# Let the Tensors Flow!

## An introduction to Google's Deep-Learning Framework

Pascal Pompey
Research Engineer,
Predictive analytics
IBM Dublin Research Laboratory
Pascal.pompey@gmail.com

# Menu

- Come on! Turn to the deep side of learning!
- Why Tensorflow?
- Tensorflow's main concepts
  - Hello world application
  - Standard NN application
- BREAK!!! May the beer be with you!
- Recap
- Recommended architecture of tensor flow projects
- SNLI

# Menu

- **Deep learning: A new hope**
  - Come on! Turn to the deep side of learning!
  - Why Tensorflow?
  - Tensorflow's main concepts
  - Hello world application
  - Standard NN application
- BIKE ACCIDENT, Wait for episode 2

Come on! Turn to the deep-side of learning

# Come on! Turn to the deep-side of learning

- Why deep learning?
- Historical context
  - Machine-learning & Data-mining
    - Bits and pieces to solve different problem
  - Data-science:
    - Solving complex problems
    - Many data-sources, unstructured-data
- Requirements of data-science
  - Integrate heterogeneous data-sources
  - Explore and generate complex feature spaces
  - Integrate many different techniques together in a single pipeline
  - Consolidate methods and workflows

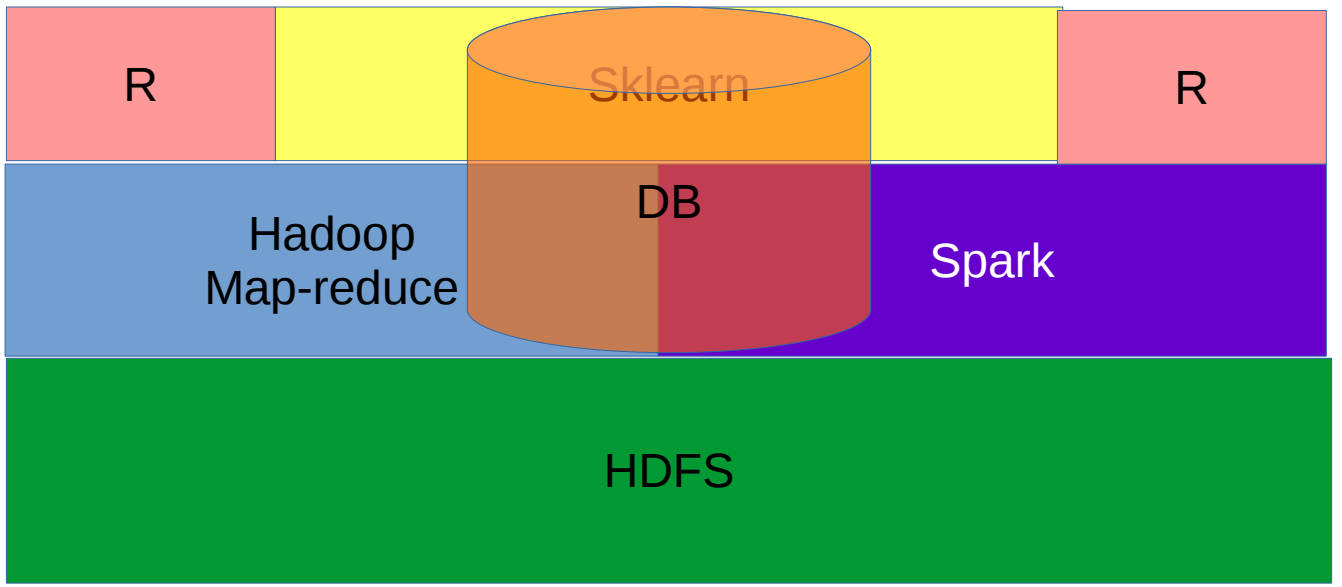
# Come on! Turn to the deep-side of learning

