visualizations. Now you are all set!



Module_1_Intro to TS Forecasting

Classical methods for time series forecasting

- Simple Exponential Smoothing (SES)
- Autoregression (AR)
- Moving average (MA)
- Autoregressive moving average (ARMA)
- Autoregressive integrated moving average (ARIMA)

11 different classical time series forecasting methods (and they are not ALL!)

@frlazzeri

1. Autoregression (AR)
2. Moving Average (MA)
3. Autoregressive Moving Average (ARMA)
4. Autoregressive Integrated Moving Average (ARIMA)
5. Seasonal Autoregressive Integrated Moving-Average (SARIMA)
6. Seasonal Autoregressive Integrated Moving-Average with Exogenous Regressors (SARIMAX)
7. Vector Autoregression (VAR)
8. Vector Autoregression Moving-Average (VARMA)
9. Vector Autoregression Moving-Average with Exogenous Regressors (VARMAX)
10. Simple Exponential Smoothing (SES)
11. Holt Winter's Exponential Smoothing (HWES)

Traditional time series forecasting models

@frlazzeri

1. Simple Exponential Smoothing

Like an average model, but the importance/weight decreases exponentially as observations get older

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \cdots$$

2. Autoregressive integrated moving average (ARIMA)

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q} + e_t$$

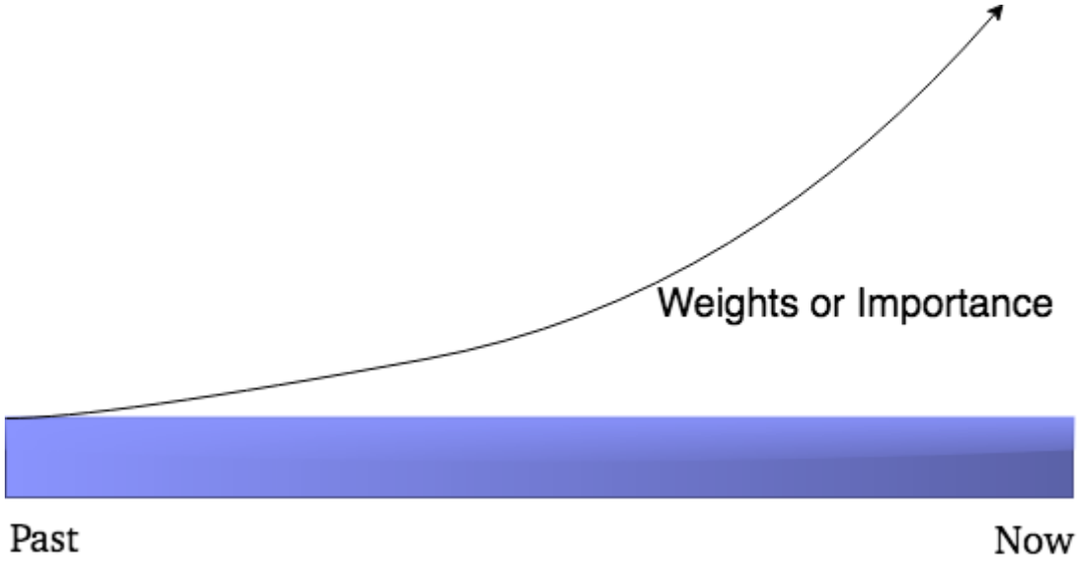
a) Autoregressive Model (AR) $y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + e_t$

b) Moving average model (MA) $y_t = c + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q}$

Simple Exponential Smoothing (SES)

@frlazzeri

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \cdots$$



- Exponential smoothing example:

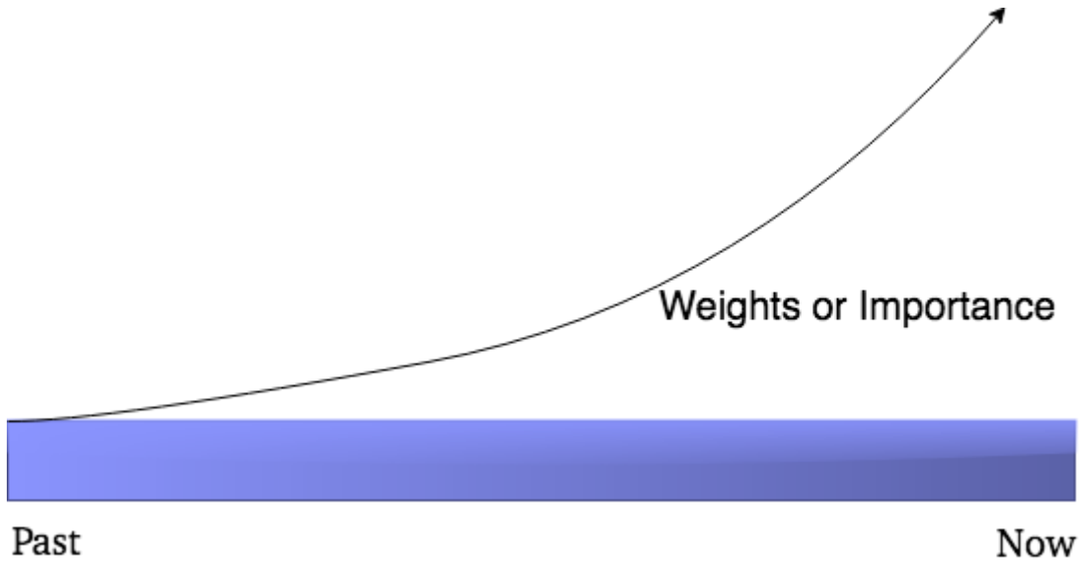
"I predicted to sell 10 units at the beginning of yesterday; At the end of yesterday, I found out I sold in fact 8 units.

*So, I will adjust the forecast of 10 (yesterday's forecast) by adding adjusted error (α * error). This will compensate over (under) forecast of yesterday".*

Simple Exponential Smoothing (SES)

@frlazzeri

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \cdots$$

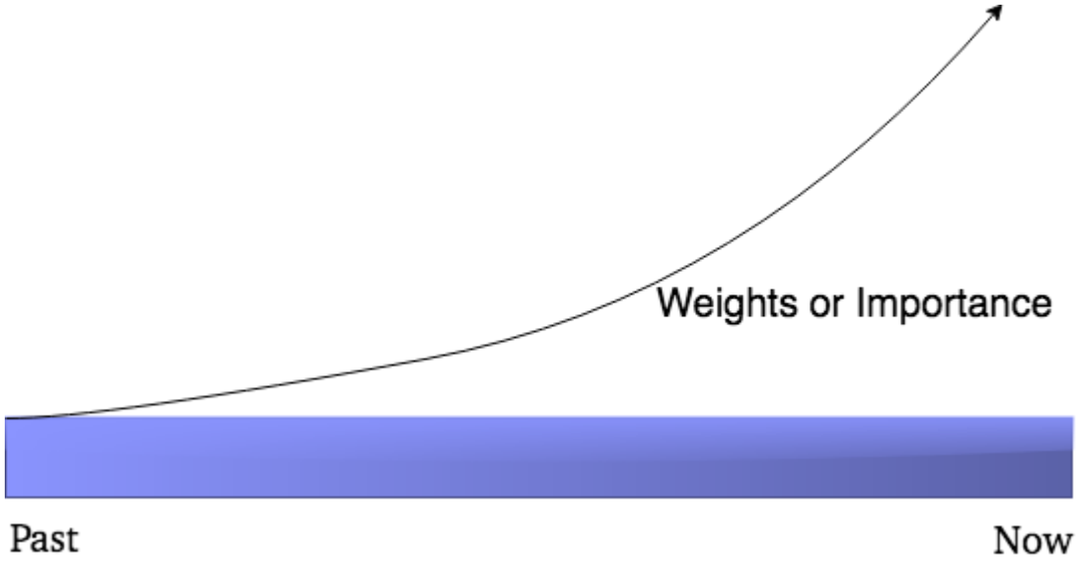


- SES methods are based on the assumption that a prediction is a weighted sum of past observations, but the model explicitly uses an **exponentially decreasing weight** for past observations
- Specifically, past observations are weighted with a **geometrically decreasing ratio**

Simple Exponential Smoothing (SES)

@frlazzeri

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \cdots$$



- When to use?

Few data points, Irregular data, No seasonality or trend.

- Math behind

Just keep in mind that SES only has one component called level (with a smoothing parameter denoted as "alpha" below). It is a weighted average of the previous level and the current observation:

$$p_t = l_t$$
$$l_t = \alpha y_t + (1-\alpha) l_{t-1}$$

Simple Exponential Smoothing

- How to choose α
  - depends on the emphasis you want to place on the most recent data
- Increasing α makes forecast more sensitive to recent data

Simple Exponential Smoothing

Properties of Simple Exponential Smoothing

- Widely used and successful model
- Requires very little data
- Larger α, more responsive forecast; Smaller α, smoother forecast

  "best" α can be found by a solver package
- Suitable for relatively stable time series

Exponential Smoothing

@frlazzeri

- Concept is *simple*!
  - Make a forecast, ***any*** forecast
  - Compare it to the actual
  - Next forecast is
    - ***Previous*** forecast plus an adjustment
    - Adjustment is fraction of previous forecast error
  - Essentially
    - Not really forecast as a function of ***time***
    - Instead, forecast as a function of ***previous actual and forecasted value***

Exponential Smoothing

- Assumes the most recent observations have the highest predictive value
  - gives more weight to recent time periods

$$F_{t+1} = F_t + \alpha(A_t - F_t)$$

e_t

F_{t+1} = Forecast value for time $t+1$

A_t = Actual value at time t

α = Smoothing constant

Need initial forecast F_t to start.

Simple Exponential Smoothing (SES)

@frlazzeri

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \cdots$$

```
# single exponential smoothing
...
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
# prepare data
data = ...
# create class
model = SimpleExpSmoothing(data)
# fit model
model_fit = model.fit(...)
# make prediction
yhat = model_fit.predict(...)
```

ExponentialSmoothing Statsmodels class

- Trend
- Damped
- Seasonal
- Seasonal periods

+

- smoothing_level (alpha)
- smoothing_slope (beta)
- smoothing_seasonal(gamma)
- damping_slope (phi)

More Information

- statsmodels.tsa.holtwinters.SimpleExpSmoothing API
- statsmodels.tsa.holtwinters.HoltWintersResults API
- Exponential smoothing on Wikipedia

Traditional time series forecasting models

@frlazzeri

1. Simple Exponential Smoothing

Like an average model, but the importance/weight decrease exponentially as observations older

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \cdots$$

2. Autoregressive integrated moving average (ARIMA)

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q} + e_t$$

a) Autoregressive Model (AR) $y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + e_t$

b) Moving average model (MA) $y_t = c + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q}$

Autoregression

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + e_t$$

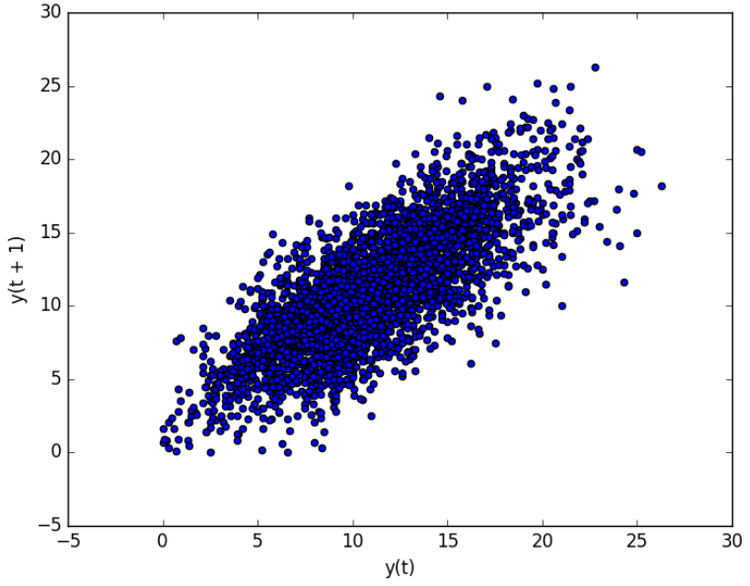
- Observations from previous time steps as input to a regression equation to predict the value at the next time step

- Regression models:

$$Yhat = b0 + (b1 \times X1)$$

$$X(t+1) = b0 + \left(b1 \times X(t)\right) + \left(b2 \times X(t-1)\right)$$

- Autocorrelation



Autoregression

@frlazzeri

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + e_t$$

```
# AR example
from statsmodels.tsa.ar_model import AR
from random import random
# contrived dataset
data = [x + random() for x in range(1, 100)]
# fit model
model = AR(data)
model_fit = model.fit()
# make prediction
yhat = model_fit.predict(len(data), len(data))
print(yhat)
```

- **LinearRegression class** in scikit-learn and manually specify the lag input variables to use
- **Statsmodels** library provides an autoregression model that automatically selects an appropriate lag value using statistical tests and trains a linear regression model
- It is provided in the **AR class** + fit()
- This returns an **ARResult** object

More Information
- statsmodels.tsa.ar_model.AR API
- statsmodels.tsa.ar_model.ARResults API
- Autoregressive model on Wikipedia

Exercise

@frlazzeri

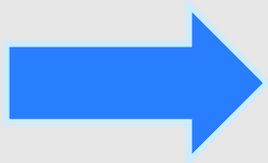
1. Go to: https://notebooks.azure.com/frlazzeri/projects/AIConferenceNYC

2. Once you are logged in to Azure Notebooks, on the top right, select '**Clone**' tab. Then type in any name you prefer for '**Project Name**' and '**Project ID**'. Once you have filled in all boxes, click '**Clone**'. Wait till you see a list of files cloned.

3. On the top left, select '**Free Compute**' as compute target option from the dropdown menu.

4. Make sure you see '**Python 3.6**' kernel on the top right for your notebook environment. If not, you can select '**Kernel**', then '**Change kernel**' to make changes.

5. Run each cell in the notebook by click '**Run**' on top. This notebook will download sample data to your environment and visualize the data. Wait and make sure you can see all the visualizations. Now you are all set!



Module_1_Intro to TS Forecasting

Moving average

@frlazzeri

$$y_t = c + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q}$$

- The **residual errors** from forecasts on a time series provide another source of information that we can model

- The difference between what was expected and what was predicted is called the residual error:

$$residual\_error = expected - predicted$$

- Residual errors from a time series can have **temporal structure** like trends, bias, and seasonality

- A simple and effective model of residual error is an **autoregression**

The Moving Average Method

- Useful in smoothing time series to see its trend
- Basic method used in measuring seasonal fluctuation
- Applicable when time series follows fairly linear trend that have definite rhythmic pattern
- Assumes average is a good estimator of future

$$F_{t+1} = \frac{A_t + A_{t-1} + A_{t-2} + \ldots + A_{t-n+1}}{n}$$

F_{t+1} = Forecast for the upcoming period, t+1
n = Number of periods to be averaged
A_t = Actual occurrence in period t

Simple Moving Average

$$F_{t+1} = \frac{A_t + A_{t-1} + A_{t-2} + \ldots + A_{t-n+1}}{n}$$

You're manager in Amazon's electronics department. You want to forecast ipod sales for months 4-6 using a 3-period moving average.

Month	Sales (000)
1	4
2	6
3	5
4	?
5	?
6	?

Simple Moving Average

$$F_{t+1} = \frac{A_t + A_{t-1} + A_{t-2} + \ldots + A_{t-n+1}}{n}$$

You're manager in Amazon's electronics department. You want to forecast ipod sales for months 4-6 using a 3-period moving average.

Month	Sales (000)	Moving Average (n=3)
1	4	NA
2	6	NA
3	5	NA
4	?	(4+6+5)/3=5
5	?	
6	?	

Simple Moving Average

@frlazzeri

What if ipod sales were actually 3 in month 4

Month	Sales (000)	Moving Average (n=3)
1	4	NA
2	6	NA
3	5	NA
4	3	5
5	?	
6	?	

Simple Moving Average

Forecast for Month 5?

@frlazzeri

Month	Sales (000)	Moving Average (n=3)
1	4	NA
2	6	NA
3	5	NA
4	3	5
5	?	(6+5+3)/3=4.667
6	?	

Simple Moving Average

Actual Demand for Month 5 = 7

@frlazzeri

Month	Sales (000)	Moving Average (n=3)
1	4	NA
2	6	NA
3	5	NA
4	3	5
5	? 7	4.667
6	?	

Simple Moving Average
Forecast for Month 6?

@frlazzeri

Month	Sales (000)	Moving Average (n=3)
1	4	NA
2	6	NA
3	5	NA
4	3	5
5	7	4.667
6	?	(5+3+7)/3=5

Moving average

@frlazzeri

$$y_t = c + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q}$$

```
# MA example
from statsmodels.tsa.arima_model import ARMA
from random import random
# contrived dataset
data = [x + random() for x in range(1, 100)]
# fit model
model = ARMA(data, order=(0, 1))
model_fit = model.fit(disp=False)
# make prediction
yhat = model_fit.predict(len(data), len(data))
print(yhat)
```

- A **moving average** model is different from calculating the moving average of the time series.
- The notation for the model involves specifying the order of the model q as a parameter to the MA function, e.g. MA(q). For example, MA(1) is a first-order moving average model.
- The method is suitable for **univariate** time series without trend and seasonal components.

More Information

- statsmodels.tsa.arima_model.ARMA API
- statsmodels.tsa.arima_model.ARMAResults API
- Moving-average model on Wikipedia

Weighted Moving Average

- A simple moving average assigns the same weight to each observation in averaging
- Weighted moving average assigns different weights to each observation
- Most recent observation receives the most weight, and the weight decreases for older data values
- In either case, the sum of the weights = 1

Exercise

@frlazzeri

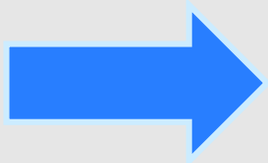
1. Go to: https://notebooks.azure.com/frlazzeri/projects/AIConferenceNYC

2. Once you are logged in to Azure Notebooks, on the top right, select '**Clone**' tab. Then type in any name you prefer for '**Project Name**' and '**Project ID**'. Once you have filled in all boxes, click '**Clone**'. Wait till you see a list of files cloned.

3. On the top left, select '**Free Compute**' as compute target option from the dropdown menu.

4. Make sure you see '**Python 3.6**' kernel on the top right for your notebook environment. If not, you can select '**Kernel**', then '**Change kernel**' to make changes.

5. Run each cell in the notebook by click '**Run**' on top. This notebook will download sample data to your environment and visualize the data. Wait and make sure you can see all the visualizations. Now you are all set!



Module_1_Intro to TS Forecasting

Autoregressive integrated moving average

@frlazzeri

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q} + e_t$$

AR

- Autoregression. A model that uses the dependent relationship between an observation and some number of lagged observations

I

- Integrated. The use of differencing of raw observations (e.g. subtracting an observation from an observation at the previous time step) in order to make the time series stationary

MA

- Moving Average. A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations

Autoregressive integrated moving average

@frlazzeri

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q} + e_t$$

Each component functions as a parameter with a standard notation. For ARIMA models, a standard notation would be ARIMA with p, d, and q, where integer values substitute for the parameters to indicate the type of ARIMA model used. The parameters can be defined as:

- p: The number of lag observations included in the model, also called the lag order
- d: The number of times that the raw observations are differenced, also called the degree of differencing
- q: The size of the moving average window, also called the order of moving average

Autoregressive integrated moving average

@frlazzeri

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q} + e_t$$

An ARIMA model can be created using the Statsmodels library as follows:

```
# ARIMA example
from statsmodels.tsa.arima_model import ARIMA
from random import random
# contrived dataset
data = [x + random() for x in range(1, 100)]
# fit model
model = ARIMA(data, order=(1, 1, 1))
model_fit = model.fit(disp=False)
# make prediction
yhat = model_fit.predict(len(data), len(data), typ='levels')
print(yhat)
```

1. Define the model by calling **ARIMA()** and passing in the p, d, and q parameters
2. The model is prepared on the training data by calling the **fit()** function
3. Predictions can be made by calling the **predict()** function and specifying the index of the time or times to be predicted

More Information

- statsmodels.tsa.arima_model.ARIMA API
- statsmodels.tsa.arima_model.ARIMAResults API
- Autoregressive integrated moving average on Wikipedia

Exercise

@frlazzeri

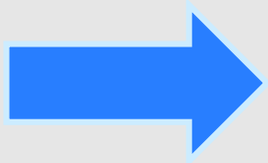
1. Go to: https://notebooks.azure.com/frlazzeri/projects/AIConferenceNYC

2. Once you are logged in to Azure Notebooks, on the top right, select '**Clone**' tab. Then type in any name you prefer for '**Project Name**' and '**Project ID**'. Once you have filled in all boxes, click '**Clone**'. Wait till you see a list of files cloned.

3. On the top left, select '**Free Compute**' as compute target option from the dropdown menu.

4. Make sure you see '**Python 3.6**' kernel on the top right for your notebook environment. If not, you can select '**Kernel**', then '**Change kernel**' to make changes.

5. Run each cell in the notebook by click '**Run**' on top. This notebook will download sample data to your environment and visualize the data. Wait and make sure you can see all the visualizations. Now you are all set!



Module_1_Intro to TS Forecasting

Microsoft

@frlazzeri

Module 2
Introduction to Azure ML Service

- Azure Machine Learning Service
- Wrangling big data with Data Prep
- Forecasting with Automated ML

Machine Learning on Azure

Domain Specific Pretrained Models
To reduce time to market

Vision | Speech | Language | Search

Familiar Data Science Tools
To simplify model development

PyCharm | Jupyter | Visual Studio Code | Command line

Popular Frameworks
To build machine learning and deep learning solutions

PyTorch | TensorFlow | Scikit-Learn | ONNX

Productive Services
To empower data science and development teams

Azure Databricks | Azure Machine Learning | Machine Learning VMs

Powerful Hardware
To accelerate deep learning

CPU | GPU | FPGA

From the Intelligent Cloud to the Intelligent Edge

@frlazzeri

Reviewing AI on Azure

AI apps & agents

Azure Bot Service
Azure Cognitive Services

Machine learning

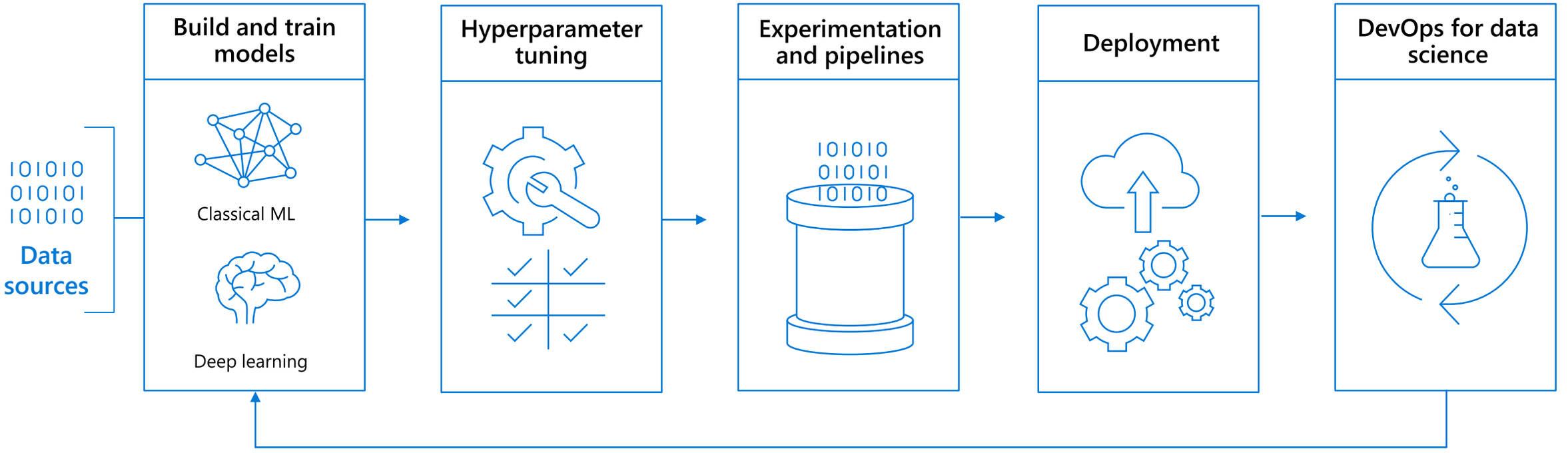
Azure Databricks
Azure Machine Learning

Knowledge mining

Azure Cognitive Search

Azure Machine Learning Service

Building blocks for a data science project



What can make this simple and streamlined?

Azure Machine Learning service

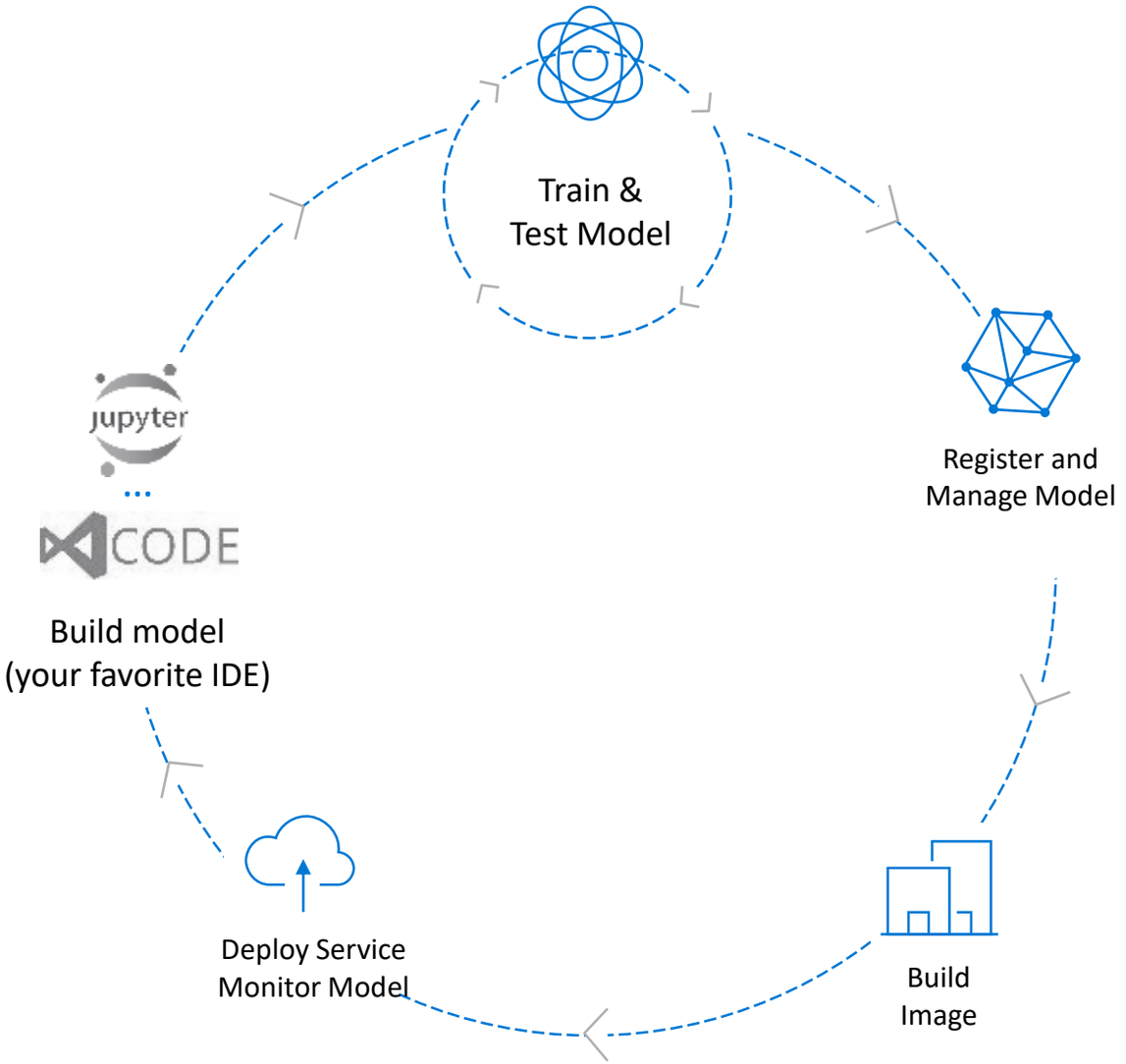
Set of Azure Cloud Services

+

Python SDK

Prepare

Prepare Data



Powerful Infrastructure

Accelerate deep learning

CPUs

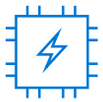
General purpose machine learning

D, F, L, M, H Series

GPUs

Deep learning

N Series



FPGAs

Specialized hardware accelerated deep learning

Project Brainwave

FPGA NEW UPDATES:

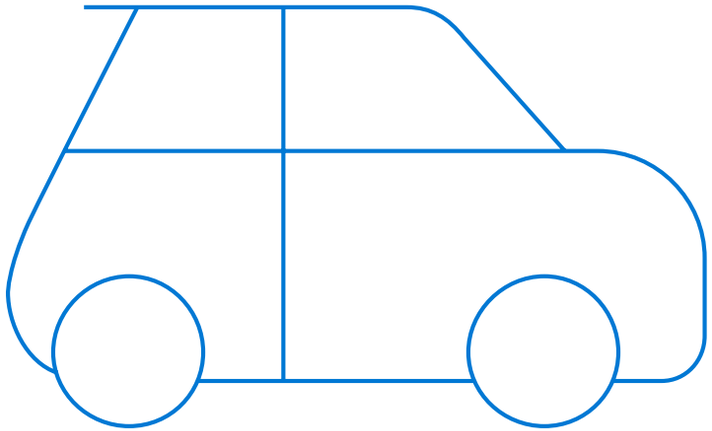
Support for image classification and recognition scenarios

ResNet 50, ResNet 152, VGG-16, SSD-VGG, DenseNet-121

How does AML service achieve the objective?

Building your own AI models

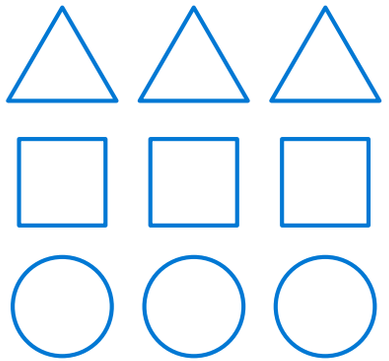
Transforming data into intelligence



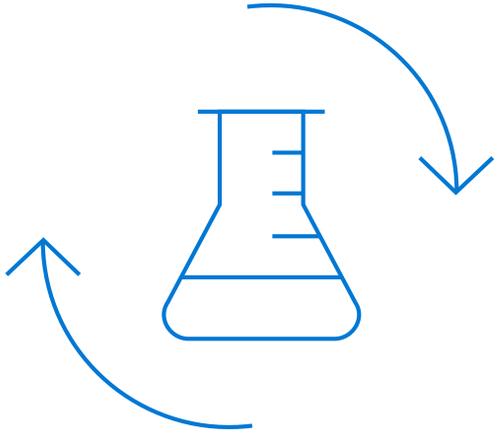
Q: How much is this car worth?

Building your own AI models

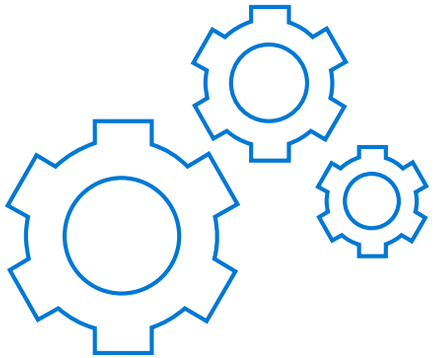
Transforming data into intelligence



Prepare data



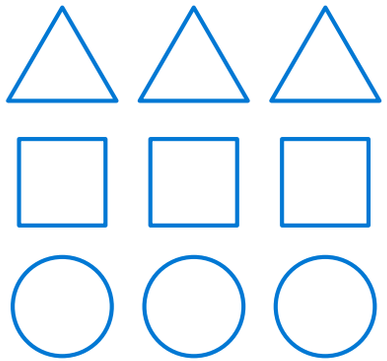
Build and train



Deploy

Building your own AI models

Transforming data into intelligence



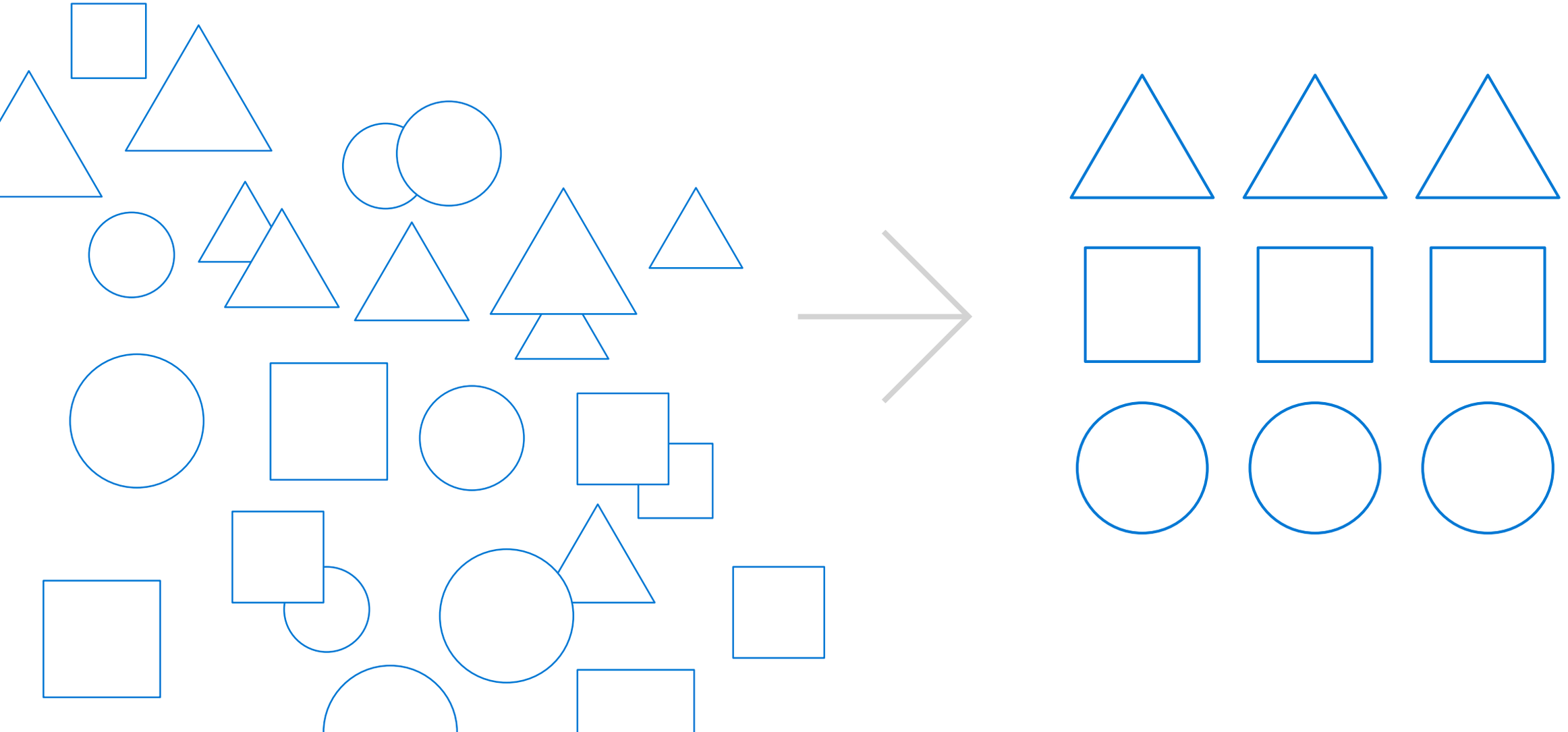
Prepare data

Build and train

Deploy

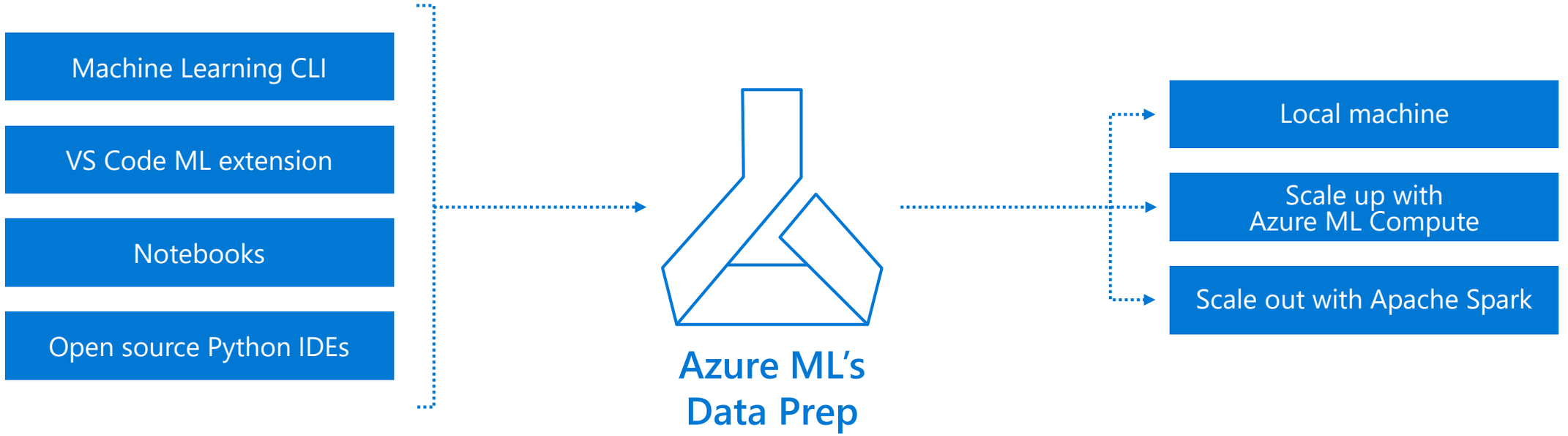
Building your own AI models

Step 1: Prepare data



Building your own AI models

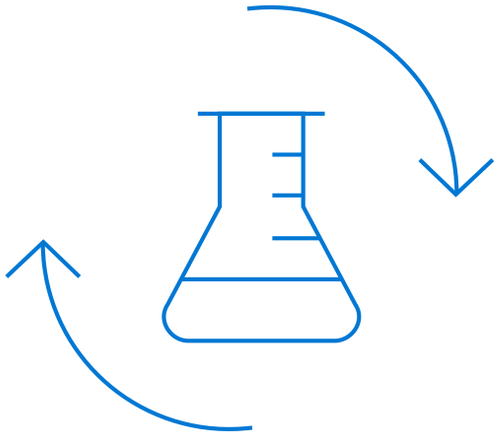
Prepare data anywhere



Building your own AI models

Transforming data into intelligence

Prepare data

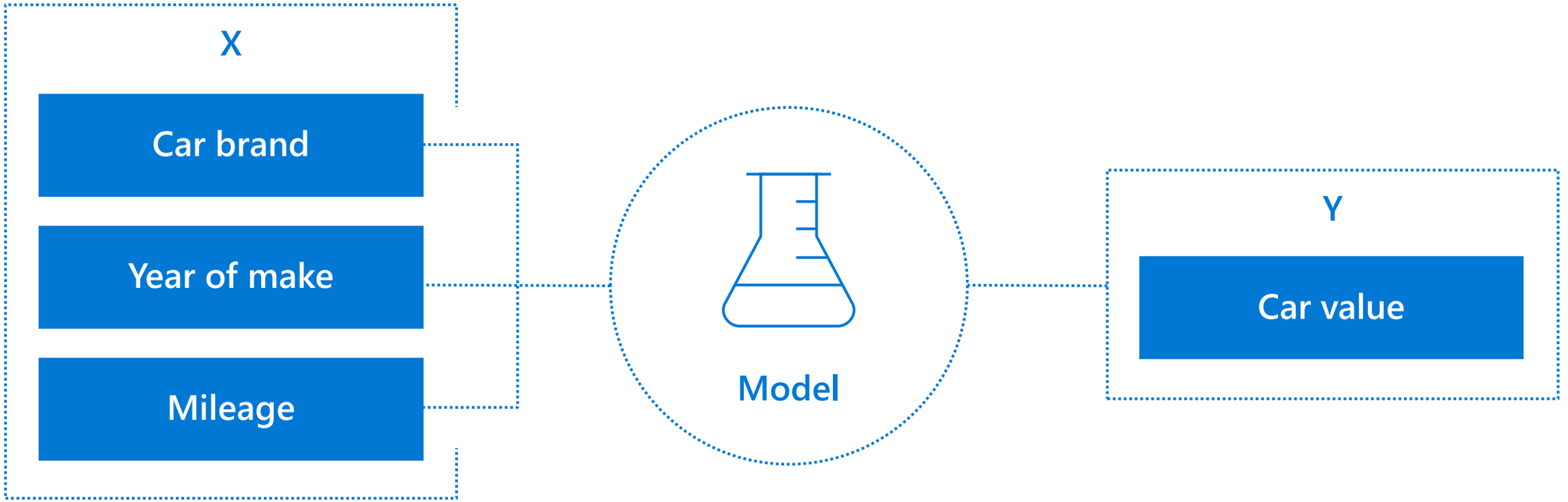


Build and train

Deploy

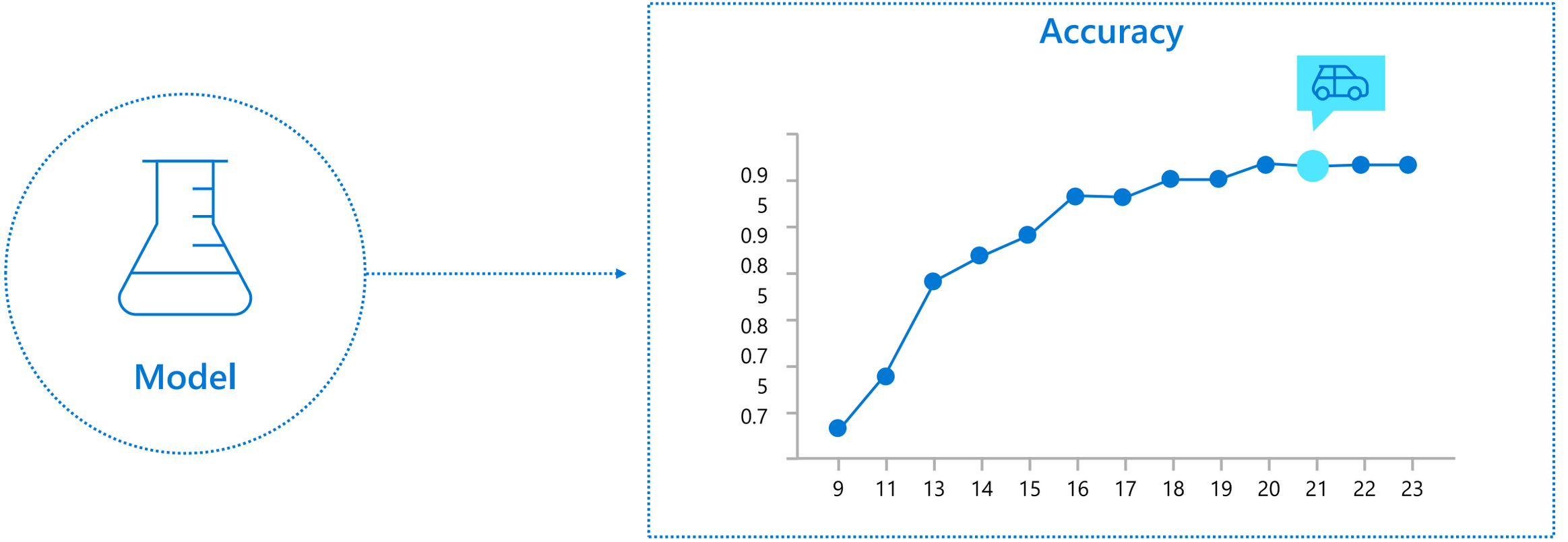
Building your own AI models

Step 2: Build and train models



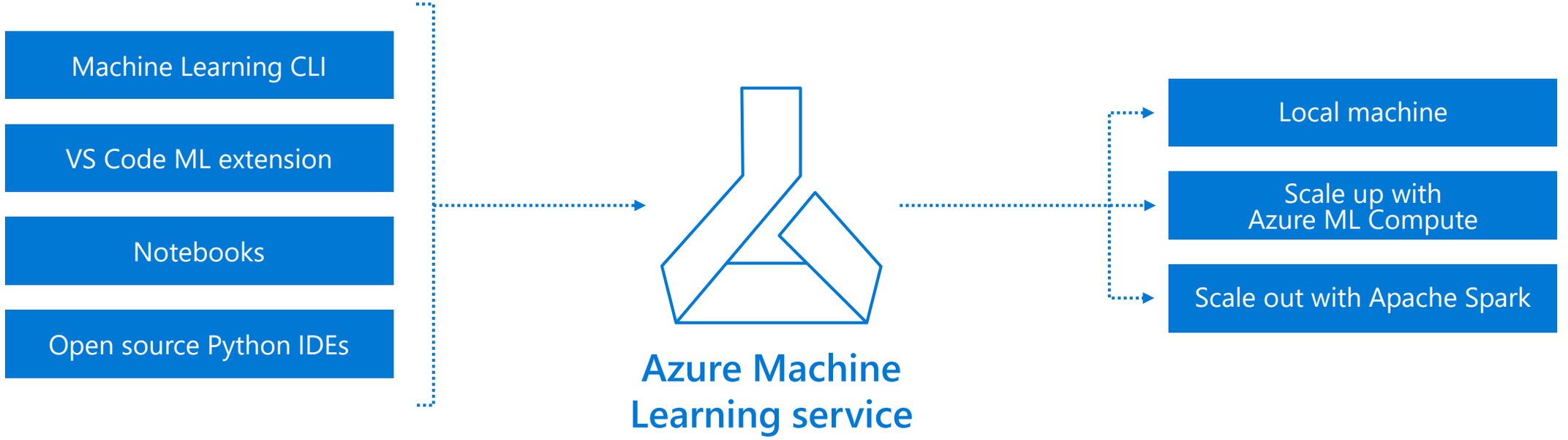
Building your own AI models

Step 2: Build and train models



Azure Machine Learning service

Build and train models anywhere



Building your own AI models

Transforming data into intelligence

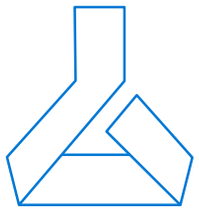
Prepare data

Build and train

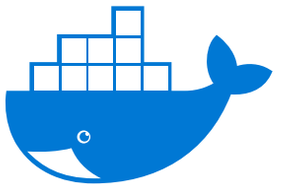
Deploy

Building your own AI models

Step 3: Deploy models



Azure Machine Learning Service

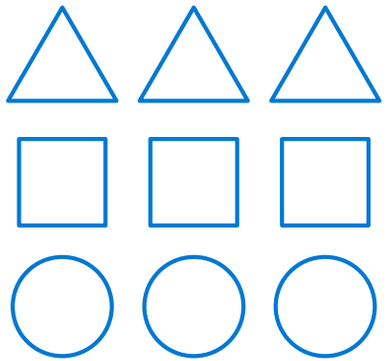


Docker

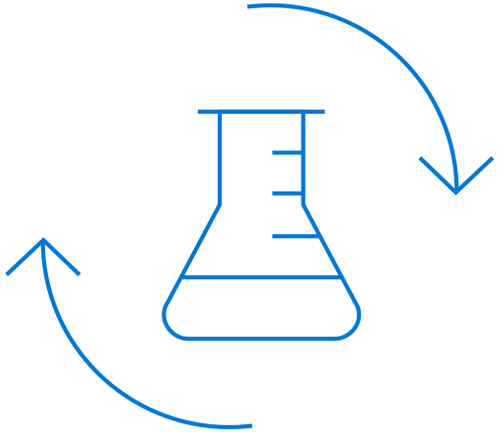
- On-prem deployment
- Azure IoT edge
- Azure Container Instance
- Azure Managed Kubernetes Service
- IoT accelerated targets
- Project Brainwave (FPGAs)

Building your own AI models

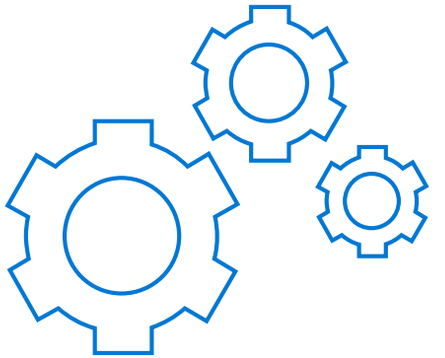
Transforming data into intelligence



Prepare data



Build and train



Deploy

Building your own AI models

Transforming data into intelligence

Azure SQL DB

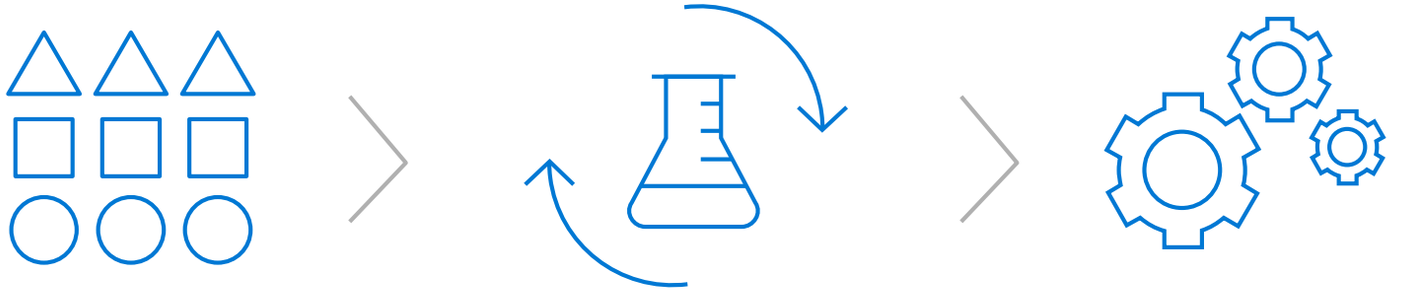
Azure Cosmos DB

Azure Data Warehouse

Azure Data Lake

Azure Blob Storage

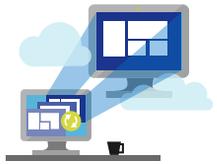
...



Azure Machine Learning service

Bring AI to everyone with an end-to-end, scalable, trusted platform

Boost your data science productivity



Increase your rate of experimentation

Deploy and manage your models everywhere

Built with your needs in mind

- Automated machine learning
- Managed compute
- Simple deployment
- DevOps for machine learning
- Support for open source frameworks
- Tool-agnostic Python SDK

Seamlessly integrated with the Azure Portfolio

Exercise – Experiment & Deploy

@frlazzeri

1. Go to: https://notebooks.azure.com/frlazzeri/projects/AIConferenceNYC

2. Once you are logged in to Azure Notebooks, on the top right, select '**Clone**' tab. Then type in any name you prefer for '**Project Name**' and '**Project ID**'. Once you have filled in all boxes, click '**Clone**'. Wait till you see a list of files cloned.

3. On the top left, select '**Free Compute**' as compute target option from the dropdown menu.

4. Make sure you see '**Python 3.6**' kernel on the top right for your notebook environment. If not, you can select '**Kernel**', then '**Change kernel**' to make changes.

5. Run each cell in the notebook by click '**Run**' on top. This notebook will download sample data to your environment and visualize the data. Wait and make sure you can see all the visualizations. Now you are all set!