- Why deep learning?
- Historical context
  - Machine-learning & Data-mining
    - Bits and pieces to solve different problem
  - Data-science:
    - Solving complex problems
    - Many data-sources, unstructured-data
- Requirements of data-science
  - Integrate heterogeneous data-sources
  - Explore and generate complex feature spaces
  - Integrate many different techniques together in a single pipeline
  - Consolidate methods and workflows

# Complex data-science applications

## How it used to be done before

- Data-flows: Different frameworks
- Learning: Different algorithms
- Programming: Different libraries / languages



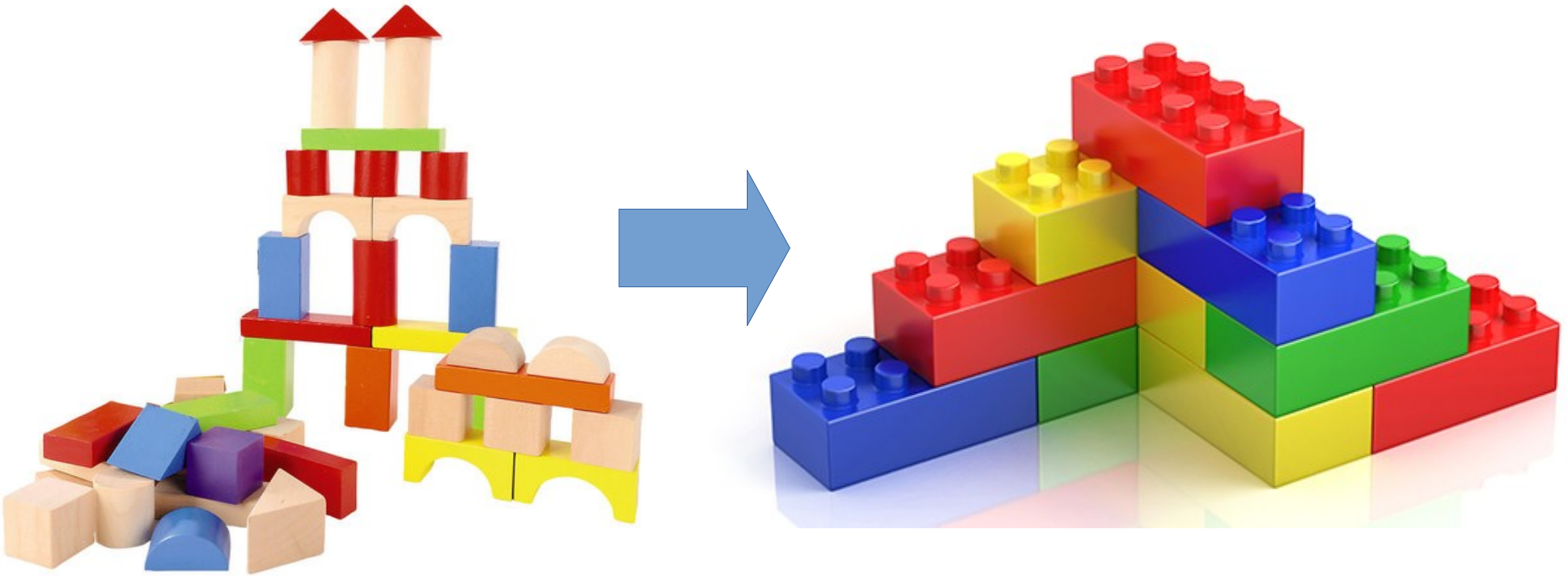
# Complex data-science applications

## How it used to be done before

# Complex data-science applications

## How it used to be done before

# The Lego idea



# Specification of a data-science lego block

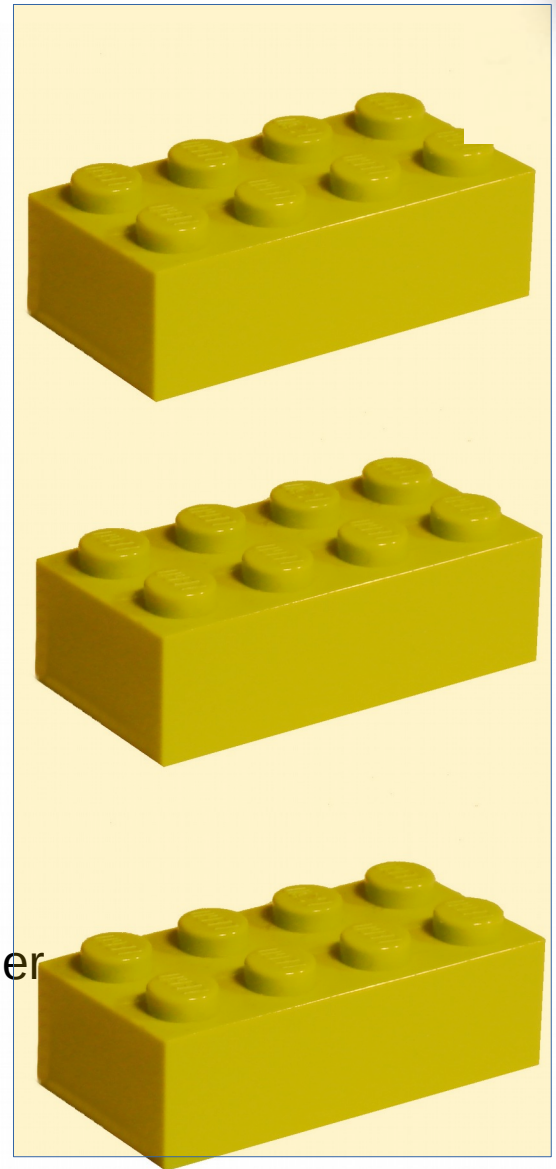
- Needs to be able to learn anything
- Needs to be able to fit in an encompassing learning problem
- Needs to scale horizontally for processing and learning

# Deep-Learning and Lego

- Neural Network: Universal approximator
- Back-propagation: ties it all together

# Deep learning and lego

Image descriptor



ConvNet

Language-model (lstm)