Module_2_Get Started with AzureML

Configuration. Ipynb

01.run-experiment.ipynb

02.deploy-web-service.ipynb

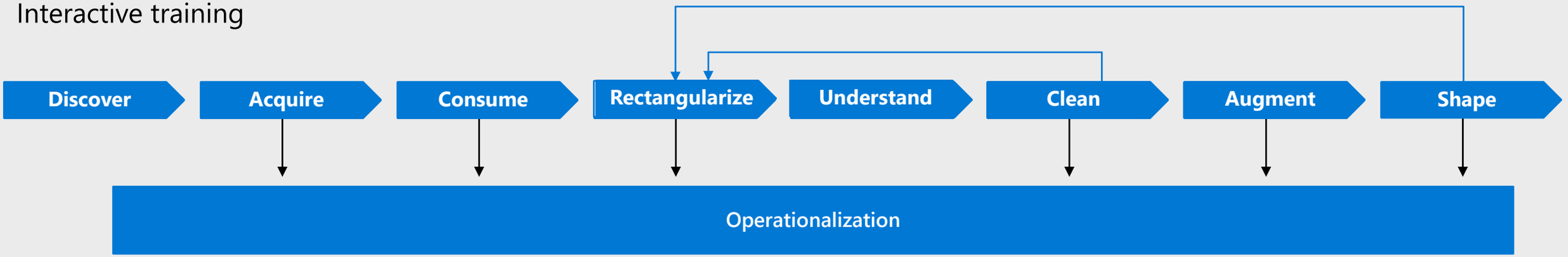
Data Prep

Garbage in, garbage out

Courtesy of xkcd: https://xkcd.com/1838/

Data wrangling lifecycle

Interactive training

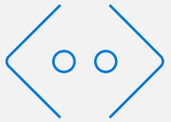


Retraining/scoring

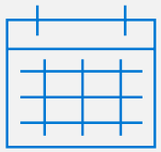
Pain points: data wrangling issues

- Understanding the semantics of data is difficult and time-consuming
- Merging data from different sources is a manual process
- Detecting, troubleshooting, and fixing errors is a high tax
- Custom code is always required
- Operationalization is challenging

Examples of manual, non-scalable work



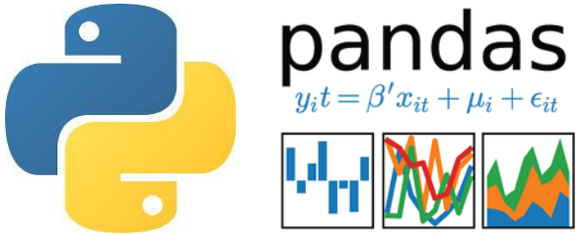
Data formatting

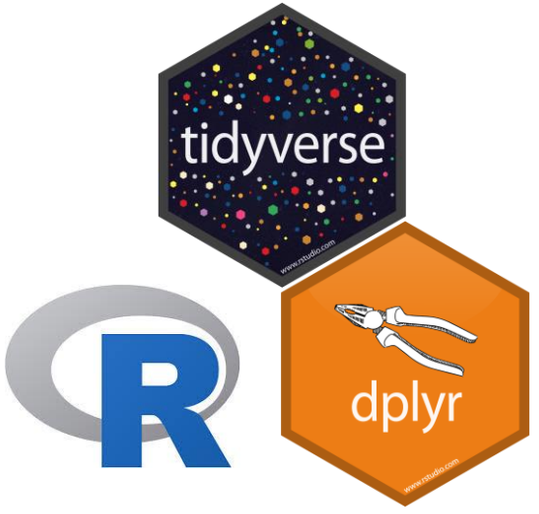


Dealing with dates

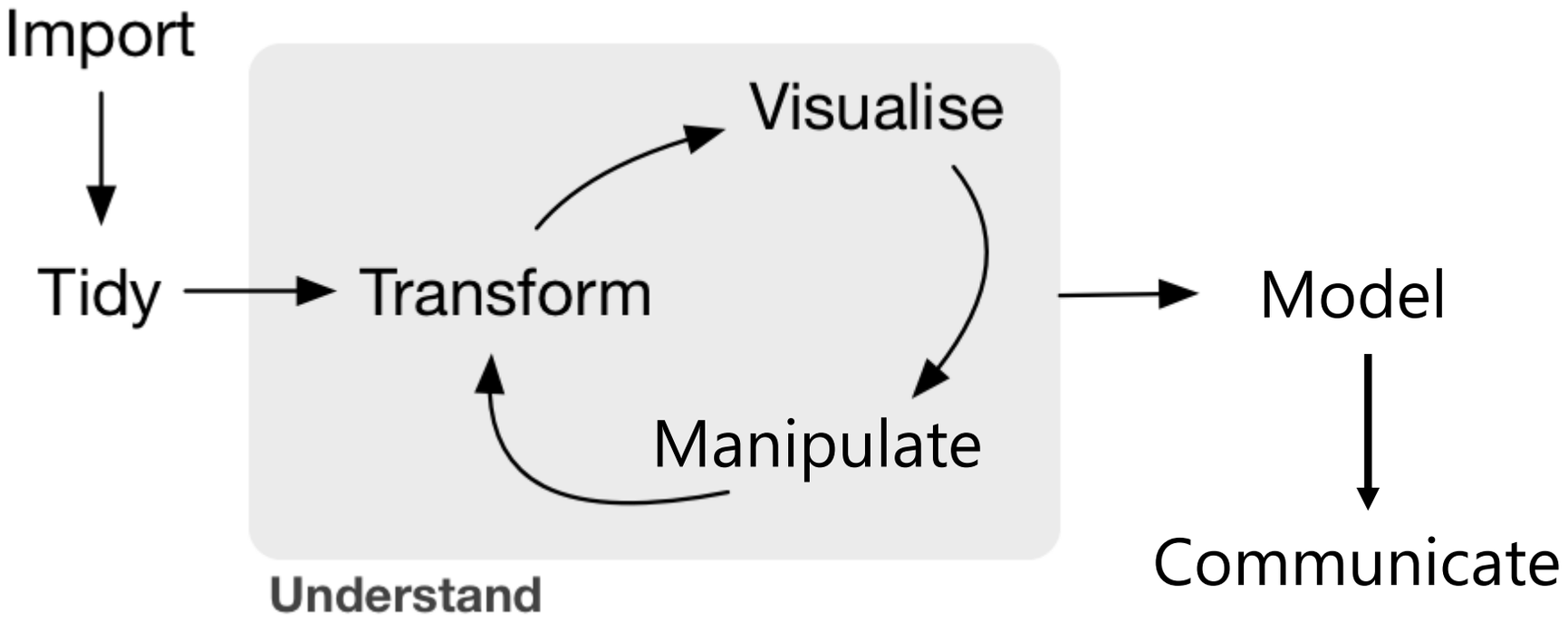
'Rectangularizing' data

Pain points: achieving scale





Design philosophy

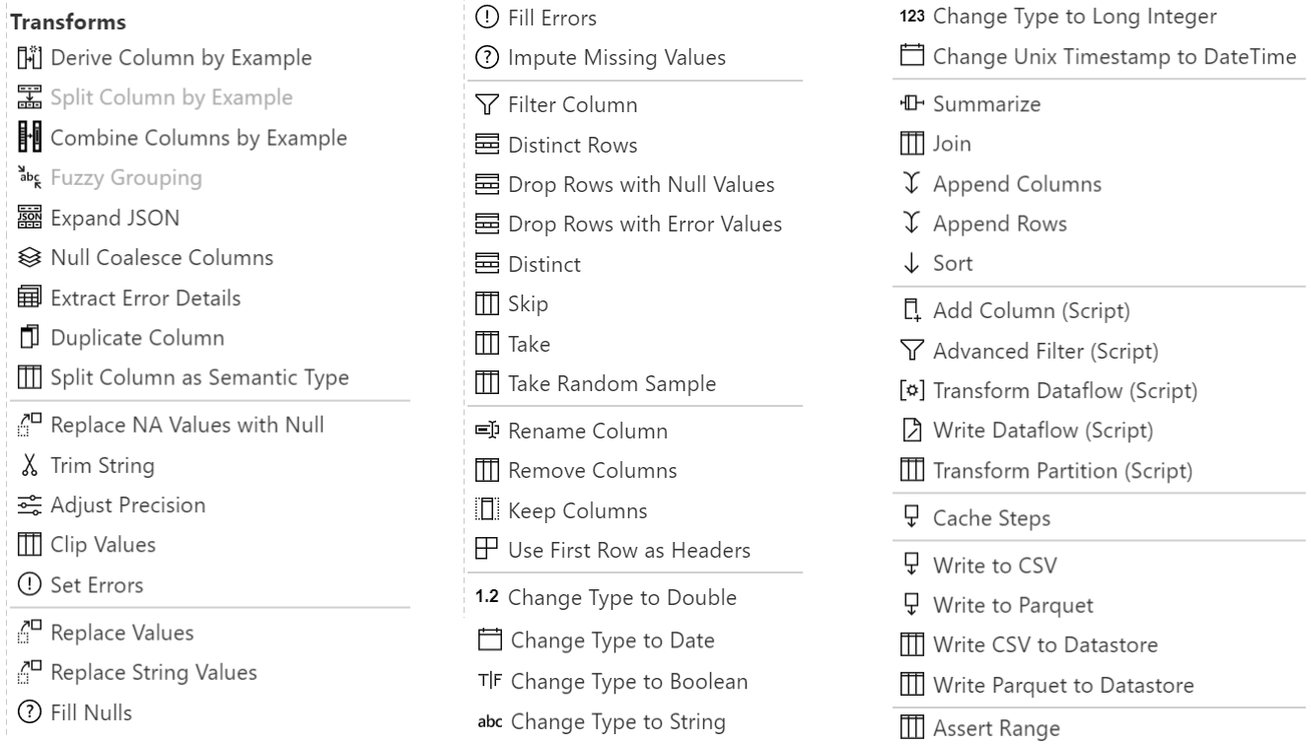


Data Prep: data wrangling issues

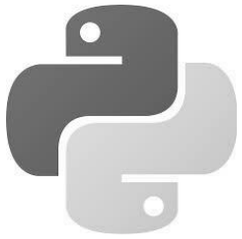
- Semantics: single-line data profiler and semantic type detection
- Merging data: join capabilities and compressed file reader
- Detecting, troubleshooting, and fixing errors: asserts
- Custom code: intelligent transforms
- Operationalization: single artifact usable in pipelines with ability to swap out data

```
profile = dflow.get_profile()
profile
```

	Type	Min	Max	Count	Missing Count	Not Missing Count	Percent missing	Error Count	Empty count	0.1% Quantile
Name	FieldType.STRING	Mr. Alexander J. Deborde	Ms. Yuhong Li	66.0	0.0	66.0	0.0	0.0	0.0	
CompanyName	FieldType.STRING	A Bike Store	Wholesale Parts	66.0	0.0	66.0	0.0	0.0	0.0	
SalesPerson	FieldType.STRING	adventure-works\david8	adventure-works\shu0	66.0	0.0	66.0	0.0	0.0	0.0	
EmailAddress	FieldType.STRING	aidan0@adventure-works.com	yuhong1@adventure-works.com	66.0	0.0	66.0	0.0	0.0	0.0	
Founded	FieldType.STRING	10-May-35	9-Jul-01	66.0	0.0	66.0	0.0	0.0	0.0	
Last Order	FieldType.DECIMAL	138	99489	66.0	0.0	66.0	0.0	0.0	0.0	138
Sales to Date	FieldType.DECIMAL	2.31478e+06	3.42244e+08	66.0	0.0	66.0	0.0	0.0	0.0	2.31478e+06
City	FieldType.STRING	S.A.	San Jose	66.0	0.0	66.0	0.0	0.0	0.0	
postal_code	FieldType.DECIMAL	94103	94133	66.0	0.0	66.0	0.0	0.0	0.0	94103

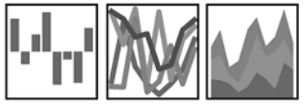


Data Prep: achieving scale



pandas

$y_i t = \beta' x_{it} + \mu_i + \epsilon_{it}$



PySpark

```
# Add 2 lines of code:
df = df.replace_datasource(big_dsource)
spark_df = df.to_spark_dataframe()
```

Scala

Data Prep SDK

Familiar pattern for complex transforms

Intelligent transforms (i.e. by example, auto-split, auto-join, fuzzy grouping)

Share pipelines via serialization

Smart file reading

Core Engine

Scale through streaming

Multiple runtimes (scale-up/scale-out) with a single artifact

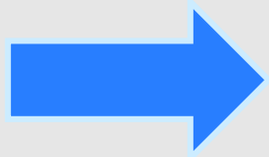
Responsive, lazy evaluations

Exercise – Data Preparation

@frlazzeri

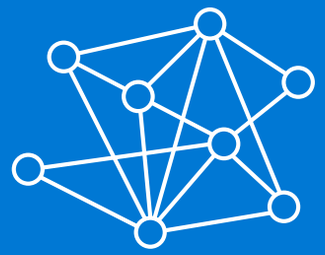
1. Go to: https://notebooks.azure.com/frlazzeri/projects/AIConferenceNYC

1. Make sure you see '**Python 3.6**' kernel on the top right for your notebook environment. If not, you can select '**Kernel**', then '**Change kernel**' to make changes.



Module_2_Get Started with AzureML

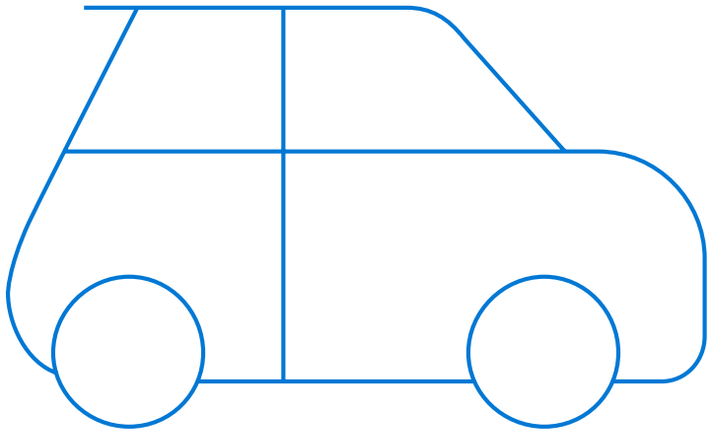
5min-dataprep-intro.ipynb



Automated machine learning

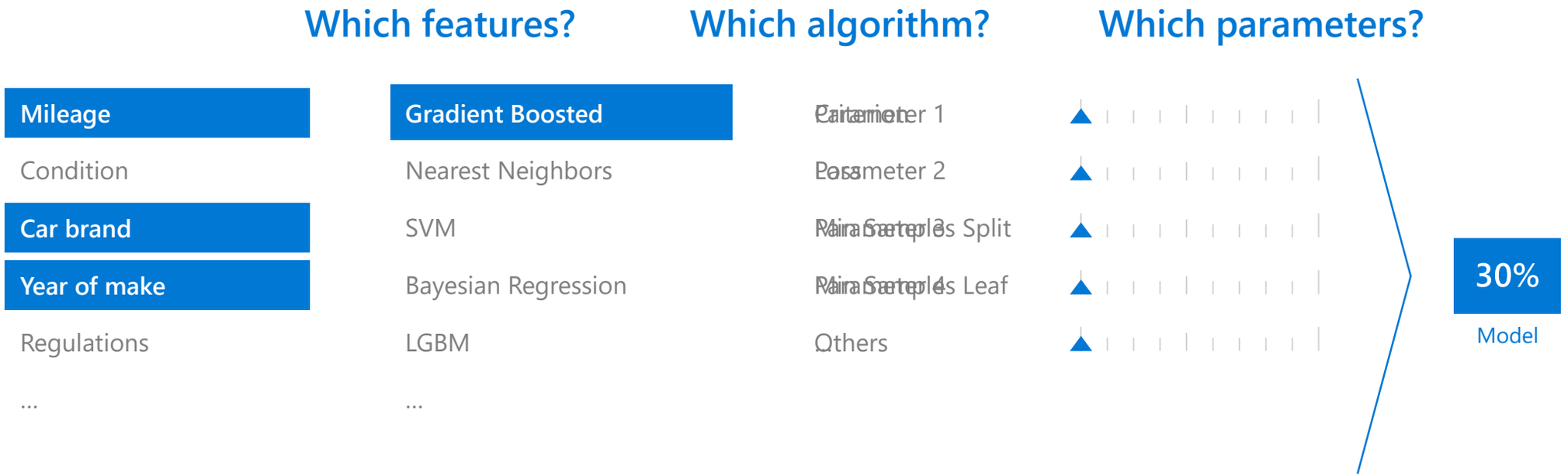
Azure Machine Learning

Automated machine learning

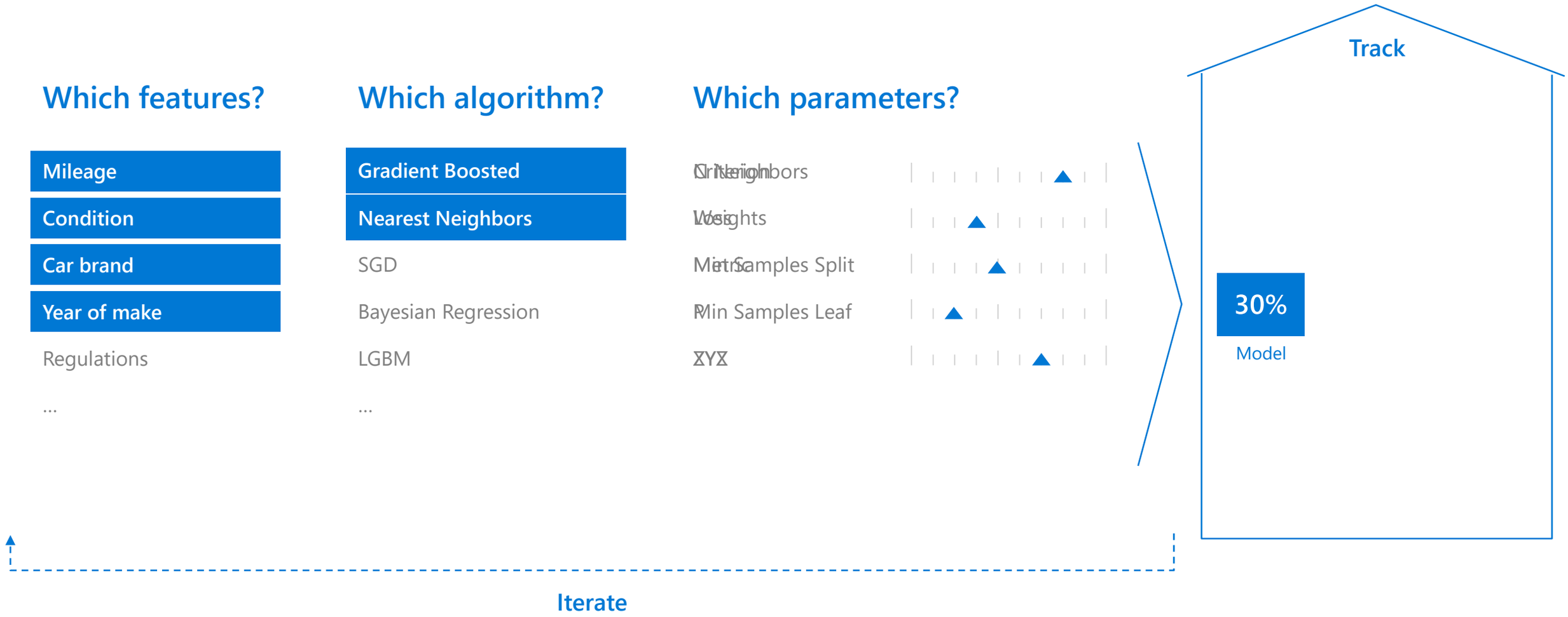


How much is this car worth?

Model creation is typically a time consuming process



Model creation is typically a time consuming process

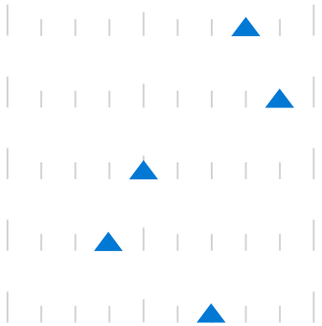


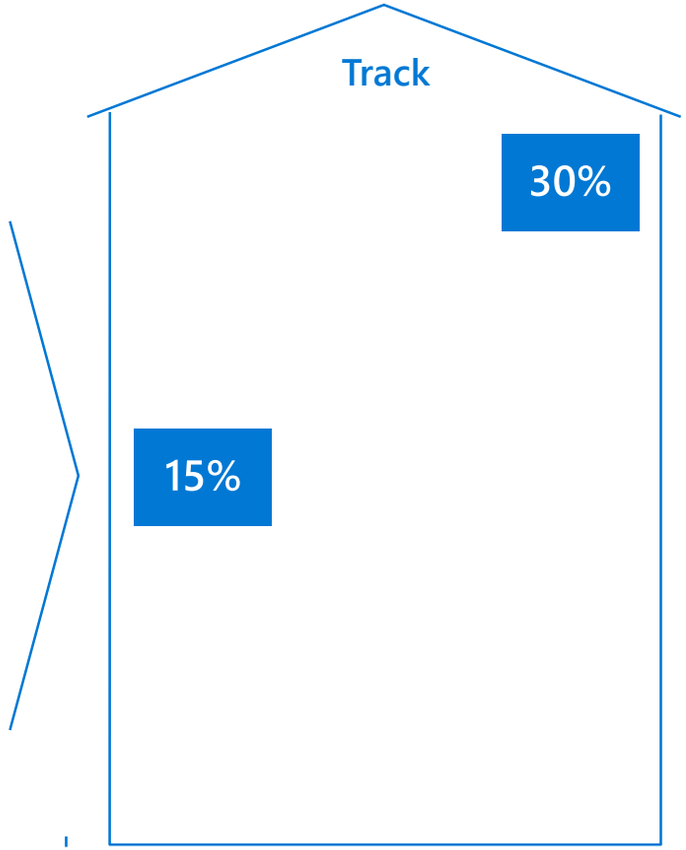
Model creation is typically a time consuming process

Which features?

Which algorithm?

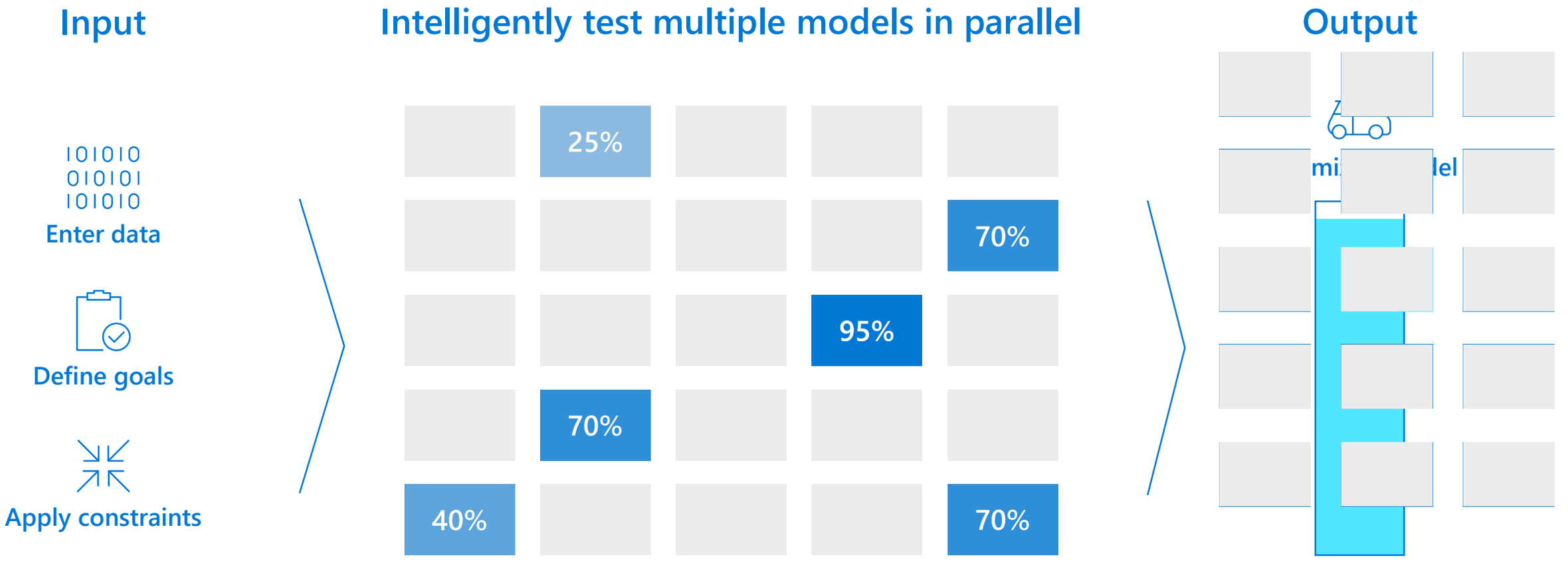
Which parameters?



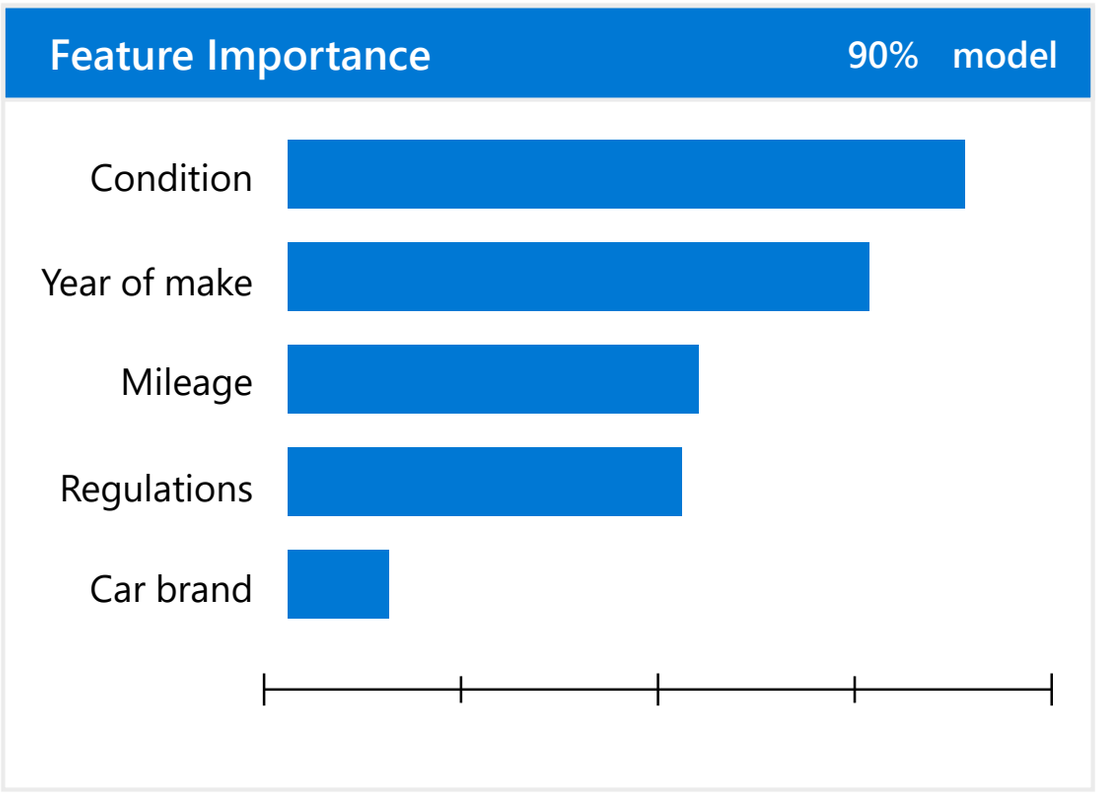


Iterate

Automated ML accelerates model development



Understand the inner workings of ML by analyzing feature importance



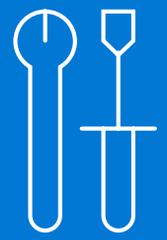
Enable model explainability for every automated ML iteration, not just the optimal model

ONNX Support

Problem	Data Scientists build ML models in Python. But, Most of LOB and Enterprise Apps in C# & Java
What are we doing about it?	AutoML will generate ONNX model by converting the Python model into ONNX format
Value Prop	Eliminates the recoding from Python to C# Faster inferencing due to the ONNX accelerators supported by hardware vendors
Features – (Apr 2019)	All scikit-learn models and transformations for classification & regression
Features – CY 2019	Time Series Forecasting

SQL Server Integration

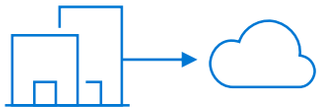
Problem	Lot of Data in SQL and not easy to move Database developers not familiar with ML
What are we doing about it?	Train and Predict using Stored Procedures in SQL
Value Prop	In DB Training and Prediction Train in AutoML and Deploy the model it in SQL as a stored procedure



Tool-agnostic Python SDK

Tool-agnostic Python SDK

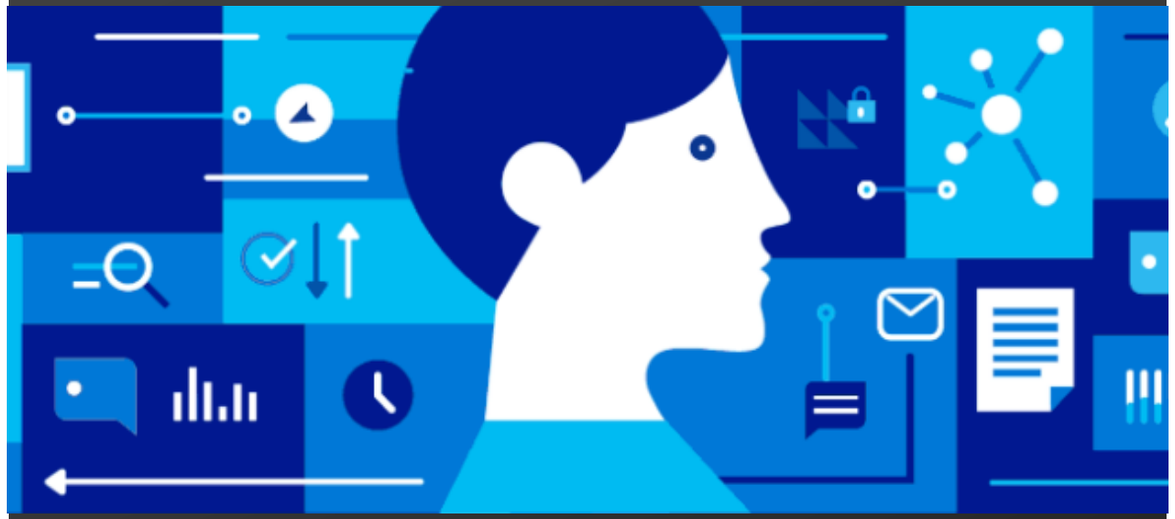
Use your favorite IDEs, editors, notebooks, and frameworks



Flexibility of your local environment or curated cloud environment

Integrate with other services like Azure Databricks

Get started quickly without any complex pre-requisites



Mission

Enable automated building of machine learning with the goal of accelerating, democratizing and scaling AI

Democratize AI

Enable Domain Experts & Developers to get rapidly build AI solutions

Accelerate AI

Improve productivity for Data scientists

Scale AI

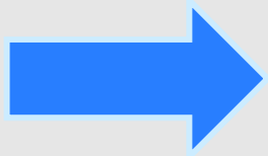
Build AI solutions at scale in an automated fashion

Exercise – Automated Machine Learning

@frlazzeri

1. Go to: https://notebooks.azure.com/frlazzeri/projects/AIConferenceNYC

2. Make sure you see '**Python 3.6**' kernel on the top right for your notebook environment. If not, you can select '**Kernel**', then '**Change kernel**' to make changes.



Module_2_Get Started with AzureML EnergyForecasting Notebook

Microsoft

@frlazzeri

Module 3

Introduction to recurrent neural networks (RNN) for time series forecasting

- Introduction to Feedforward Neural Networks
- Introduction to Recurrent Neural Networks

Why RNN?

- RNN has been shown perform well in many scenarios
  - 2014 Global Energy Forecasting Competition (link)
  - 2016 CIF International Time Series Competition (link)
  - 2017 Web Traffic Time Series Forecasting (link)
  - 2018 Corporación Favorita Grocery Sales Forecasting (link)
- RNN model is very flexible
- RNN can learn from big data

Microsoft

@frlazzeri

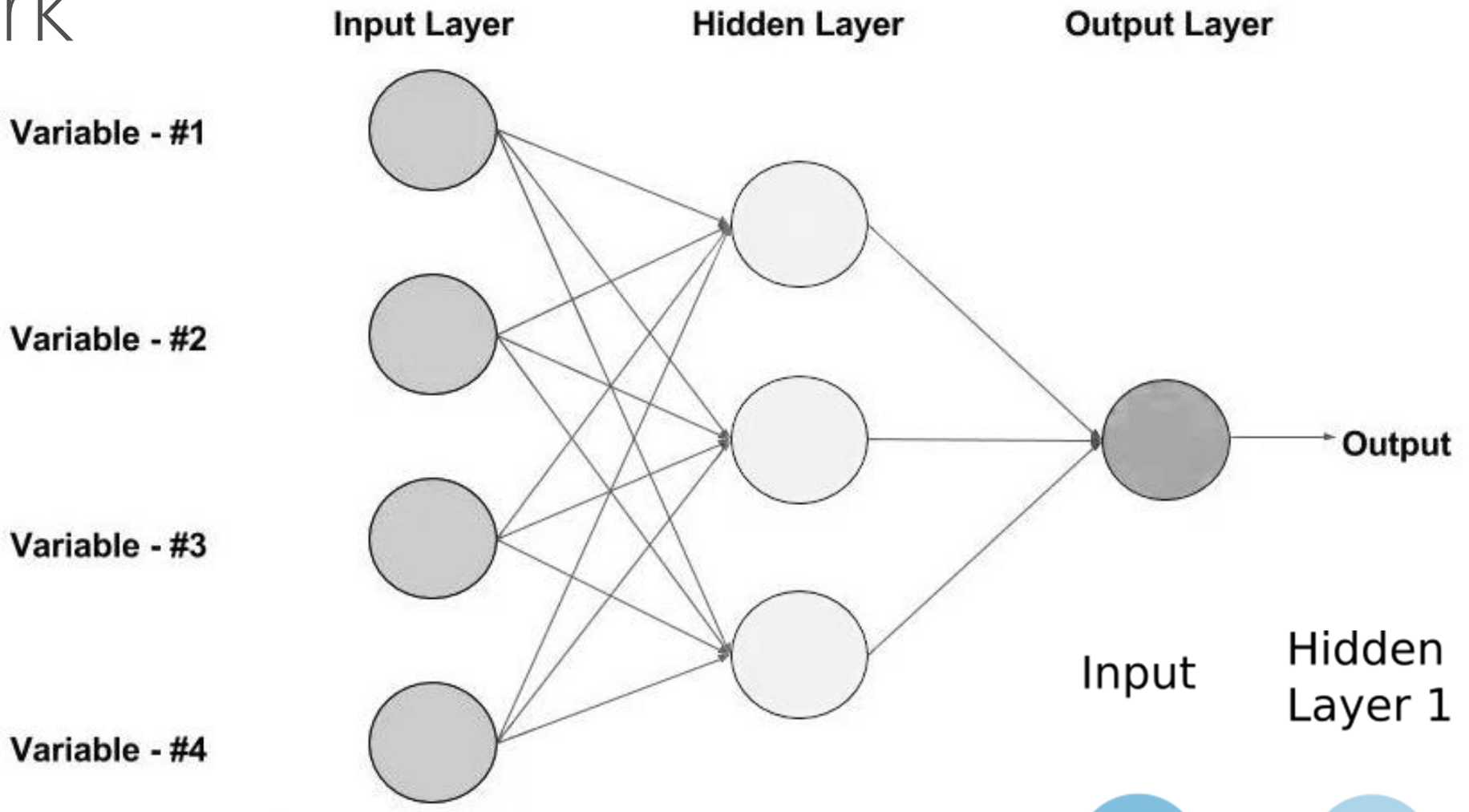
Introduction to Feedforward Neural Networks

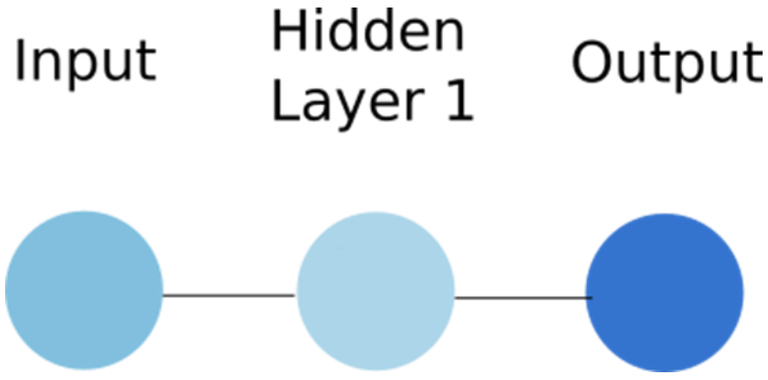
- Perceptron & Multilayer Perceptron
- Activation Functions
- Neural Network Training
  - Loss function
  - Gradient descent
  - Backpropagation

Resource: https://www.learnopencv.com/understanding-feedforward-neural-networks/

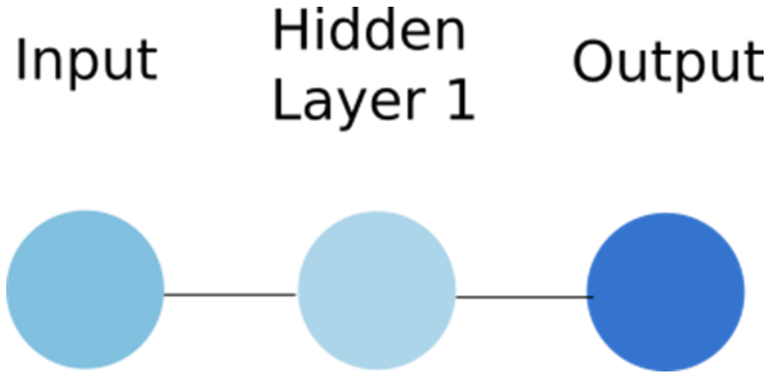
Understanding Feedforward Neural Network

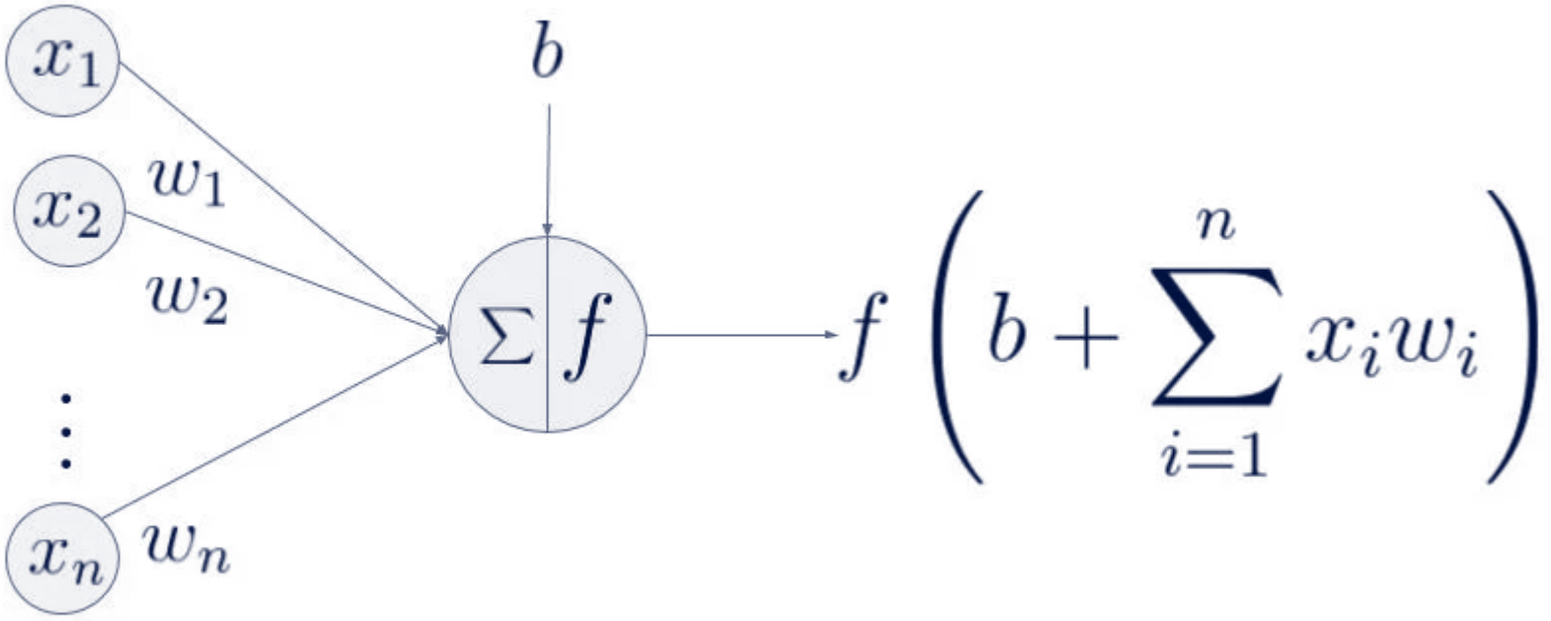
@frlazzeri





What is a Neuron?

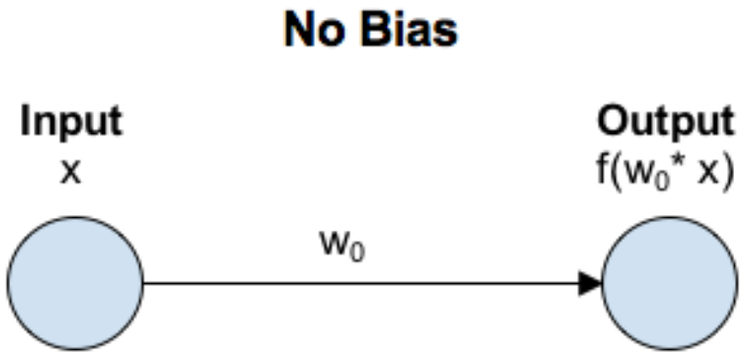


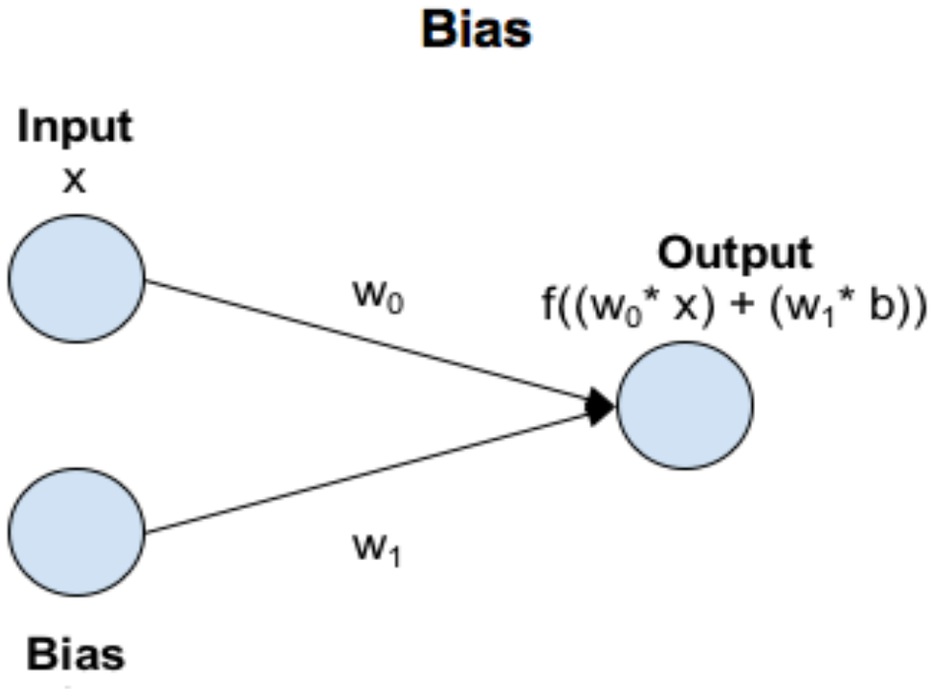


Weights and Bias

- Weights shows the strength of the particular node

- A bias value allows you to shift the activation function curve up or down

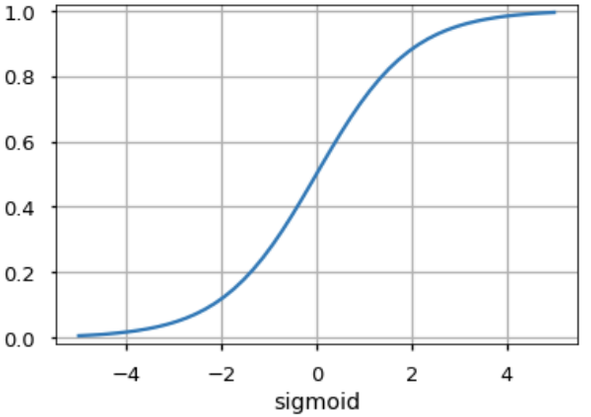




Activation Functions

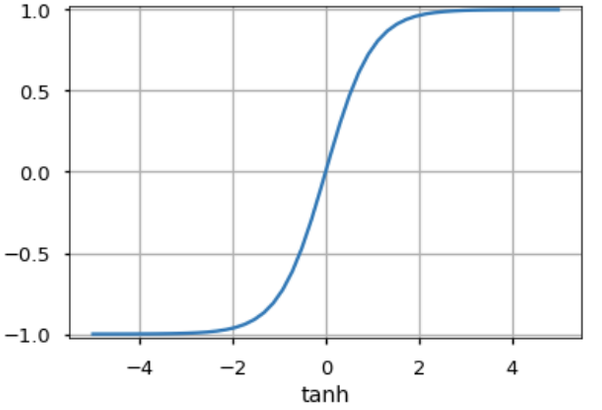
Sigmoid $f(x') = \sigma(x') = \frac{1}{1+e^{-x'}}$

It maps the input (y-axis) to values between 0 and 1



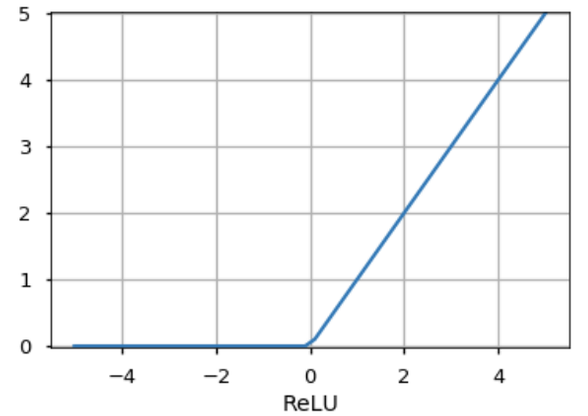
Tanh (hyperbolic tangent) $f(x') = \frac{e^{x'} - e^{-x'}}{e^{x'} + e^{-x'}}$

It is similar to the sigmoid function but maps the input to values between -1 and 1.



Rectifier linear unit (ReLU) $f(x') = \begin{cases} 0 \text{ for } x' < 0 \\ x' \text{ for } x' \geq 0 \end{cases}$

It allows only positive values to pass through it. The negative values are mapped to zero.



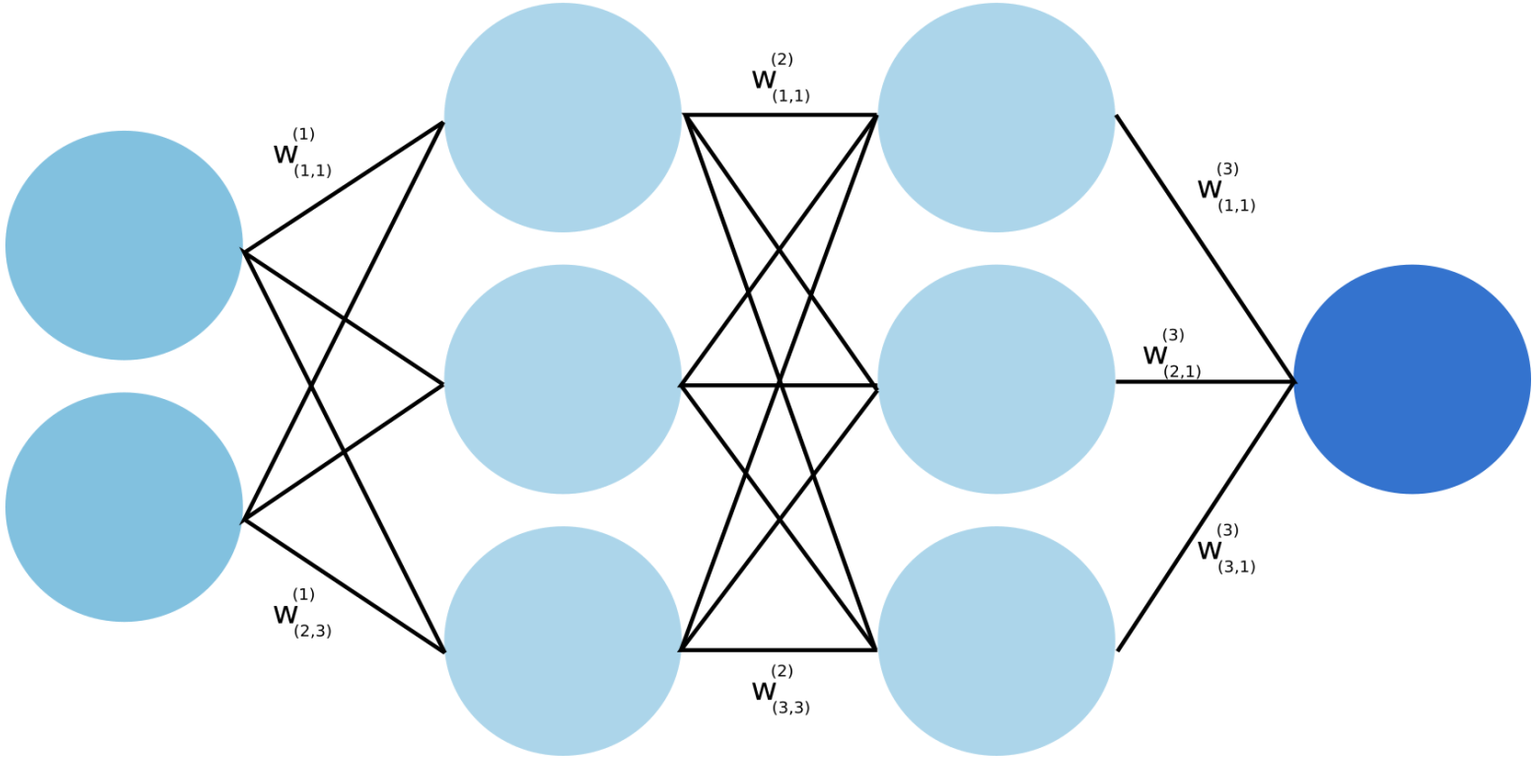
Multilayer Perceptron

@frlazzeri

also known as Feed-Forward Neural Network and Deep Neural Network

Input | Hidden Layer 1 | Hidden Layer 2 | Output

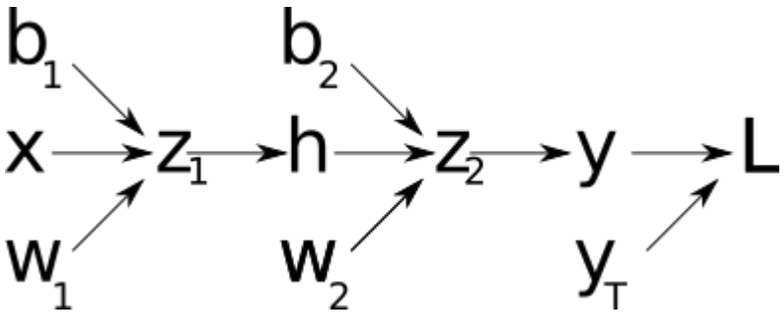
$$L = \frac{1}{2}(y - y_T)^2$$

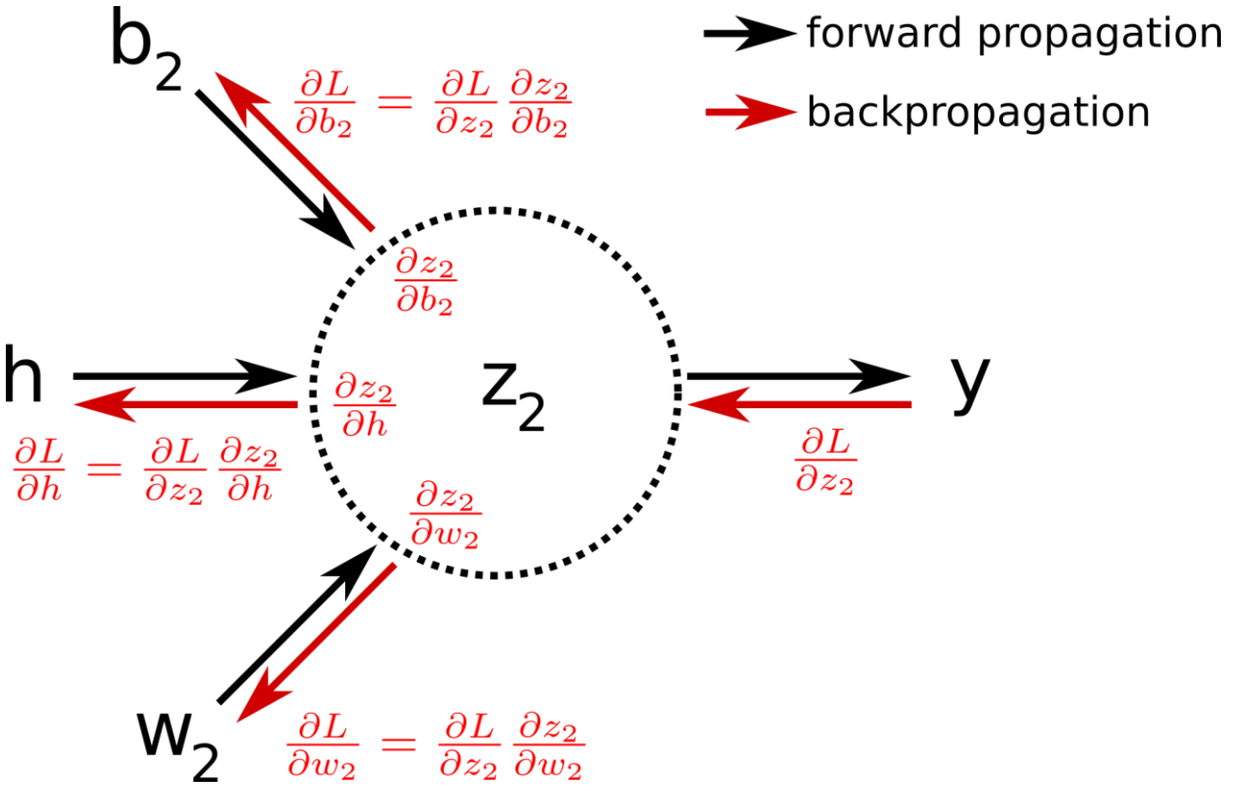


How does the network learn?

- Use training examples to find *good* weights and biases of the network.
- The training samples are passed through the network and the output obtained from the network is compared with the actual output.
- This *error* is used to change the weights of the neurons such that the error decreases gradually.
- This is done using the *Backpropagation* algorithm.

Backpropagation algorithm





Loss/Cost function

Commonly used loss functions in time series forecasting:

- Mean-squared-error (MSE): $\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$
- Mean Absolute Percentage Error (MAPE): $\frac{100}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$
- Symmetric MAPE (sMAPE): $\frac{100}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y}_i|}{(|y_i| + |\hat{y}_i|)/2}$

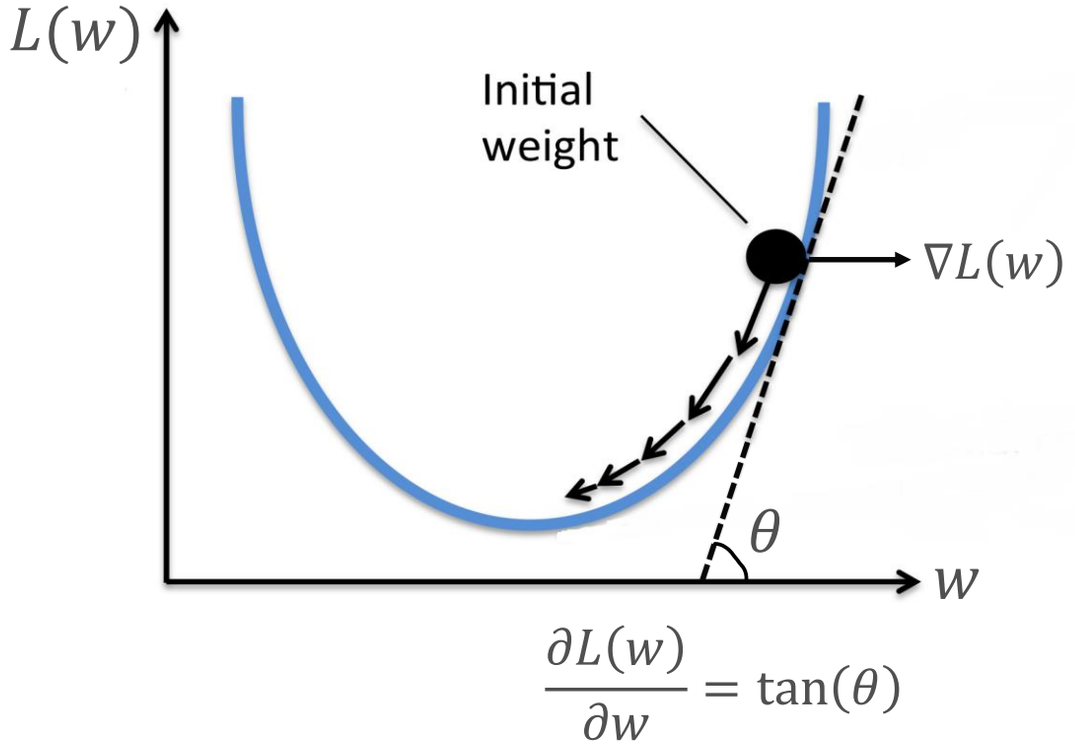
$\hat{y}_i$ is a function of weights w and biases b.

Loss function decomposition $L(w, b) = \frac{1}{N}\sum_{i=1}^{N} L_i(w, b)$

Gradient descent

Gradient $\nabla L(w) = \left(\frac{\partial L(w)}{\partial w_1}, \frac{\partial L(w)}{\partial w_2}, \ldots, \frac{\partial L(w)}{\partial w_d}\right)$ - direction of the maximal increase of $L(w)$

1D example: $\nabla L(w) = \left(\frac{\partial L(w)}{\partial w}\right)$



Optimization algorithm

Initialization: $w = w_0$

While stopping criterion not met:

$$w = w - \alpha \cdot \nabla L(w)$$

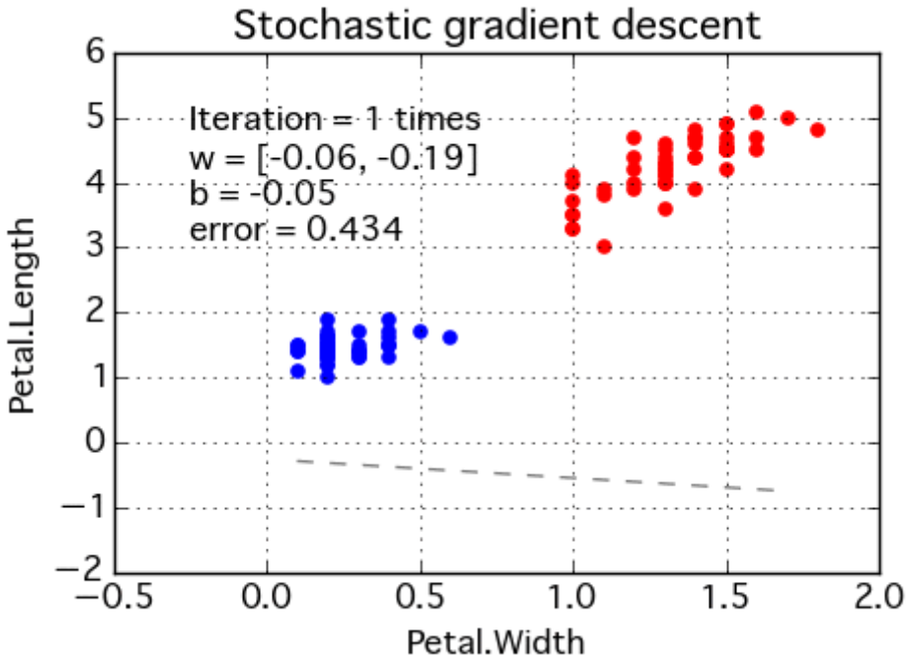
learning rate

Types of Gradient Descents

- **Batch Gradient Descent**: This method calculates the margin of error, but only updates the model once the entire dataset has been evaluated. It is computationally efficient, but requires memory, and does not always achieve the most accurate results.

- **Stochastic Gradient Descent**: Instead of updating the model once the dataset has been evaluated, it does so after every training example. The frequent updates allow you to see the detailed improvement until convergence—error rate can vary.

- **Mini-Batch Gradient Descent**: This is the method that is used most often in deep learning and training neural networks, and it is a combination of the BGD and the SGD. The dataset is divided into small batches of about 50 to 256, then evaluates each of the batches separately.



Batch gradient descent in machine learning

Loss function decomposition $L(w) = \frac{1}{N}\sum_{i=1}^{N} L_i(w)$

$$\nabla L(w) = \frac{1}{N}\sum_{i=1}^{N} \nabla L_i(w)$$

Initialization: $w = w_0$

While stopping criterion not met:

$$w = w - \frac{\alpha}{N}\sum_{i=1}^{N} \nabla L_i(w)$$

Need to go over all examples to complete a single optimization step.

With large N, gradient descent has a very slow convergence.

Stochastic Gradient Descent (SGD)

$$L(w) = \frac{1}{N}\sum_{i=1}^{N} L_i(w) \qquad \nabla L(w) = \frac{1}{N}\sum_{i=1}^{N} \nabla L_i(w)$$

Initialization: $w = w_0$

While stopping criterion not met:

Shuffle examples randomly

For $i = 1 \dots N$

$w = w - \alpha \nabla L_i(w)$

} epoch

[Robbins and Monro, 1951]

Batch gradient descent

Initialization: $w = w_0$

While stopping criterion not met:

$$w = w - \frac{\alpha}{N}\sum_{i=1}^{N}\nabla L_i(w)$$

- Each update is slow
- Each update is very accurate (w moves towards optimal value)
- Small number of updates to converge

SGD

Initialization: $w = w_0$

While stopping criterion not met:

Shuffle examples randomly

For $i = 1 \ldots N$

$$w = w - \alpha\nabla L_i(w)$$

- Each update is very fast
- Each update is very noisy (w doesn't necessary move towards optimal value)
- Large number of updates to converge

Minibatch Stochastic Gradient Descent

Initialization: $w = w_0$

While stopping criterion not met:

- Shuffle examples randomly
- Partition examples into batches of size m
- For $\mathbf{batch=1...N/m}$ examples
  - $w = w - \frac{\alpha}{m}\sum_{i=1}^{m}\nabla L_i(w)$

} epoch

m – mini-batch size (default value 32 in Keras)

Exercise

1. Go to: https://notebooks.azure.com/frlazzeri/projects/AIConferenceNYC

2. Once you are logged in to Azure Notebooks, on the top right, select '**Clone**' tab. Then type in any name you prefer for '**Project Name**' and '**Project ID**'. Once you have filled in all boxes, click '**Clone**'. Wait till you see a list of files cloned.

3. On the top left, select '**Free Compute**' as compute target option from the dropdown menu.

4. Make sure you see '**Python 3.6**' kernel on the top right for your notebook environment. If not, you can select '**Kernel**', then '**Change kernel**' to make changes.

5. Run each cell in the notebook by click '**Run**' on top. This notebook will download sample data to your environment and visualize the data. Wait and make sure you can see all the visualizations. Now you are all set!

Module_3_Intro to NNs for TS Forecasting

Microsoft

Introduction to Recurrent Neural Networks

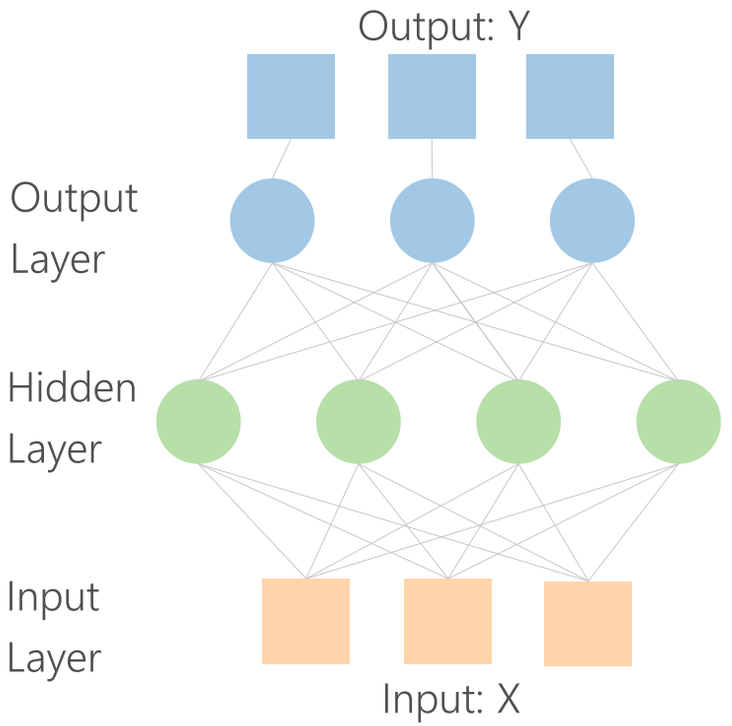
- What are RNNs?
- How RNNs are trained: Backpropagation through time (BPTT)
- Vanilla RNN and its gradient problems
- Other RNN units
  - LSTM
  - GRU

Reference: https://github.com/Azure/RNNForTimeSeriesForecasting

What are RNNs?

@frlazzeri

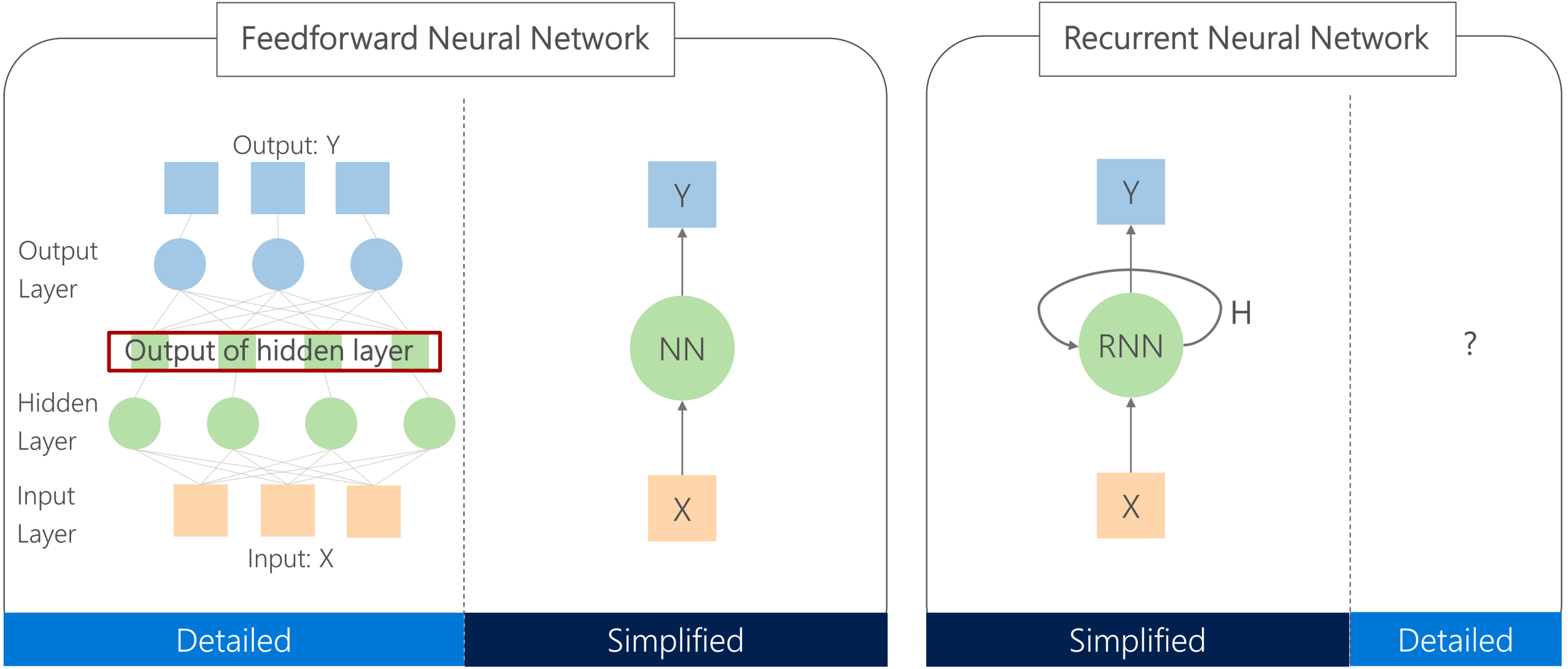
Feedforward Neural Network



Detailed

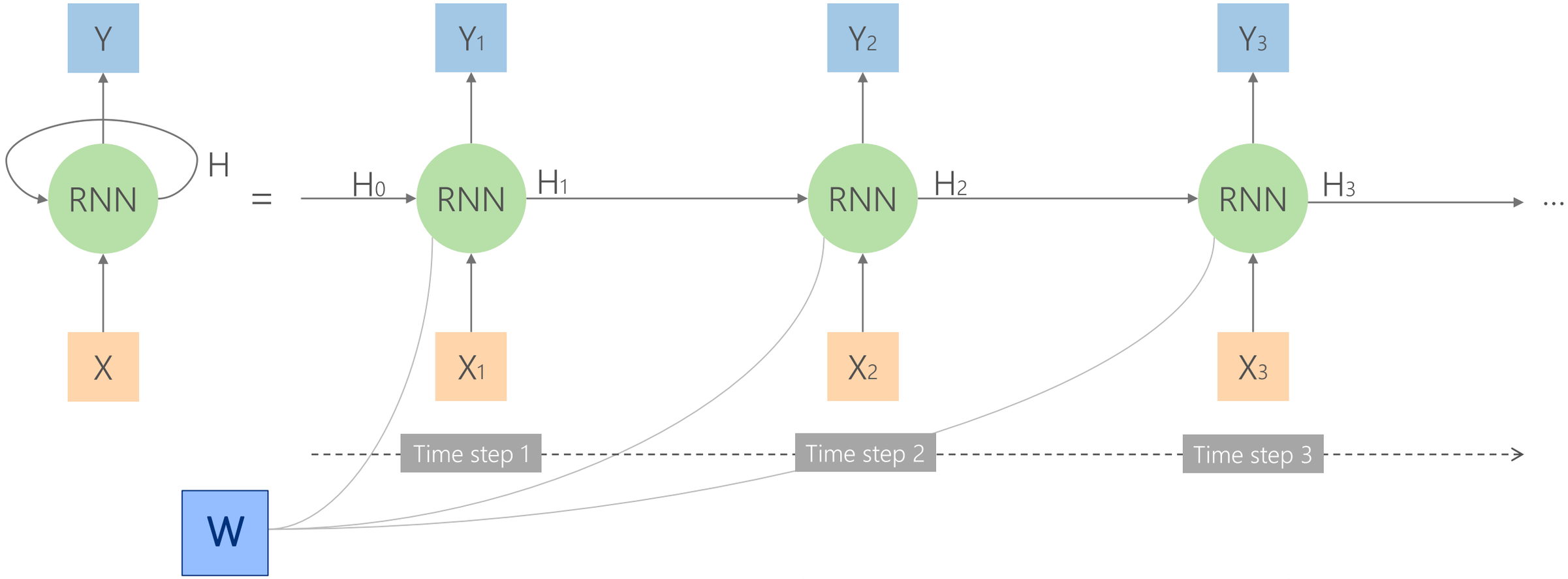
What are RNNs?

RNN has internal hidden state which can be fed back to network



Unrolled RNN

The same weight and bias shared across all the steps



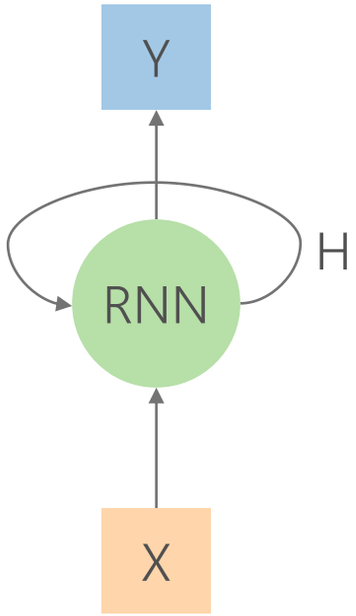
In Keras you will see "timesteps" when set data input shape

Unrolled RNN

@frlazzeri

*In Keras, the parameter **"units"** is dimensions of hidden state.*
(Think of it as feedforward neural network number of units in hidden layer.)

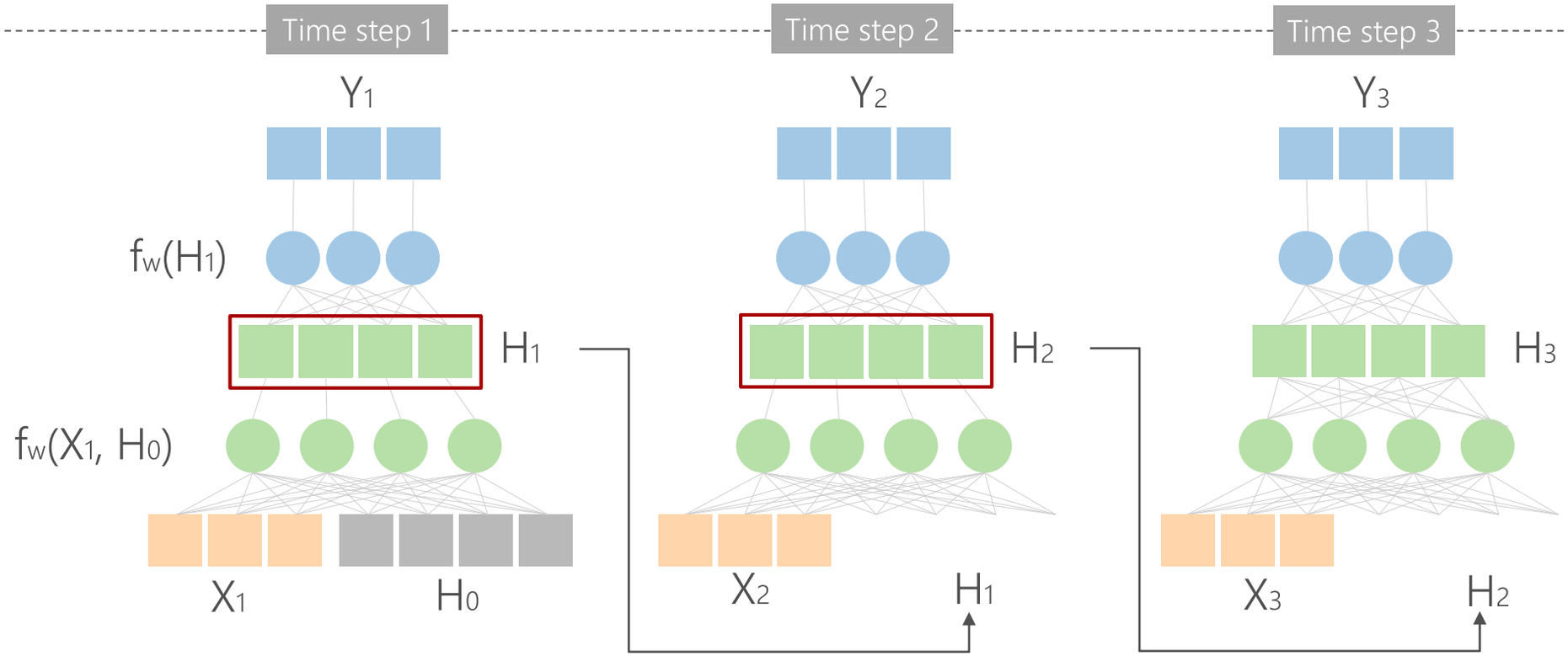
```
model = Sequential()
model.add(RNN(4, input_shape=(timesteps,
data_dim)))
model.add(Dense(3, activation='softmax'))
```



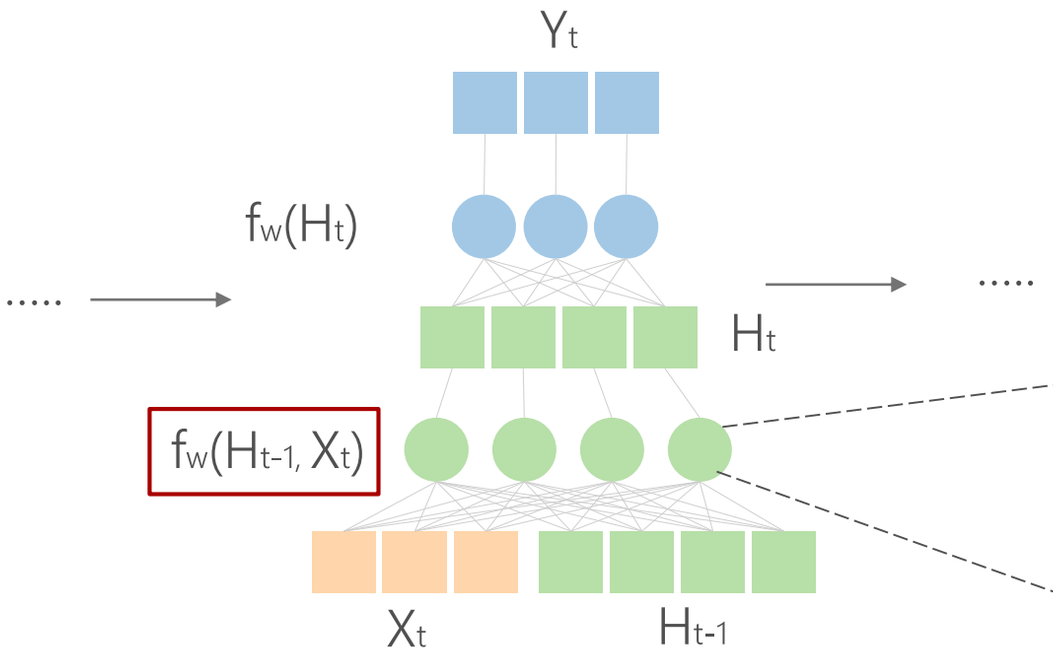
=

Simplified

Detailed

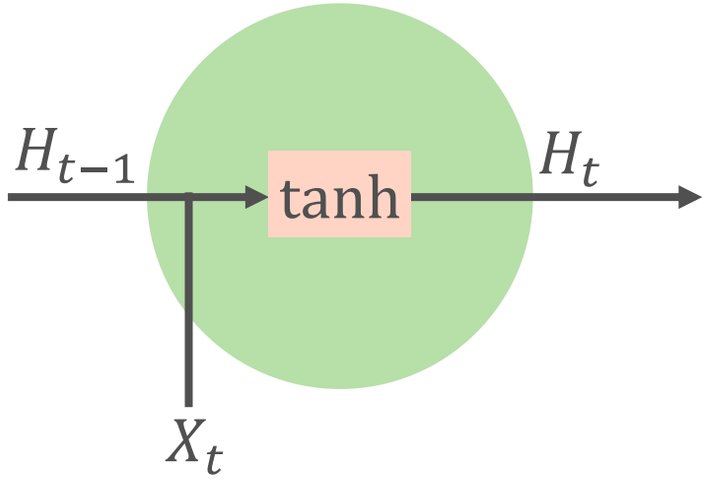


Vanilla RNN



Vanilla RNN:

$$H_t = \tanh(W_h H_{t-1} + W_x X_t)$$
$$= \tanh(\mathbf{W} \cdot [H_{t-1}, X_t])$$

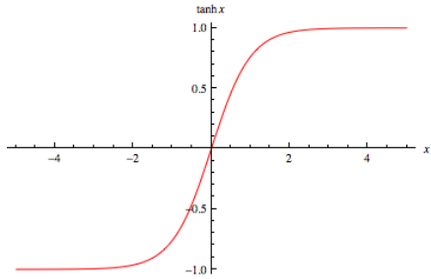


some function with parameter W

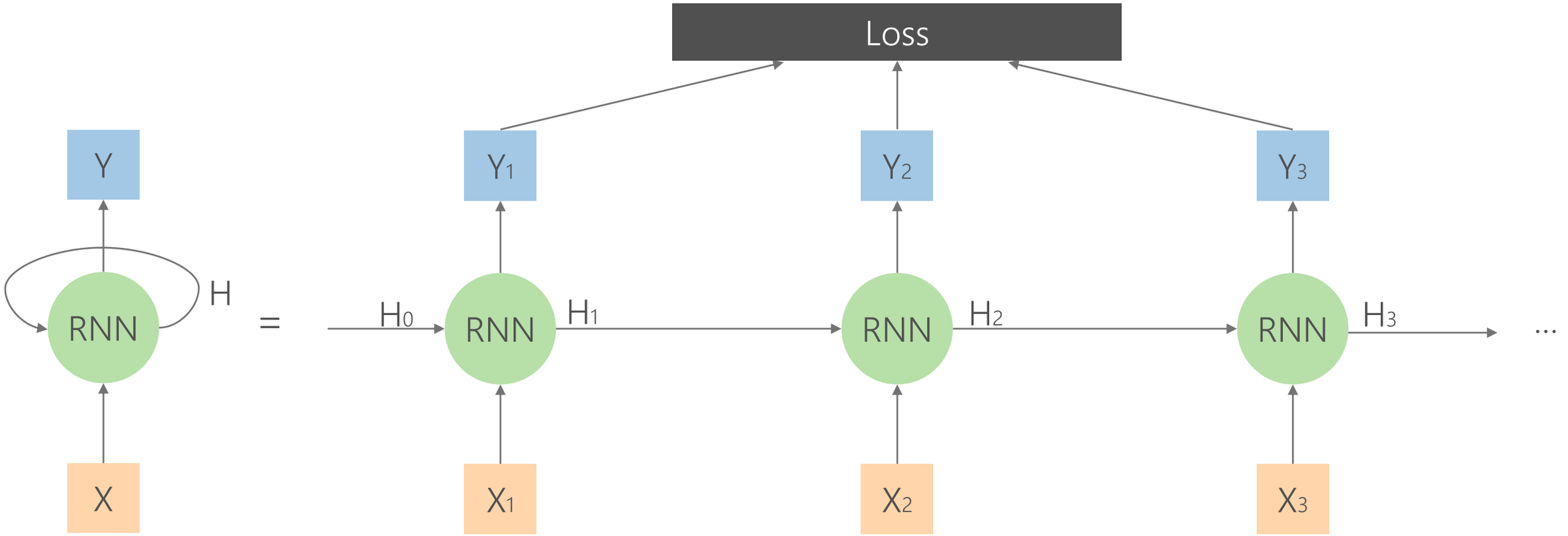
$$H_t = f_W(H_{t-1}, X_t)$$

new state — old state — Input vector at some time step

$$Y_t = \text{softmax}(W_y H_t)$$



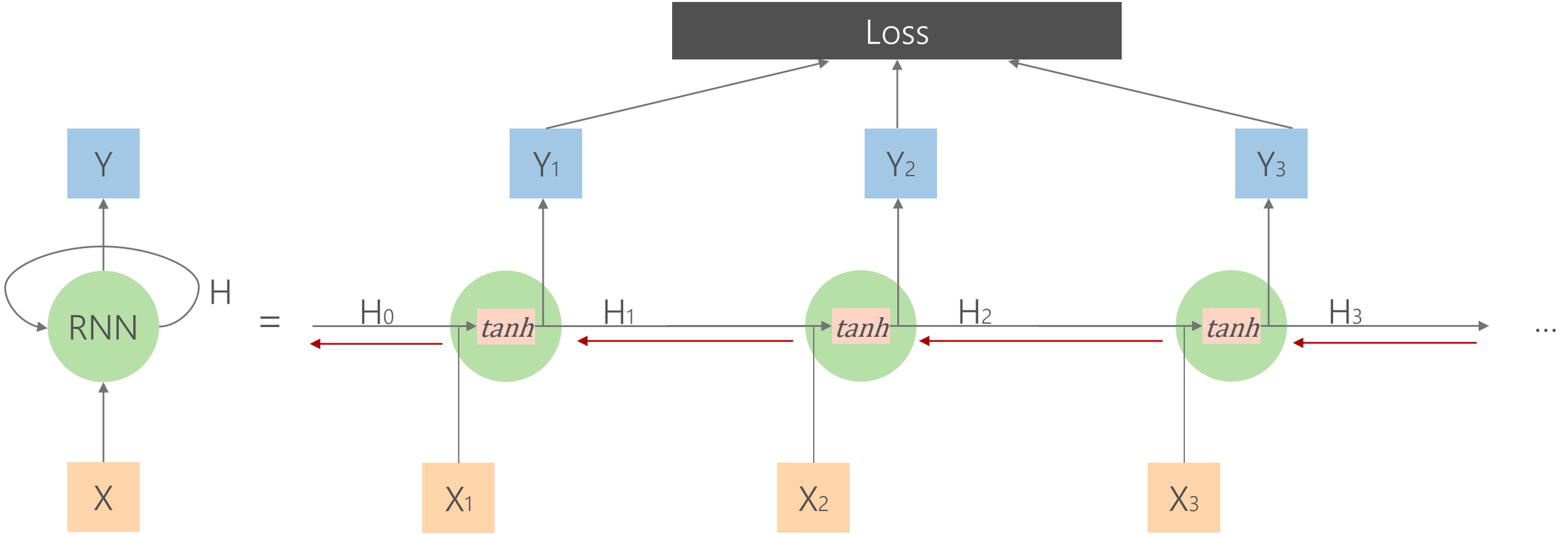
Backpropagation through time (BPTT)



Forward through entire sequence to compute loss

then backward through entire sequence to compute gradient

Vanilla RNN BPTT

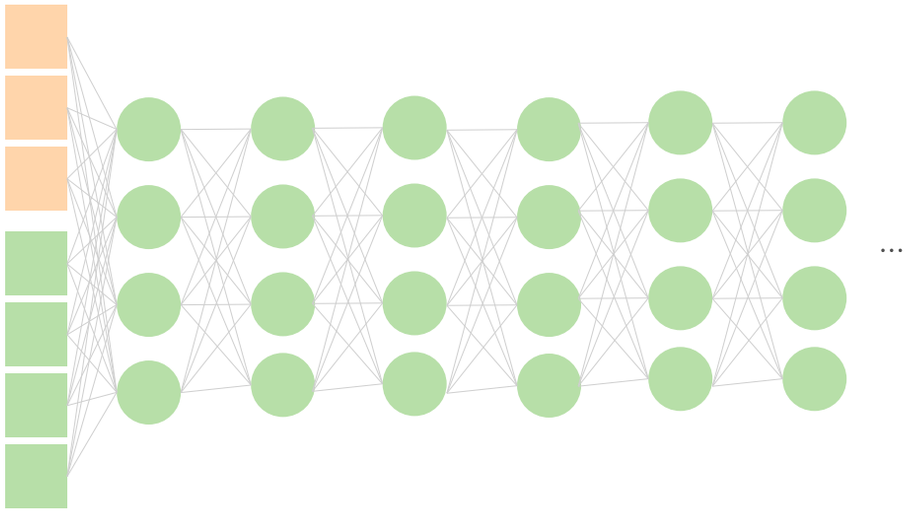


Computing gradient of H_0 involves repeated tanh and many factors of W

Vanilla RNN Gradient Problems

Computing gradient of h_0 involves repeated tanh and many factors of W which causes:

- **Exploding gradient** (e.g. 5*5*5*5*5*5*.....)
- **Vanishing gradients** (e.g. 0.7*0.7*0.7*0.7*0.7*0.7*.....)



100 time steps is similar to 100 layers feedforward neural net

Exploding Gradient

- What Is the Problem with Exploding Gradients?
  - In deep multilayer Perceptron networks, exploding gradients can result in an unstable network that at best cannot learn from the training data and at worst results in NaN weight values that can no longer be updated.

- How to Fix Exploding Gradients?
  - Re-Design the Network Model
  - Use Long Short-Term Memory Networks
  - Use Gradient Clipping
  - Use Weight Regularization

Vanishing Gradient

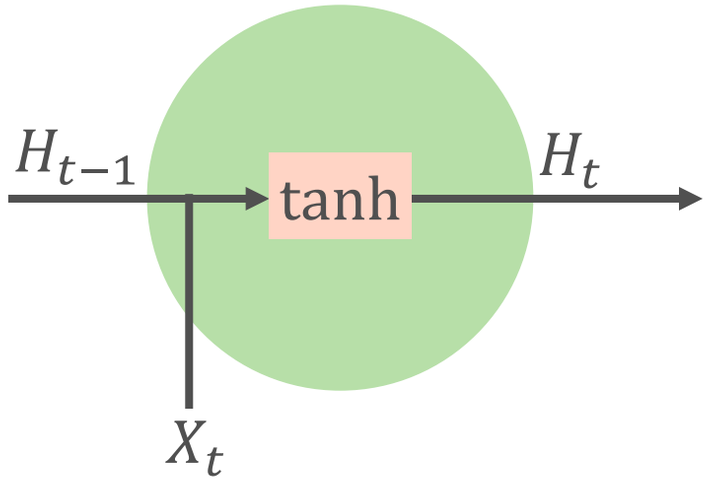
- Vanishing gradients are more problematic because it is not obvious when they occur or how to deal with them

- Solutions:
  - Proper initialization
  - Change activation function from tanh to ReLU
  - Regularization
  - Change architecture to LSTM or GRU

Long Short-Term Memory (LSTM)

Vanilla RNN:

$$H_t = \tanh(W_h H_{t-1} + W_x X_t)$$
$$= \tanh(\mathbf{W} \cdot [H_{t-1}, X_t])$$

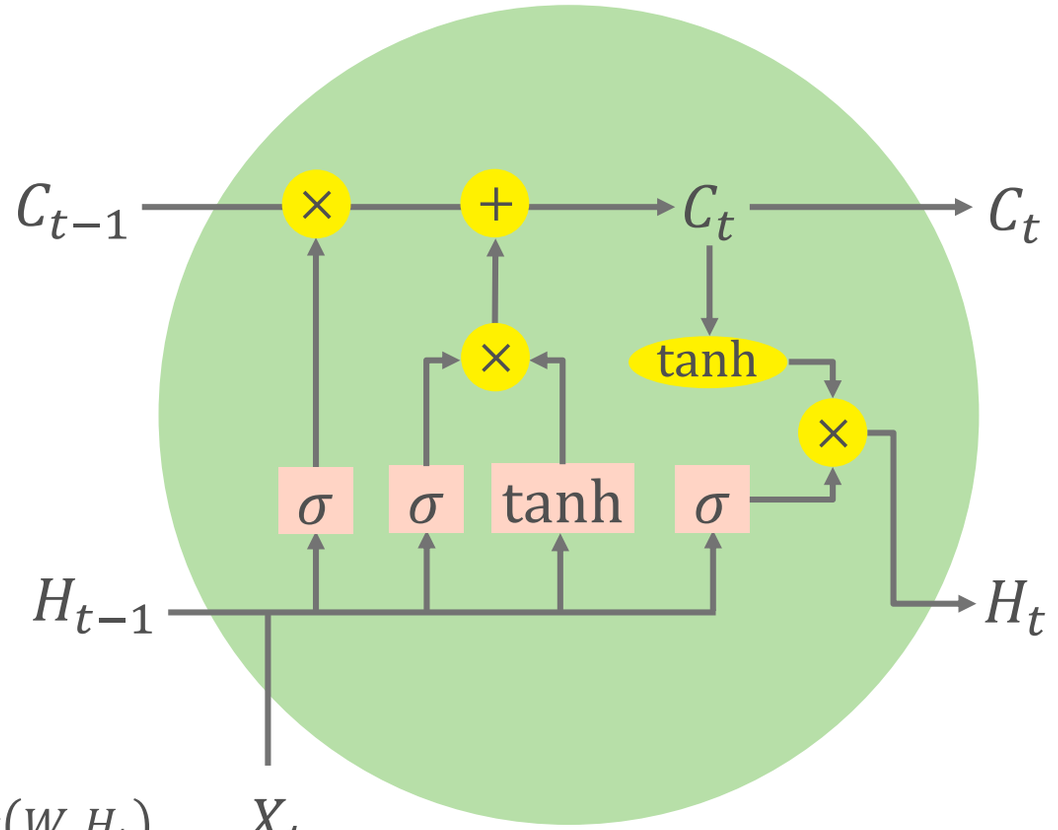


$$Y_t = \text{softmax}(W_y H_t)$$

LSTM:

1 more state $C_t = f \times C_{t-1} + i \times g$

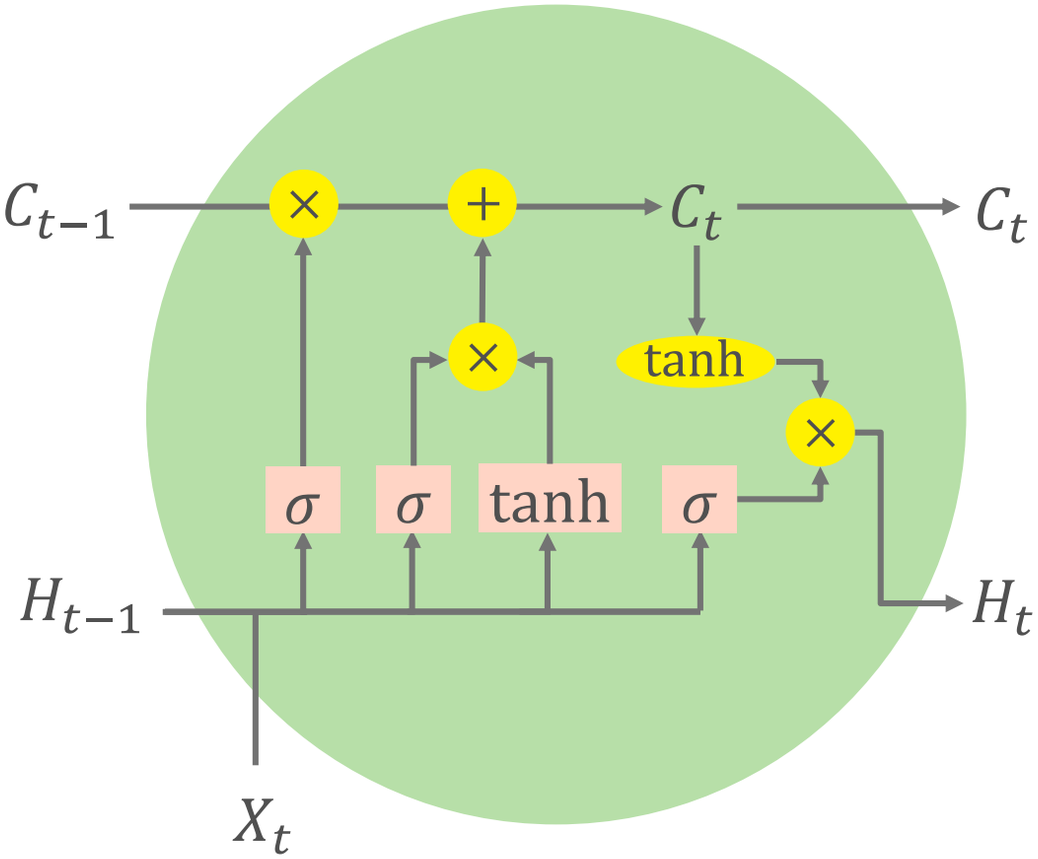
$$H_t = o \times \tanh(C_t)$$



$$Y_t = \text{softmax}(W_y H_t)$$

@frlazzeri

Long Short-Term Memory (LSTM)



Forget gates: $f = \sigma(W_f \cdot [H_{t-1}, X_t])$

Input gates: $i = \sigma(W_i \cdot [H_{t-1}, X_t])$

Gate gates: $g = \tanh(W_g \cdot [H_{t-1}, X_t])$

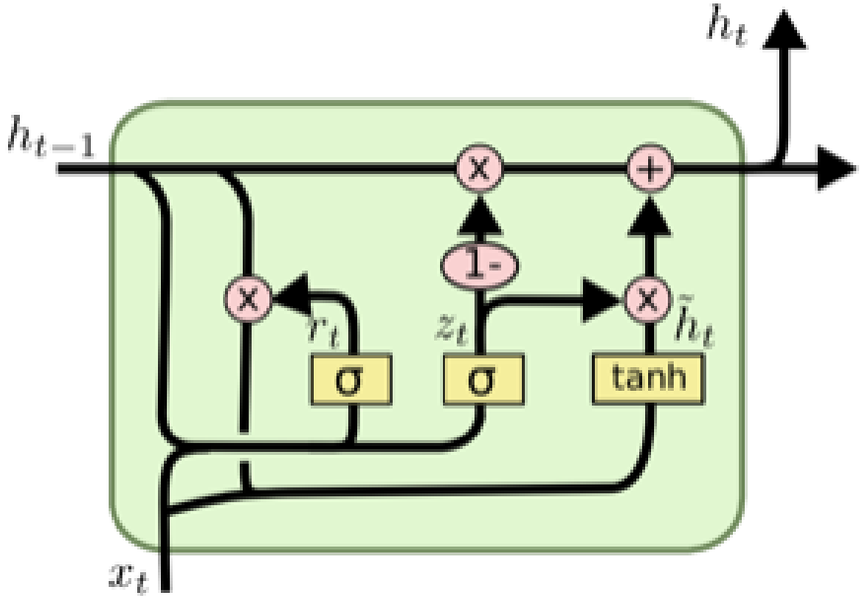
Output gates: $o = \sigma(W_o \cdot [H_{t-1}, X_t])$

Hidden cell state: $C_t = f \times C_{t-1} + i \times g$

Hidden state: $H_t = o \times \tanh(C_t)$

Gated Recurrent Unit (GRU)

Cheaper LSTM. Only 3 gates instead of 4 gates. Each gates has its own weight and bias, saving the computation of the 4th weight and bias.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

GRU [Learning phrase representations using rnn encoderdecoder for statistical machine translation, Cho et al. 2014]
LSTM: A Search Space Odyssey, Greff et al., 2015

Exercise

1. Go to: https://notebooks.azure.com/frlazzeri/projects/AIConferenceNYC

2. Once you are logged in to Azure Notebooks, on the top right, select '**Clone**' tab. Then type in any name you prefer for '**Project Name**' and '**Project ID**'. Once you have filled in all boxes, click '**Clone**'. Wait till you see a list of files cloned.

3. On the top left, select '**Free Compute**' as compute target option from the dropdown menu.

4. Make sure you see '**Python 3.6**' kernel on the top right for your notebook environment. If not, you can select '**Kernel**', then '**Change kernel**' to make changes.

5. Run each cell in the notebook by click '**Run**' on top. This notebook will download sample data to your environment and visualize the data. Wait and make sure you can see all the visualizations. Now you are all set!

Module_3_Intro to NNs for TS Forecasting

Microsoft

@frlazzeri

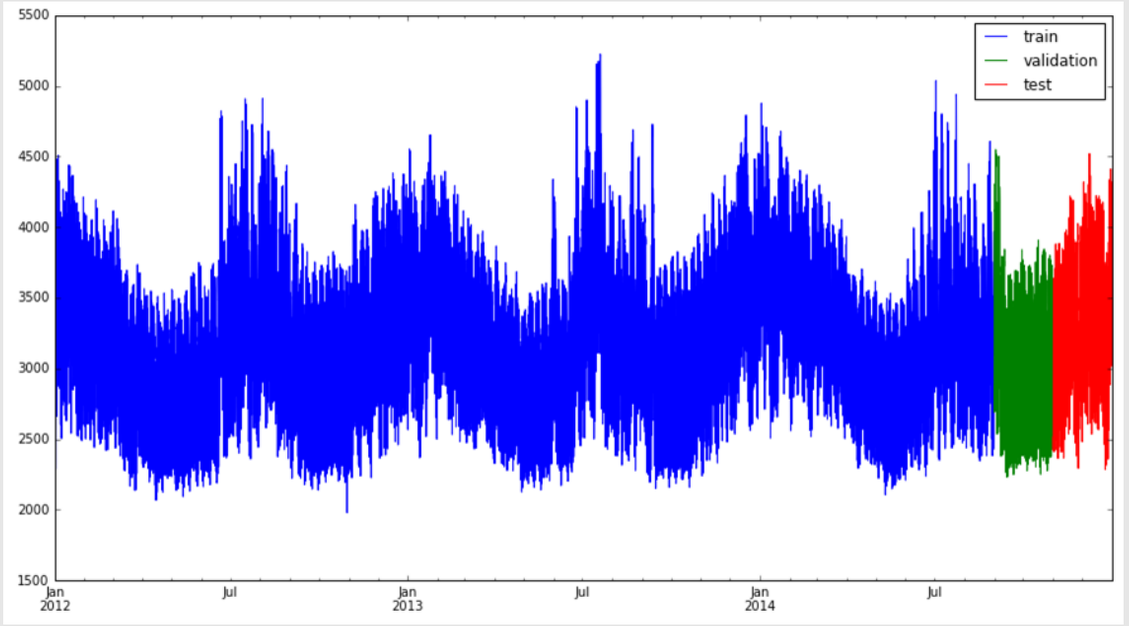
Module 4
Build your own time series forecasting model

- Time series forecasting models based on:
  - Energy demand forecasting use case
  - Stock market predictions with LSTMs

Use Cases

- Scenario: Univariate time series 1 step ahead forecast
  - Introduce the hardware and software that we will use to solve the problem
  - Data preparation for the problem
    - Data pre-processing (i.e. scaling, ….)
    - Construct data in the format that Keras accepts
  - Model building
  - Model tuning

Scenario:
Energy Load Forecasting



Short Term vs. Long Term Prediction

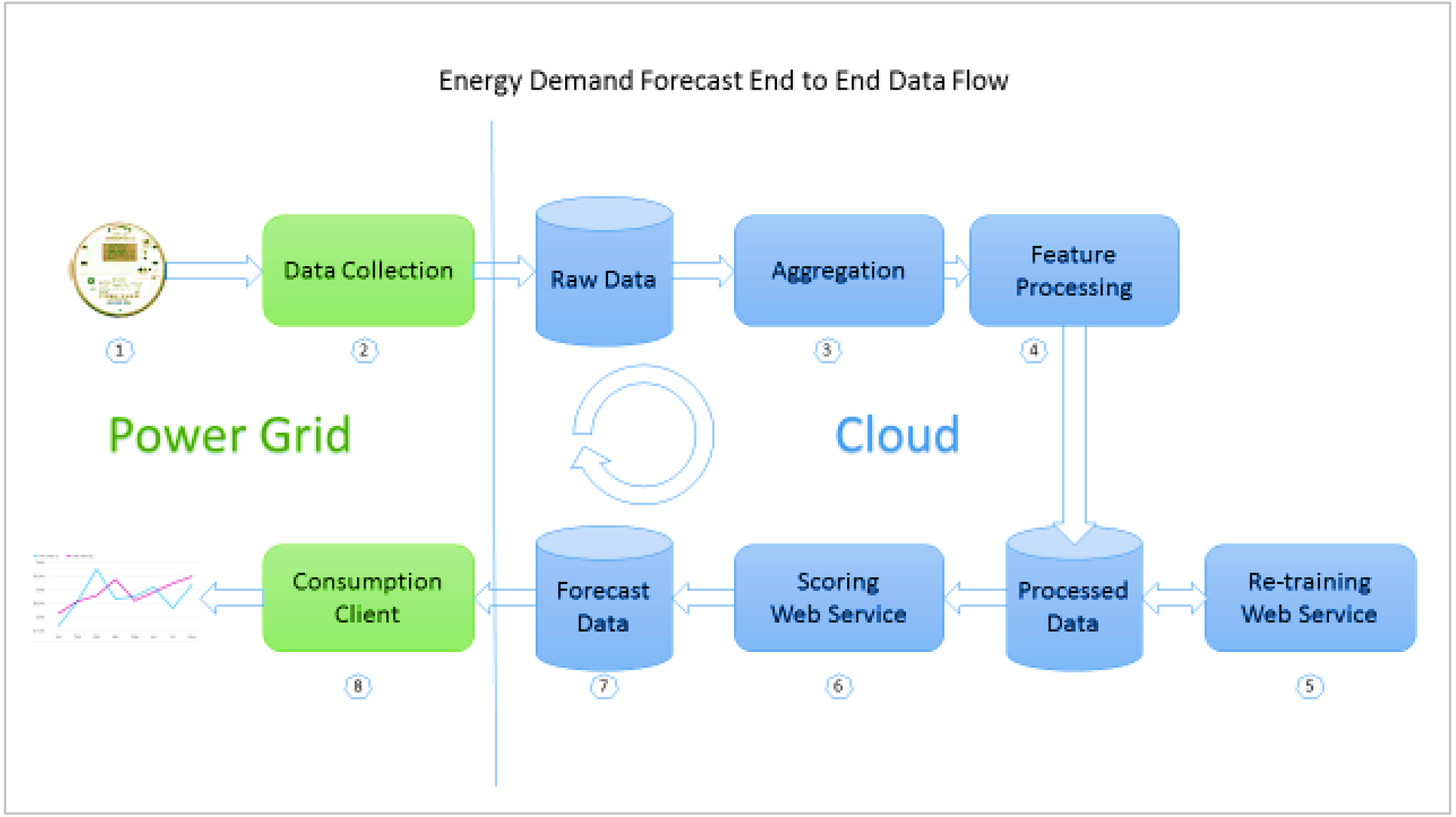
@frlazzeri

Attribute	Short Term Load Forecast	Long Term Load Forecast
Forecast Horizon	From 1 hour to 48 hours	From 1 to 6 months or more
Data granularity	Hourly	Hourly or daily
Typical use cases	• Demand/Supply balancing • Pick hour forecasting • Demand response	• Long term planning • Grid assets planning • Resource planning
Typical predictors	• Day or week • Hour of day • Hourly temperature	• Month of year • Day of month • Long term temperature and climate
Historical data range	Two to three years' worth of data	Five to 10 years' worth of data
Typical accuracy	MAPE (Mean Average Percent Error) of 5% or lower	MAPE (Mean Average Percent Error) of 25% or lower
Forecast frequency	Produced every hour or every 24 hours	Produced once monthly, quarterly or yearly

Scenario: energy load forecasting

- Energy grid operators need to forecast the load to ensure they can balance supply and demand
- In the short term (hours), operators can manage resources by increasing or decreasing the supply of electricity on the grid
- Data in this example was used in the GEFCom2014 energy load forecasting competition

Energy Demand Forecast E-2-E Solution

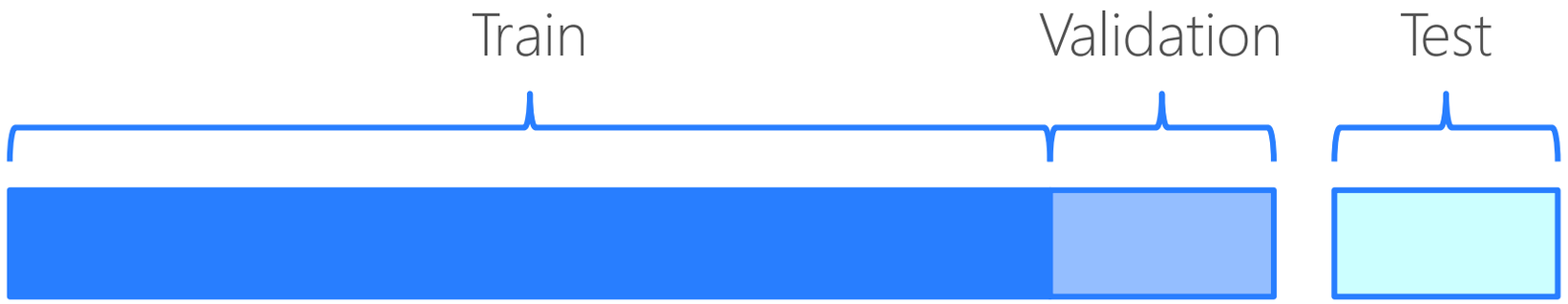


New England ISO data

- 26,000 hourly load values
- Annual, weekly and daily seasonality
- Load is also highly dependent on temperature

Experimental setup

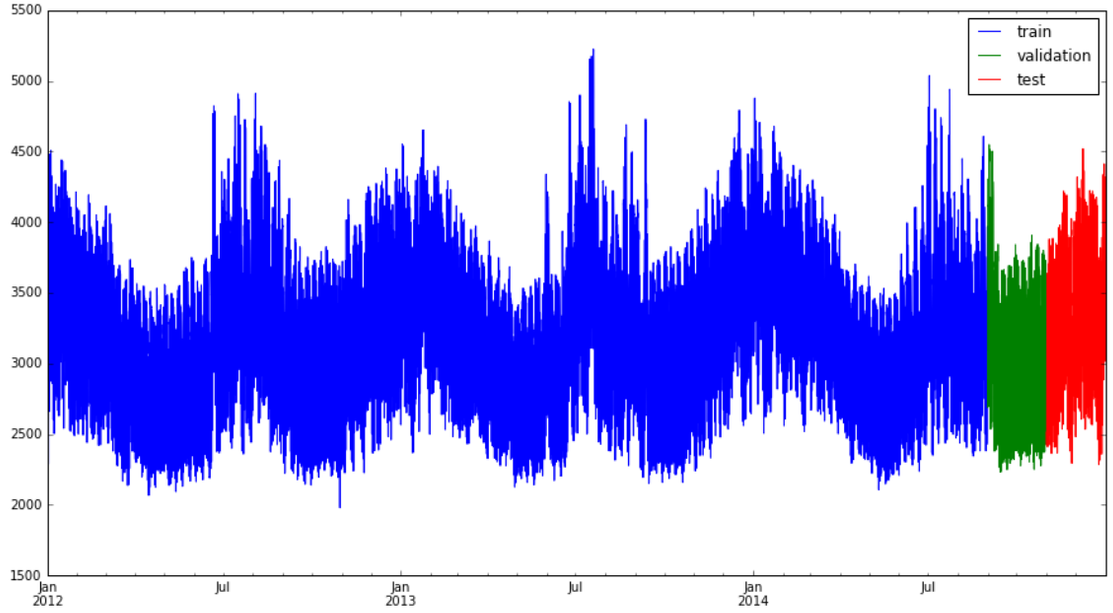
Train, Validation and Test Datasets




```
# split data
data = ...
train, validation, test = split(data)

# tune model hyperparameters
parameters = ...
for params in parameters:
    model = fit(train, params)
    skill = evaluate(model, validation)

# evaluate final model for comparison with other models
model = fit(train)
skill = evaluate(model, test)
```

Training, validation and test sets

- **Train** – used for model training. Contains no information from the other datasets
- **Validation** – used for testing different model hyperparameters
- **Test** – used to evaluate the model and compare to others



Data preparation

Load data into Pandas DataFrame

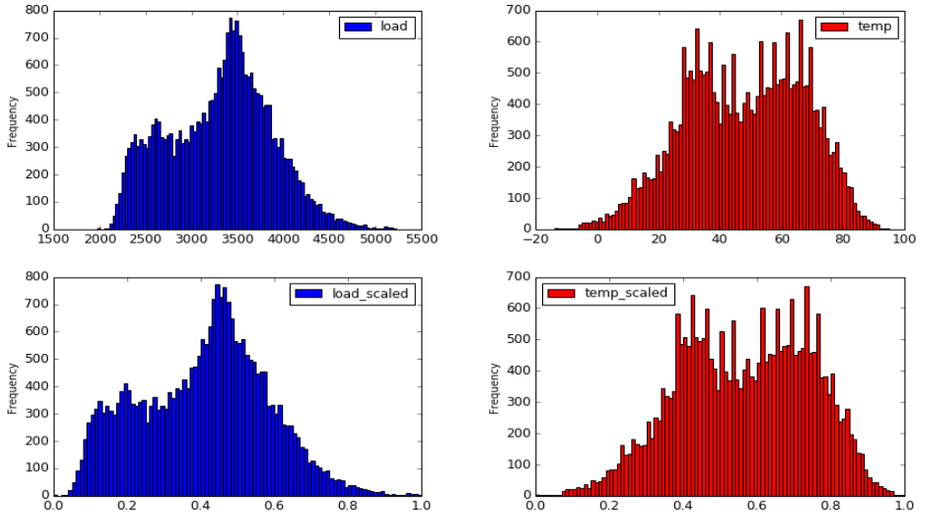
With data loaded in a Pandas DataFrame you can:

- Index the data on timestamp for time-based filtering
- Visualize your data in tables with named columns
- Identify missing periods and missing values
- Create leading and lagged variables

	load	temp
2012-01-01 00:00:00	2,698.00	32.00
2012-01-01 01:00:00	2,558.00	32.67
2012-01-01 02:00:00	2,444.00	30.00
2012-01-01 03:00:00	2,402.00	31.00
2012-01-01 04:00:00	2,403.00	32.00
2012-01-01 05:00:00	2,453.00	31.33
2012-01-01 06:00:00	2,560.00	30.00
2012-01-01 07:00:00	2,719.00	29.00
2012-01-01 08:00:00	2,916.00	29.00
2012-01-01 09:00:00	3,105.00	33.33

Feature scaling

- Large values cause problems for optimization during training
- Optimizers are not scale invariant



ARIMA forecast

Time series forecasting with ARIMA

- Traditionally, time series forecasting is done using curve fitting methods that are aimed at capturing seasonality, trend, and autocorrelation in the data (or some combination of these).

- These methods require a deeper analysis and understanding of the time series to better select the most fitting method.

Autoregressive integrated moving average

@frlazzeri

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q} + e_t$$

AR

- Autoregression. A model that uses the dependent relationship between an observation and some number of lagged observations

I

- Integrated. The use of differencing of raw observations (e.g. subtracting an observation from an observation at the previous time step) in order to make the time series stationary

MA

- Moving Average. A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations

Keras

- High-level API for deep learning
- Simplified syntax for common DL tasks
- Operates over efficient backends (e.g. TensorFlow, CNTK)
- Less control over low-level operations
- Sometimes not as efficient as other frameworks

Exercise

@frlazzeri

1. Go to: https://notebooks.azure.com/frlazzeri/projects/AIConferenceNYC

1. Once you are logged in to Azure Notebooks, on the top right, select '**Clone**' tab. Then type in any name you prefer for '**Project Name**' and '**Project ID**'. Once you have filled in all boxes, click '**Clone**'. Wait till you see a list of files cloned.

2. On the top left, select '**Free Compute**' as compute target option from the dropdown menu.

3. Make sure you see '**Python 3.6**' kernel on the top right for your notebook environment. If not, you can select '**Kernel**', then '**Change kernel**' to make changes.

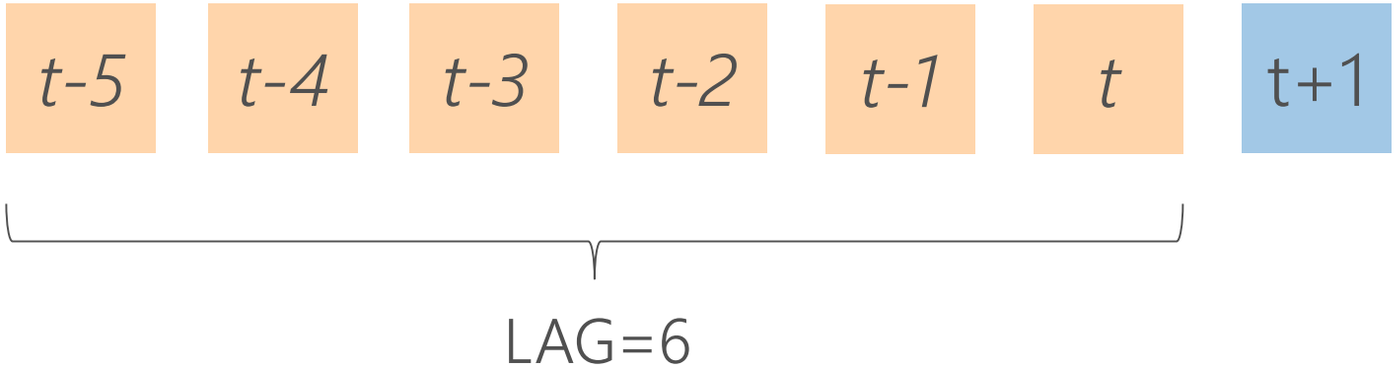
4. Run each cell in the notebook by click '**Run**' on top. This notebook will download sample data to your environment and visualize the data. Wait and make sure you can see all the visualizations. Now you are all set!

Module_4_Financial TS Forecasting

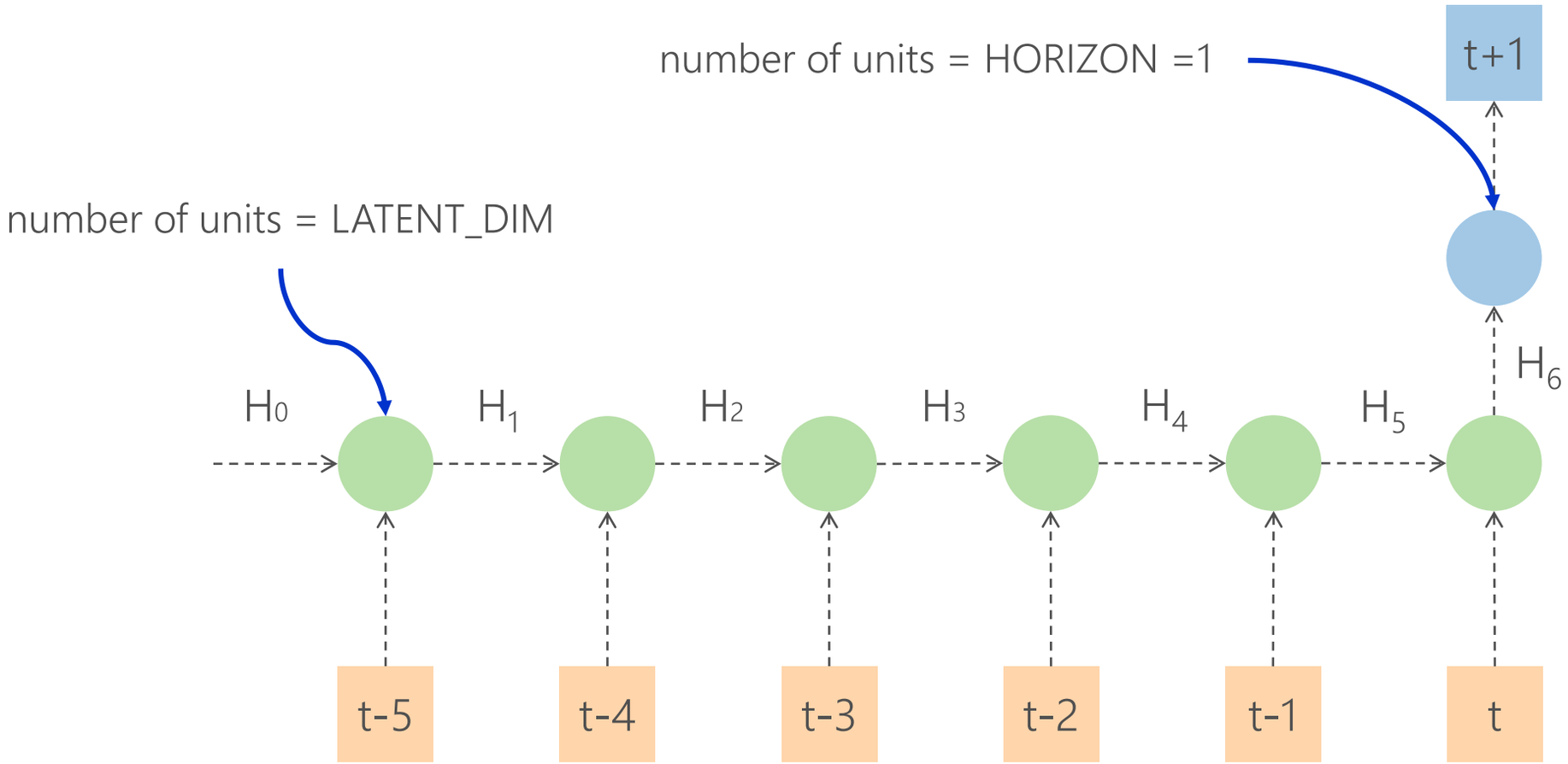
RNN One-step forecast

One-step forecast

- Assuming we are at time *t* …
- … predict the value at time *t+1* …
- … conditional on the previous *LAG* values of the time series



Network structure



Format the data

- Time series needs to be transformed into two tensors:

X	y
(samples, time steps, features)	(samples, horizon)

Format the data

- Time series needs to be transformed into two tensors:

X	y
(samples, time steps, features)	(samples, horizon)
(23370 , 6 , 1)	(23370 , 1)

Format the data

- Step 1: shift *load* to create target variable (y_{t+1}):

assume dataframe is indexed on time *t*, the time the prediction is made

	load	y_t+1
2012-01-01 00:00:00	0.22	0.18
2012-01-01 01:00:00	0.18	0.14
2012-01-01 02:00:00	0.14	0.13
2012-01-01 03:00:00	0.13	0.13
2012-01-01 04:00:00	0.13	0.15
2012-01-01 05:00:00	0.15	0.18
2012-01-01 06:00:00	0.18	0.23
2012-01-01 07:00:00	0.23	0.29
2012-01-01 08:00:00	0.29	0.35
2012-01-01 09:00:00	0.35	0.37

Format the data

- Step 1: shift *load* to create target variable (y_{t+1}):

```
train_shifted['y_t+1'] = train_shifted['load'].shift(-1, freq='H')
```

shift the load value forward 1 time step

ensure shifting is correct, even with missing time periods

Format the data

- Step 2: shift *load* ***LAG*** times to create feature ($load_{t-5}$, … , $load_t$)

	load	y_t+1	load_t-5	load_t-4	load_t-3	load_t-2	load_t-1	load_t-0
2012-01-01 00:00:00	0.22	0.18	nan	nan	nan	nan	nan	0.22
2012-01-01 01:00:00	0.18	0.14	nan	nan	nan	nan	0.22	0.18
2012-01-01 02:00:00	0.14	0.13	nan	nan	nan	0.22	0.18	0.14
2012-01-01 03:00:00	0.13	0.13	nan	nan	0.22	0.18	0.14	0.13
2012-01-01 04:00:00	0.13	0.15	nan	0.22	0.18	0.14	0.13	0.13
2012-01-01 05:00:00	0.15	0.18	0.22	0.18	0.14	0.13	0.13	0.15
2012-01-01 06:00:00	0.18	0.23	0.18	0.14	0.13	0.13	0.15	0.18
2012-01-01 07:00:00	0.23	0.29	0.14	0.13	0.13	0.15	0.18	0.23
2012-01-01 08:00:00	0.29	0.35	0.13	0.13	0.15	0.18	0.23	0.29
2012-01-01 09:00:00	0.35	0.37	0.13	0.15	0.18	0.23	0.29	0.35

Format the data

- Step 2: shift *load* ***LAG*** times to create feature ($load_{t-5}$, … , $load_t$)

```
for t in range(1, W+1):
    train_shifted['load_t-'+str(LAG-t)] = train_shifted['load'].shift(LAG-t, freq='H')
```

Format the data

- Step 3: remove incomplete samples

	load	y_t+1	load_t-5	load_t-4	load_t-3	load_t-2	load_t-1	load_t-0
2012-01-01 00:00:00	0.22	0.18						0.22
2012-01-01 01:00:00	0.18	0.14					0.22	0.18
2012-01-01 02:00:00	0.14	0.13				0.22	0.18	0.14
2012-01-01 03:00:00	0.13	0.13			0.22	0.18	0.14	0.13
2012-01-01 04:00:00	0.13	0.15		0.22	0.18	0.14	0.13	0.13
2012-01-01 05:00:00	0.15	0.18	0.22	0.18	0.14	0.13	0.13	0.15
2012-01-01 06:00:00	0.18	0.23	0.18	0.14	0.13	0.13	0.15	0.18
2012-01-01 07:00:00	0.23	0.29	0.14	0.13	0.13	0.15	0.18	0.23
2012-01-01 08:00:00	0.29	0.35	0.13	0.13	0.15	0.18	0.23	0.29
2012-01-01 09:00:00	0.35	0.37	0.13	0.15	0.18	0.23	0.29	0.35

Format the data

- Step 3: remove incomplete samples:

```
train_shifted = train_shifted.dropna(how='any')
```

	load	y_t+1	load_t-5	load_t-4	load_t-3	load_t-2	load_t-1	load_t-0
2012-01-01 05:00:00	0.15	0.18	0.22	0.18	0.14	0.13	0.13	0.15
2012-01-01 06:00:00	0.18	0.23	0.18	0.14	0.13	0.13	0.15	0.18
2012-01-01 07:00:00	0.23	0.29	0.14	0.13	0.13	0.15	0.18	0.23
2012-01-01 08:00:00	0.29	0.35	0.13	0.13	0.15	0.18	0.23	0.29
2012-01-01 09:00:00	0.35	0.37	0.13	0.15	0.18	0.23	0.29	0.35

Format the data

- Step 4: transpose into numpy arrays

	load	y_t+1	load_t-5	load_t-4	load_t-3	load_t-2	load_t-1	load_t-0
2012-01-01 05:00:00	0.15	0.18	0.22	0.18	0.14	0.13	0.13	0.15
2012-01-01 06:00:00	0.18	0.23	0.18	0.14	0.13	0.13	0.15	0.18
2012-01-01 07:00:00	0.23	0.29	0.14	0.13	0.13	0.15	0.18	0.23

y

```
array([[ 0.18],
       [ 0.23],
       [ 0.29]])
```

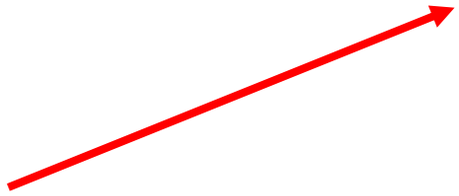
X

```
array([[[ 0.22],
        [ 0.18],
        [ 0.14],
        [ 0.13],
        [ 0.13],
        [ 0.15]],

       [[ 0.18],
        [ 0.14],
        [ 0.13],
        [ 0.13],
        [ 0.15],
        [ 0.18]],

       [[ 0.14],
        [ 0.13],
        [ 0.13],
        [ 0.15],
        [ 0.18],
        [ 0.23]]])
```

Format the data

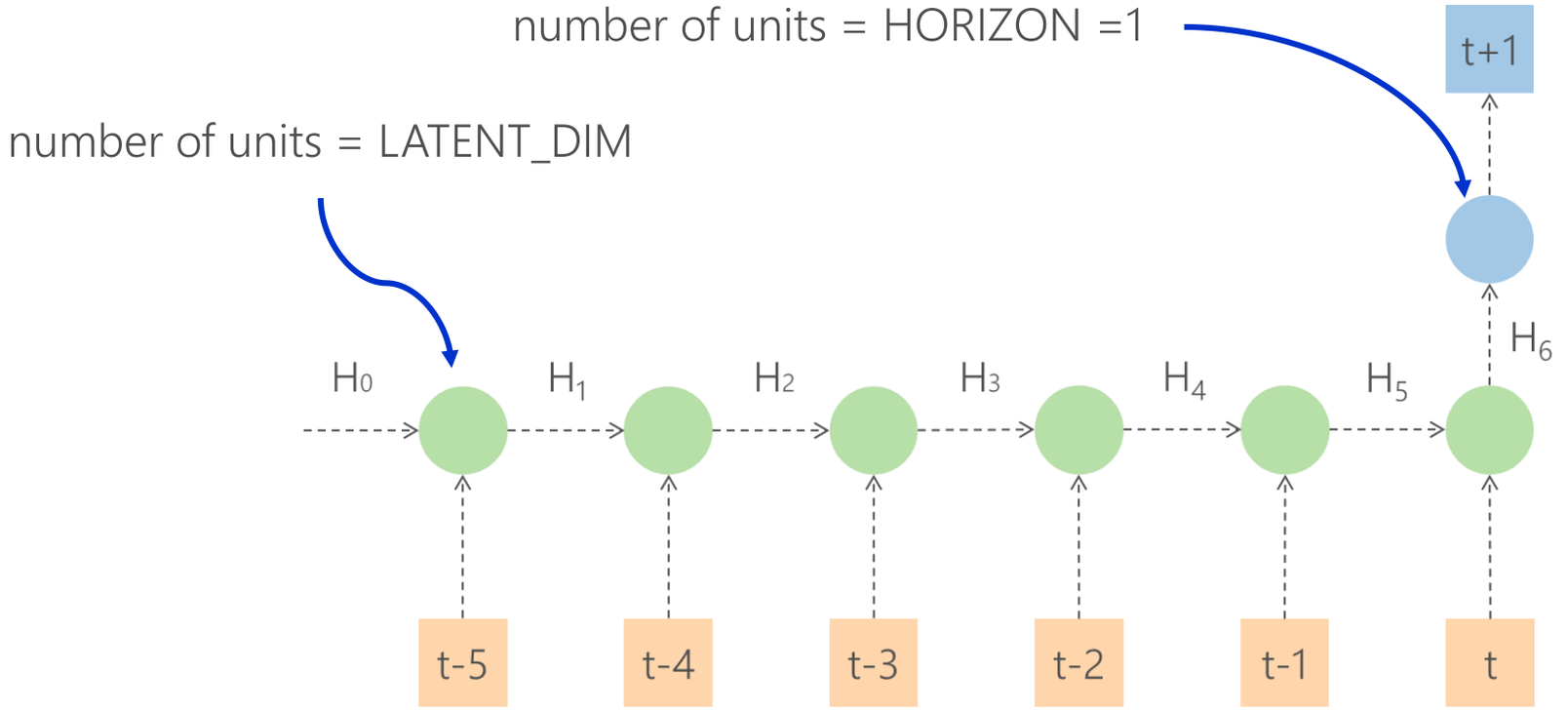
- Step 4: transpose into numpy arrays

```
y_train = train_shifted['y_t+1'].as_matrix()

X_train = train_shifted[['load_t-'+str(LAG-t) for t in range(1, LAG+1)]].as_matrix()
X_train = X_train.reshape(X_train.shape[0], LAG, 1)
```



(samples, time steps features)

Network implementation



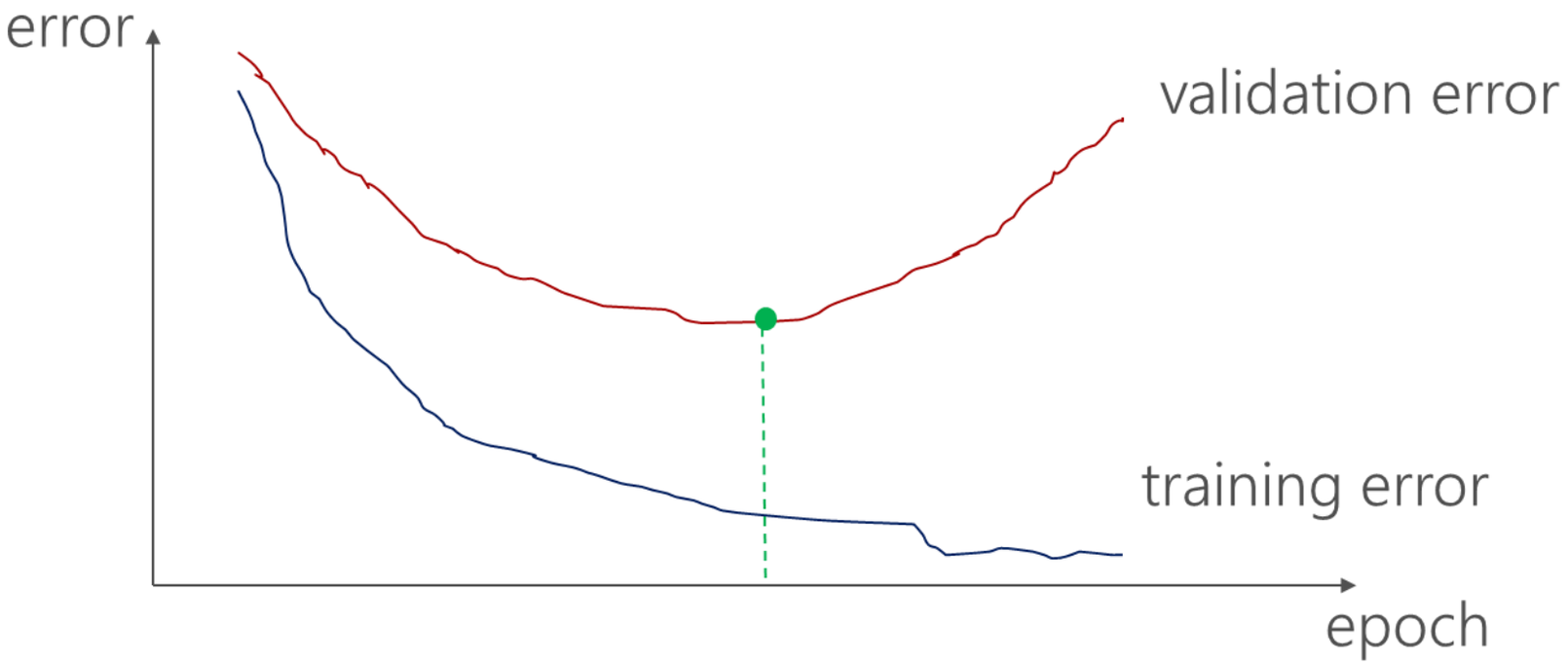

```
model = Sequential()
model.add(GRU(LATENT_DIM, input_shape=(LAG, 1)))
model.add(Dense(HORIZON))
```

Network compilation

- Compiling builds the network's static graph
- Ensures the input/output dimensions of consecutive layers are consistent
- Choose:
  - optimizer (e.g. stochastic gradient descent, Adam, RMSprop)
  - loss function (e.g. mean squared error)

```
model.compile(optimizer='sgd', loss='mse')
```

Early stopping




```
earlystop = EarlyStopping(monitor='val_loss’,
                          min_delta=0,
                          patience=5)
```

Fit the model

stores metrics (e.g. training and validation loss after each epoch)

the maximum epochs, overridden by early stopping criteria

```
history = model.fit(X_train,
                    y_train,
                    batch_size=BATCH_SIZE,
                    epochs=EPOCHS,
                    validation_data=(X_valid, y_valid),
                    callbacks=[earlystop],
                    verbose=1)
```

check against early stopping criteria after each epoch

validation loss computed after each epoch

Evaluation

1. Make predictions on the test set:

```
predictions = model.predict(X_test)
```

2. Compare to the actuals:

	timestamp	h	prediction	actual
0	2014-11-01 05:00:00	t+1	2,693.81	2,714.00
1	2014-11-01 06:00:00	t+1	2,951.02	2,970.00
2	2014-11-01 07:00:00	t+1	3,184.23	3,189.00
3	2014-11-01 08:00:00	t+1	3,308.91	3,356.00
4	2014-11-01 09:00:00	t+1	3,452.54	3,436.00

Evaluation

@frlazzeri

time

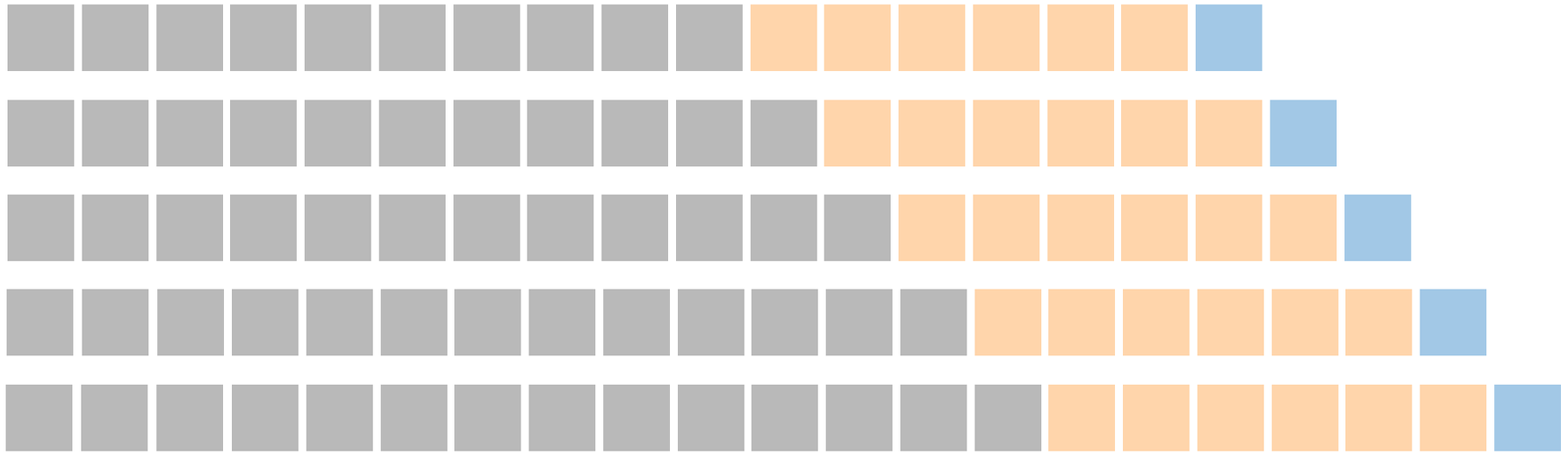
Simulate a “walk-forward” process

Evaluation

time

Simulate a “walk-forward” process

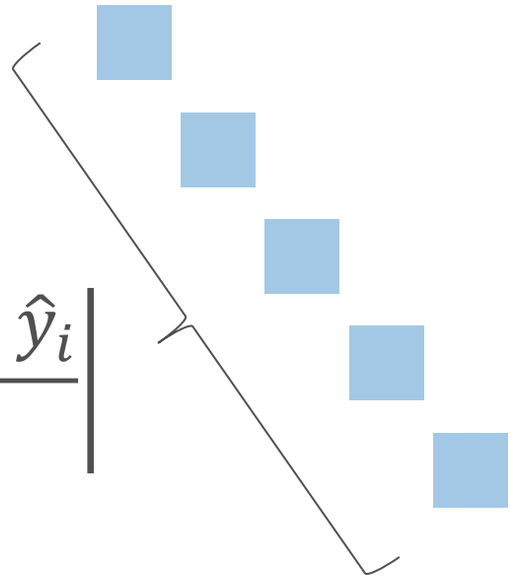
Evaluation



Simulate a “walk-forward” process

Evaluation

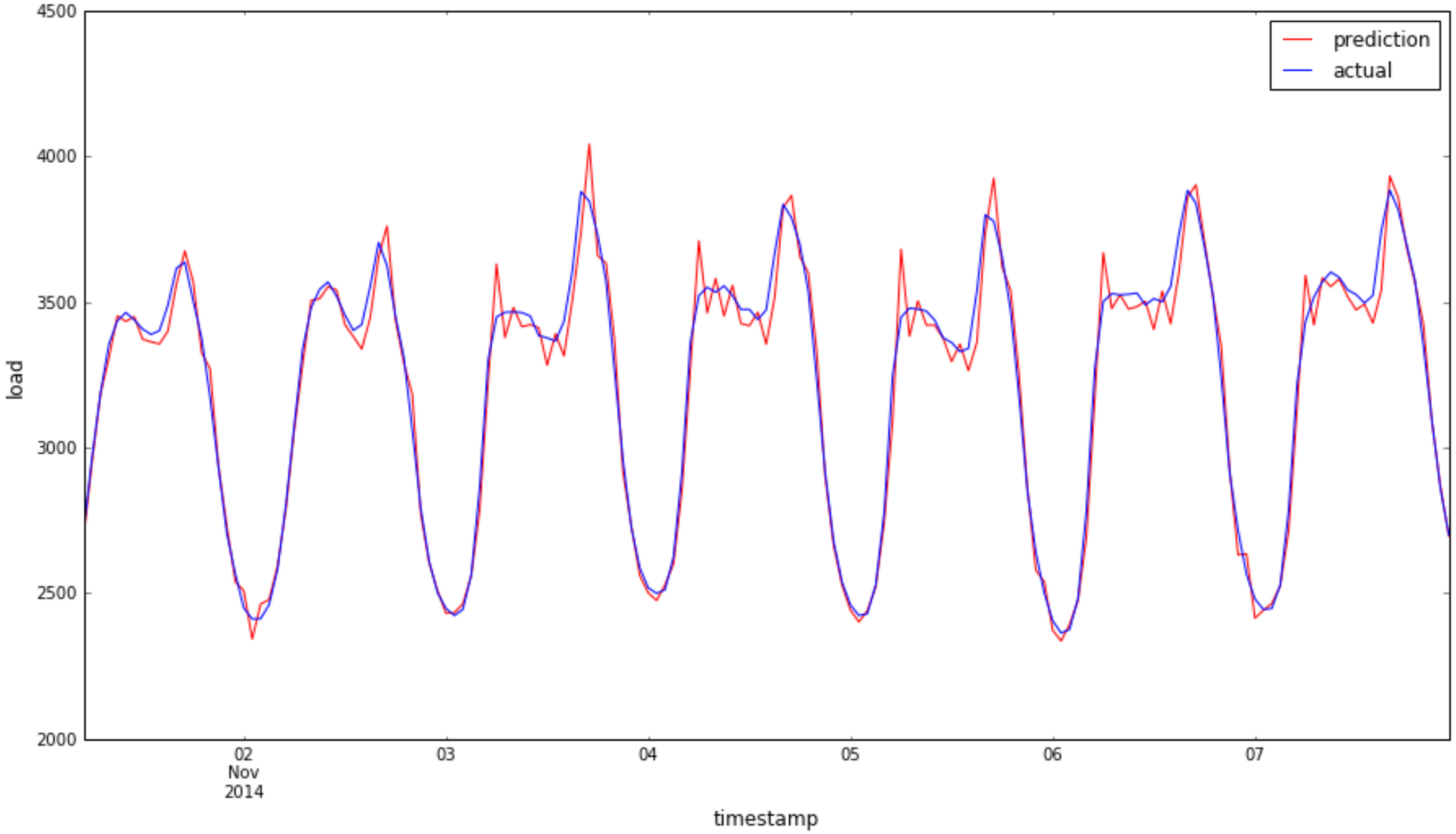
$$\frac{100}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$



Compute evaluation metric e.g. MAPE

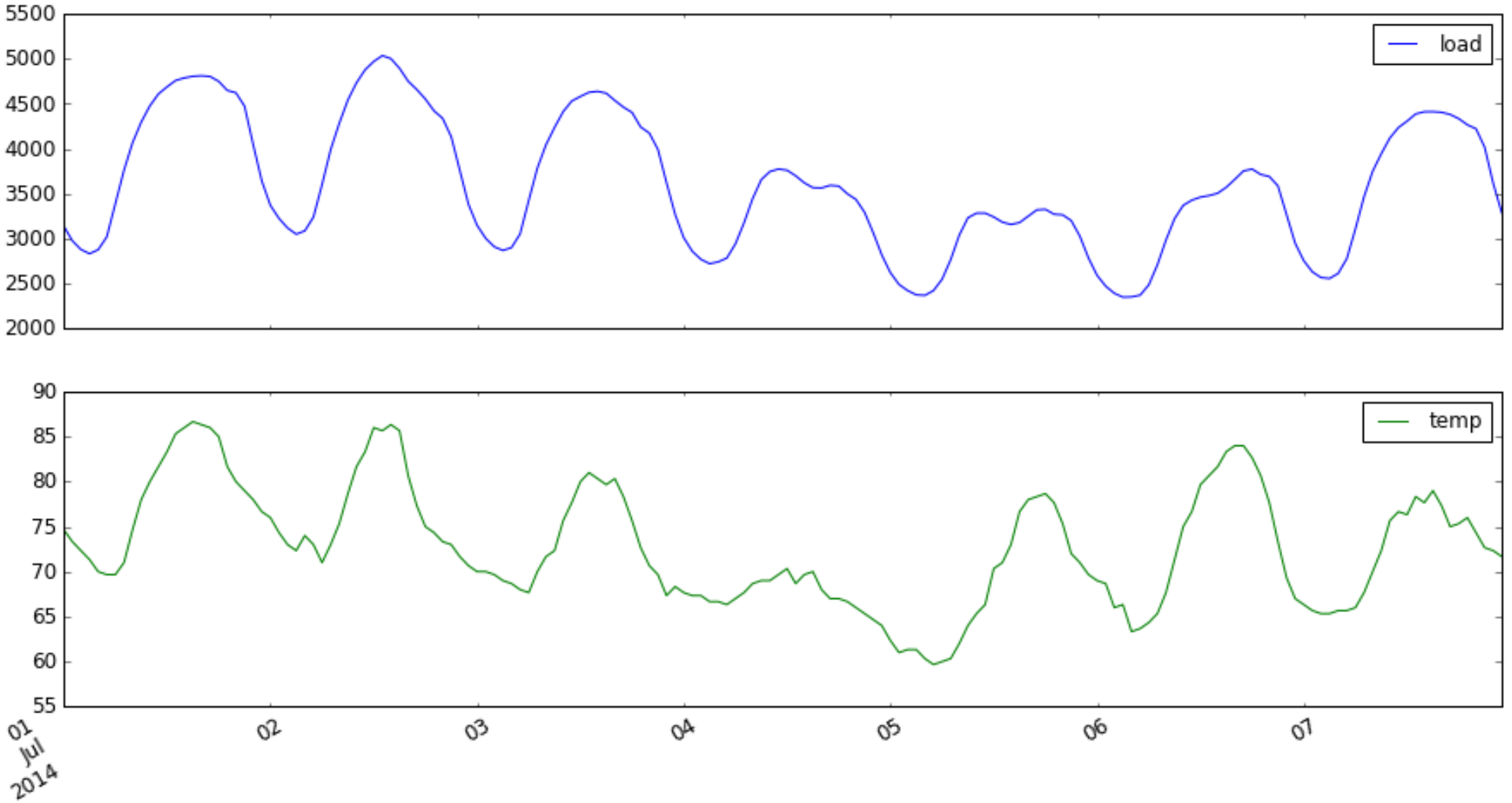
Evaluation

@frlazzeri

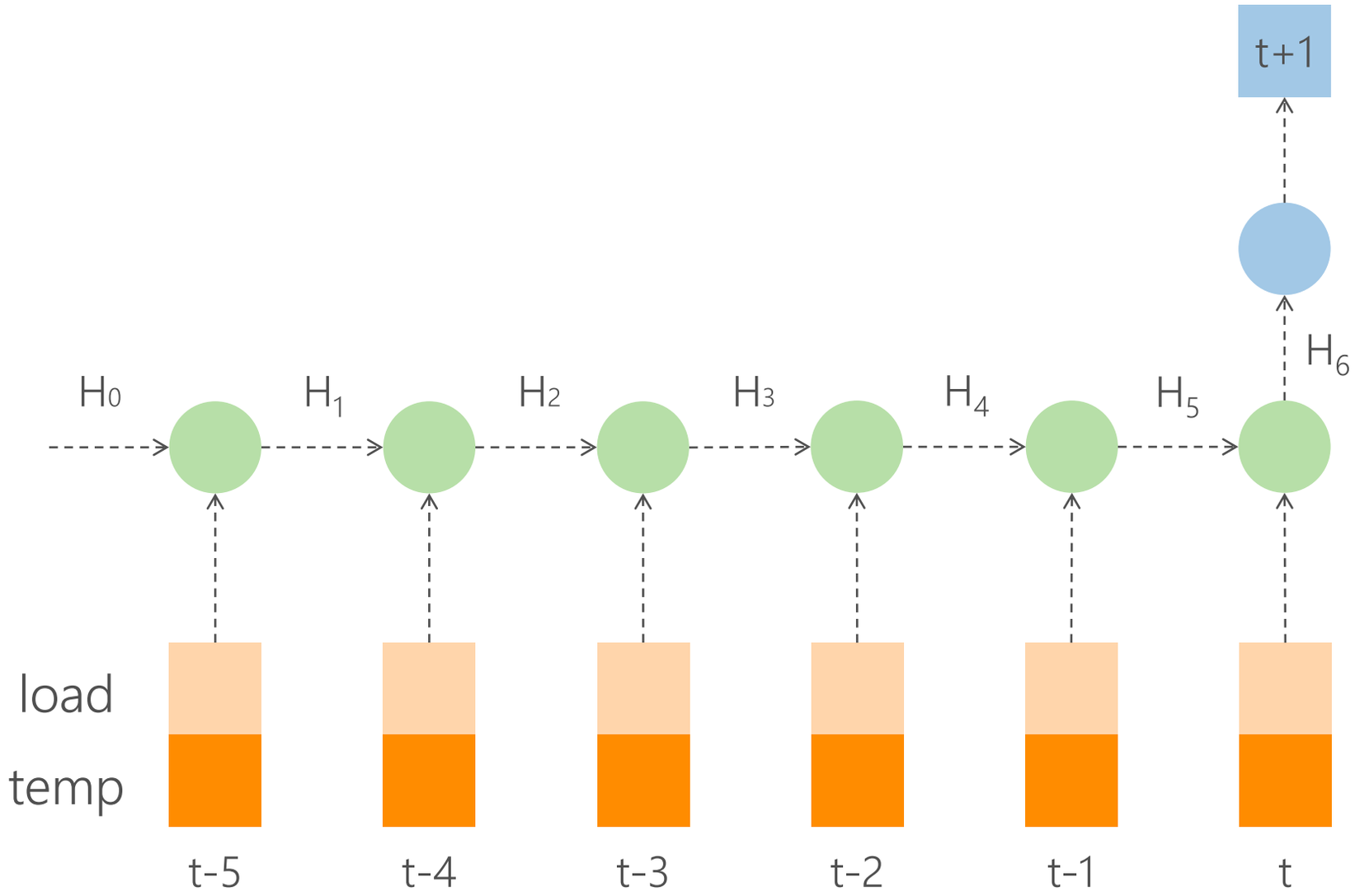


Multivariate input

@frlazzeri



Multivariate input



Multivariate input

feature	y	load						temp					
time step	t+1	t-5	t-4	t-3	t-2	t-1	t-0	t-5	t-4	t-3	t-2	t-1	t-0
2012-01-01 05:00:00	0.18	0.22	0.18	0.14	0.13	0.13	0.15	0.42	0.43	0.40	0.41	0.42	0.41
2012-01-01 06:00:00	0.23	0.18	0.14	0.13	0.13	0.15	0.18	0.43	0.40	0.41	0.42	0.41	0.40
2012-01-01 07:00:00	0.29	0.14	0.13	0.13	0.15	0.18	0.23	0.40	0.41	0.42	0.41	0.40	0.39
2012-01-01 08:00:00	0.35	0.13	0.13	0.15	0.18	0.23	0.29	0.41	0.42	0.41	0.40	0.39	0.39
2012-01-01 09:00:00	0.37	0.13	0.15	0.18	0.23	0.29	0.35	0.42	0.41	0.40	0.39	0.39	0.43
2012-01-01 10:00:00	0.37	0.15	0.18	0.23	0.29	0.35	0.37	0.41	0.40	0.39	0.39	0.43	0.46
2012-01-01 11:00:00	0.37	0.18	0.23	0.29	0.35	0.37	0.37	0.40	0.39	0.39	0.43	0.46	0.50
2012-01-01 12:00:00	0.36	0.23	0.29	0.35	0.37	0.37	0.37	0.39	0.39	0.43	0.46	0.50	0.53
2012-01-01 13:00:00	0.35	0.29	0.35	0.37	0.37	0.37	0.36	0.39	0.43	0.46	0.50	0.53	0.52
2012-01-01 14:00:00	0.36	0.35	0.37	0.37	0.37	0.36	0.35	0.43	0.46	0.50	0.53	0.52	0.54

Multivariate input

(23370, 6, 2)

feature	y	load										temp
time step	t+1	t-5	t-4	t-3	t-2	t-1	t-0	t-5	t-4	t-3	t-2	t-1	t-0
2012-01-01 05:00:00	0.18	0.22	0.18	0.14	0.13	0.13	0.15	0.42	0.43	0.40	0.41	0.42	0.41
2012-01-01 06:00:00	0.23	0.18	0.14	0.13	0.13	0.15	0.18	0.43	0.40	0.41	0.42	0.41	0.40
2012-01-01 07:00:00	0.29	0.14	0.13	0.13	0.15	0.18	0.23	0.40	0.41	0.42	0.41	0.40	0.39

```
array([[[ 0.22,  0.42],
        [ 0.18,  0.43],
        [ 0.14,  0.4 ],
        [ 0.13,  0.41],
        [ 0.13,  0.42],
        [ 0.15,  0.41]],

       [[ 0.18,  0.43],
        [ 0.14,  0.4 ],
        [ 0.13,  0.41],
        [ 0.13,  0.42],
        [ 0.15,  0.41],
        [ 0.18,  0.4 ]],

       [[ 0.14,  0.4 ],
        [ 0.13,  0.41],
        [ 0.13,  0.42],
        [ 0.15,  0.41],
        [ 0.18,  0.4 ],
        [ 0.23,  0.39]]])
```

X

y

```
array([[ 0.18],
       [ 0.23],
       [ 0.29]])
```

Exercise

@frlazzeri

1. Go to: https://notebooks.azure.com/frlazzeri/projects/AIConferenceNYC

1. Once you are logged in to Azure Notebooks, on the top right, select '**Clone**' tab. Then type in any name you prefer for '**Project Name**' and '**Project ID**'. Once you have filled in all boxes, click '**Clone**'. Wait till you see a list of files cloned.

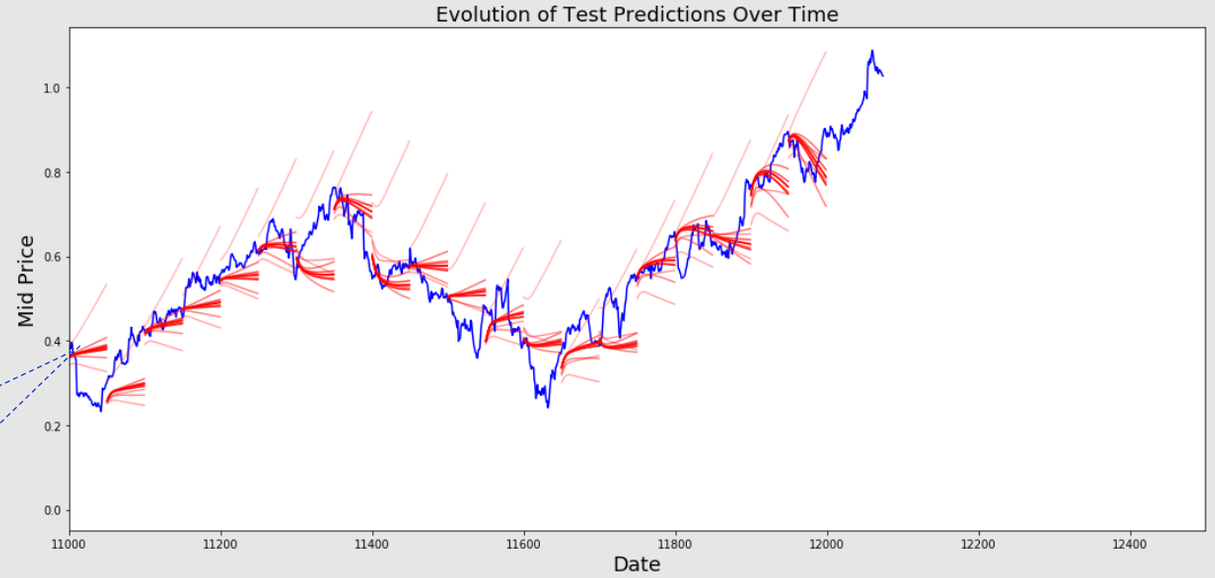
2. On the top left, select '**Free Compute**' as compute target option from the dropdown menu.

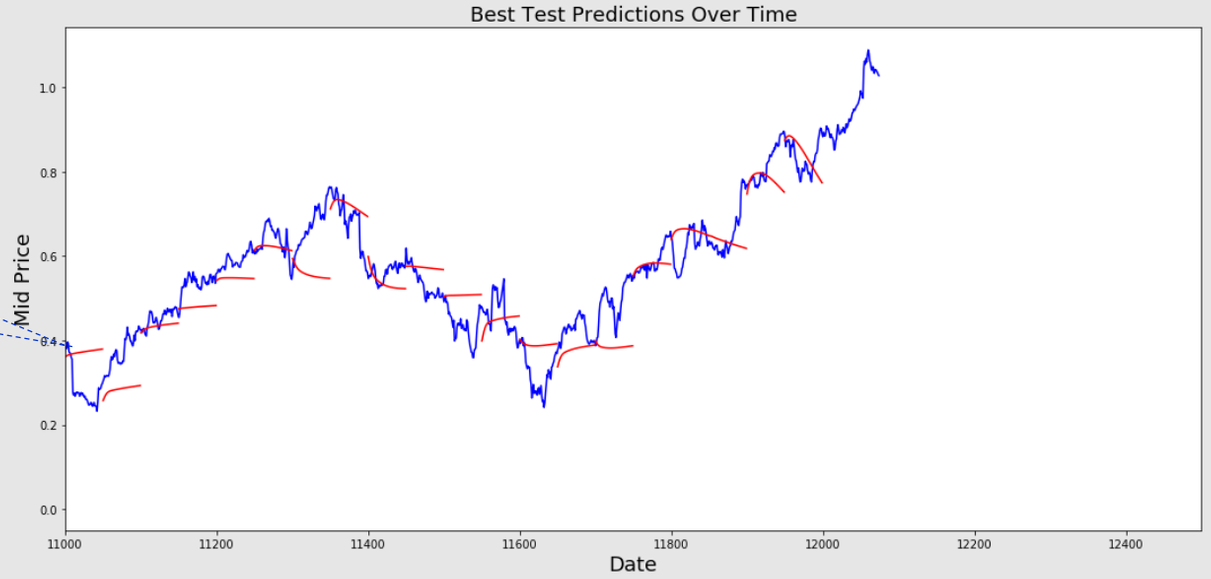
3. Make sure you see '**Python 3.6**' kernel on the top right for your notebook environment. If not, you can select '**Kernel**', then '**Change kernel**' to make changes.

4. Run each cell in the notebook by click '**Run**' on top. This notebook will download sample data to your environment and visualize the data. Wait and make sure you can see all the visualizations. Now you are all set!

Module_4_Financial TS Forecasting

Scenario: Stock market predictions





Why Do You Need Time Series Models?

- You would like to model stock prices correctly, so as a stock buyer you can reasonably decide when to buy stocks and when to sell them to make a profit.

- This is where time series modelling comes in.

- You need good machine learning models that can look at the history of a sequence of data and correctly predict what the future elements of the sequence are going to be.

Stock Prices

- **Open**: Opening stock price of the day
- **Close**: Closing stock price of the day
- **High**: Highest stock price of the data
- **Low**: Lowest stock price of the day

	Date	Open	High	Low	Close
0	1970-01-02	0.30627	0.30627	0.30627	0.30627
1	1970-01-05	0.30627	0.31768	0.30627	0.31385
2	1970-01-06	0.31385	0.31385	0.30996	0.30996
3	1970-01-07	0.31385	0.31385	0.31385	0.31385
4	1970-01-08	0.31385	0.31768	0.31385	0.31385

Data

Alphavantage

- Alpha Vantage. Before you start, however, you will first need an API key, which you can obtain for free here. After that, you can assign that key to the api_key variable.
- You will first load in the data from Alpha Vantage. Since you're going to make use of the American Airlines Stock market prices to make your predictions, you set the ticker to "AAL".

Kaggle

- Data found on Kaggle is a collection of csv files and you don't have to do any preprocessing, so you can directly load the data into a Pandas DataFrame.
- Use the data from this page. You will need to copy the *Stocks* folder in the zip file to your project home folder.

Exercise

@frlazzeri

1. Go to: https://notebooks.azure.com/frlazzeri/projects/AIConferenceNYC

1. Once you are logged in to Azure Notebooks, on the top right, select '**Clone**' tab. Then type in any name you prefer for '**Project Name**' and '**Project ID**'. Once you have filled in all boxes, click '**Clone**'. Wait till you see a list of files cloned.

2. On the top left, select '**Free Compute**' as compute target option from the dropdown menu.

3. Make sure you see '**Python 3.6**' kernel on the top right for your notebook environment. If not, you can select '**Kernel**', then '**Change kernel**' to make changes.

4. Run each cell in the notebook by click '**Run**' on top. This notebook will download sample data to your environment and visualize the data. Wait and make sure you can see all the visualizations. Now you are all set!

Module_4_Financial TS Forecasting

Microsoft

@frlazzeri

Conclusion

Additional Resources

@frlazzeri

- Azure Machine Learning Service: https://aka.ms/AzureMLService
  - Data Prep:
    - What is Data Prep? http://aka.ms/AzureMLDataPrep (video)
    - Data Prep documentation: http://aka.ms/AzureMLDataPrepNB
  - Automated ML:
    - What is Automated ML? https://aka.ms/AutomatedML
    - Automated ML documentation: https://aka.ms/AutomatedMLDocs

- Lear More about Time Series Forecasting: http://aka.ms/LazzeriMLForecast
- Azure Notebooks: https://aka.ms/AzureNB
- Python Microsoft: https://aka.ms/PythonMS

- Additional data sets for time series analysis:
  https://www.kaggle.com/c/rossmann-store-sales
  https://www.kaggle.com/c/recruit-restaurant-visitor-forecasting

Microsoft

O'REILLY

Have questions about this session?

I'll be at the Microsoft Booth #208

Grab some SWAG!

For more details visit https://conferences.oreilly.com/artificial-intelligence/ai-ny

Microsoft

Thank You!

IN-PERSON TRAINING

***Francesca Lazzeri**, PhD*
@frlazzeri

Wee Hyong Tok
@weehyong

Krishna Anumalasetty
@KrishnaAnumalas