ConvNet

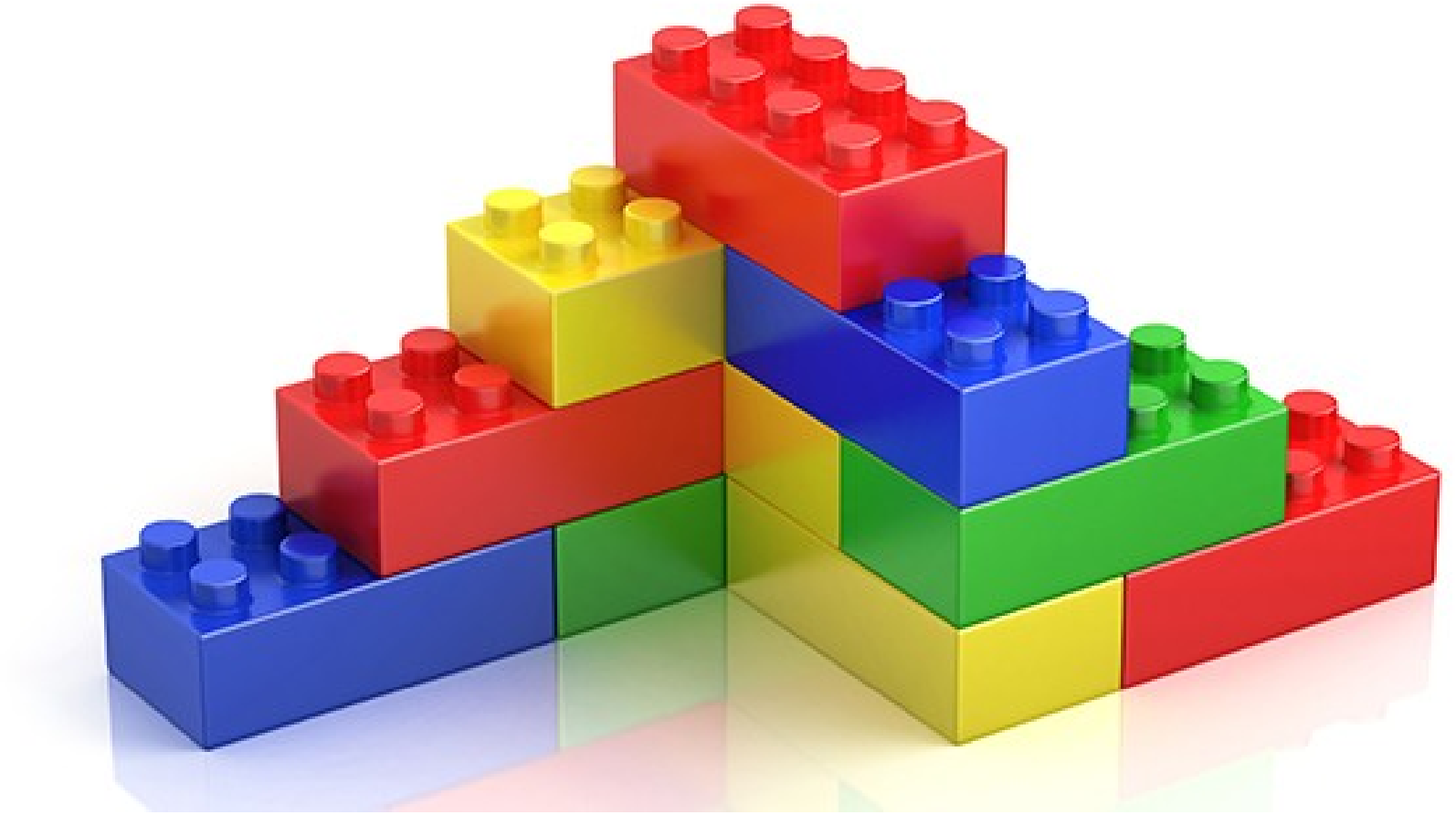
NER

Auto-encoder

Word-embedding

# The Lego idea

## Deep-learning implements the lego interface

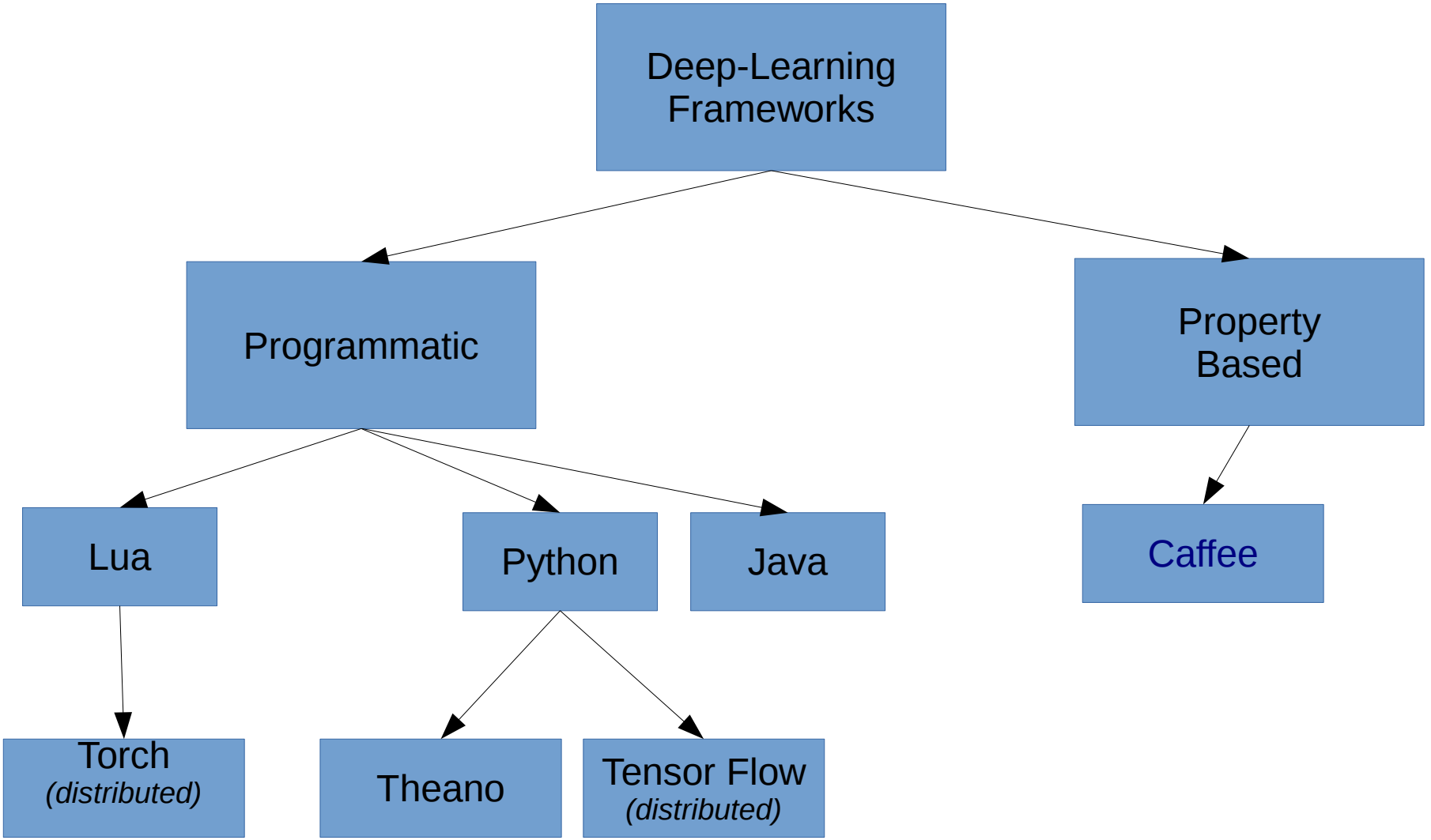


# Deep-Learning and Lego

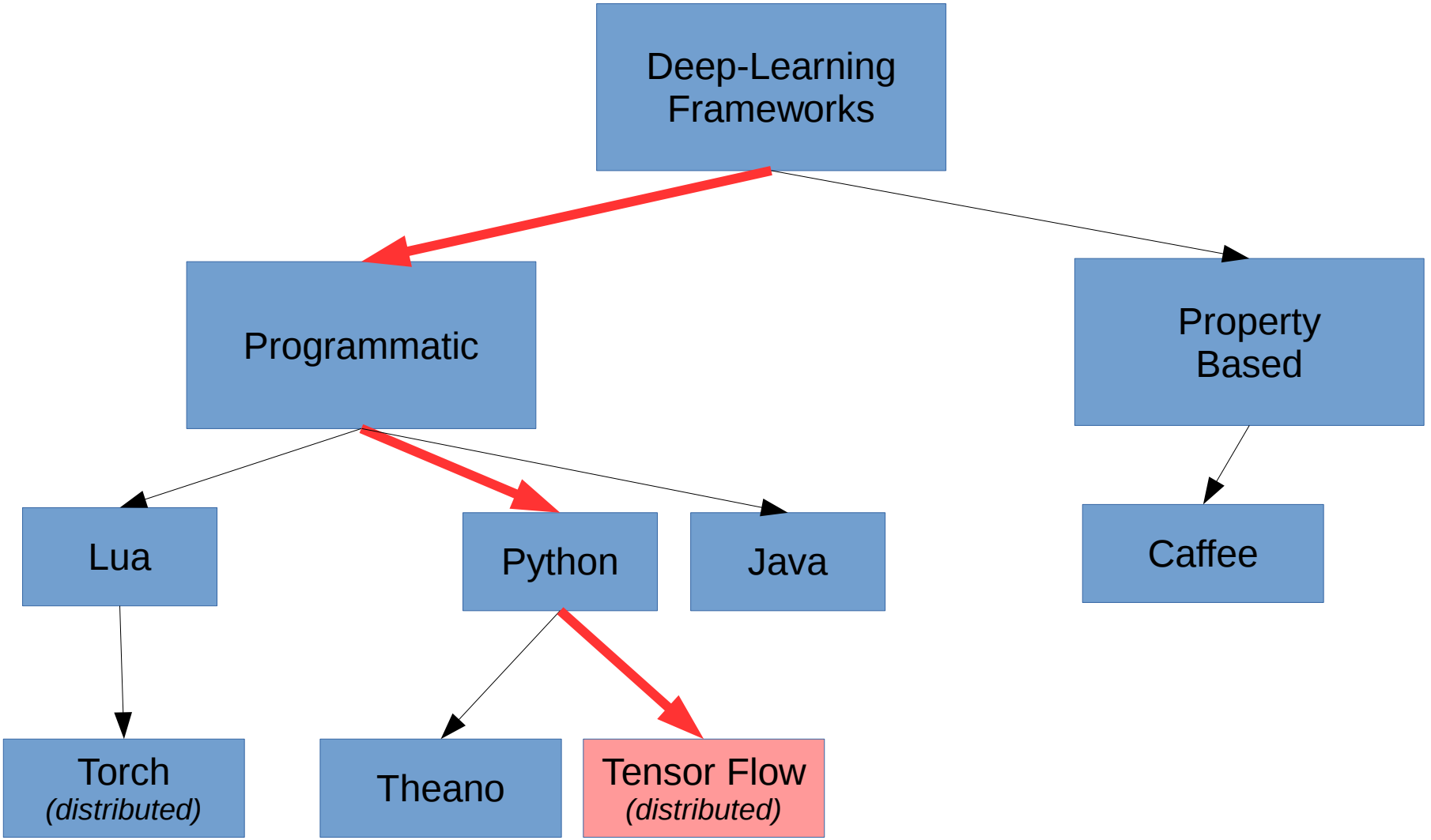
- Neural Network: Universal approximator
- Back-propagation: ties it all together

# Why Tensor-Flow?

# Deep-learning libraries classified



# Deep-learning libraries classified



# The Tensor-Flow school of deep-learning

Underpinning concepts of tensor-flow

# What are tensors?

- Double: 3.14159..
- Vector: [1.0, 2.0, 3.0, 5.0, 8.0, 13.0, …]
  - e.g. time-series
- Matrix: [ [1, 2], [1, 4], [1, 8], [1, 16], ... ]
  - e.g. images
- Tensor: [ [matrix], [matrix], [matrix], [matrix] ]
  - e.g. list of images

# Why tensors?

## What are tensors?

- Double: 3.14159.. ← Tensor of rank 0, shape []
- Vector: [1.0, 2.0, 3.0, 5.0, 8.0, 13.0, …]
  - e.g. time-series ← Tensor of rank 1, shape [n]
- Matrix: [ [1, 2], [1, 4], [1, 8], [1, 16], ... ]
  - e.g. images ← Tensor of rank 2, shape [n, m]
- Tensor: [ [matrix], [matrix], [matrix], [matrix] ]
  - e.g. list of images ← Tensor of rank 3, shape [n, m,k]
- Tensor of rank r, shape [d1, d2, …]
  - You get the idea

Tensors are defined by a shape and a type

Tensors are the main data-structure holding data in tensor flow

# Why tensors?

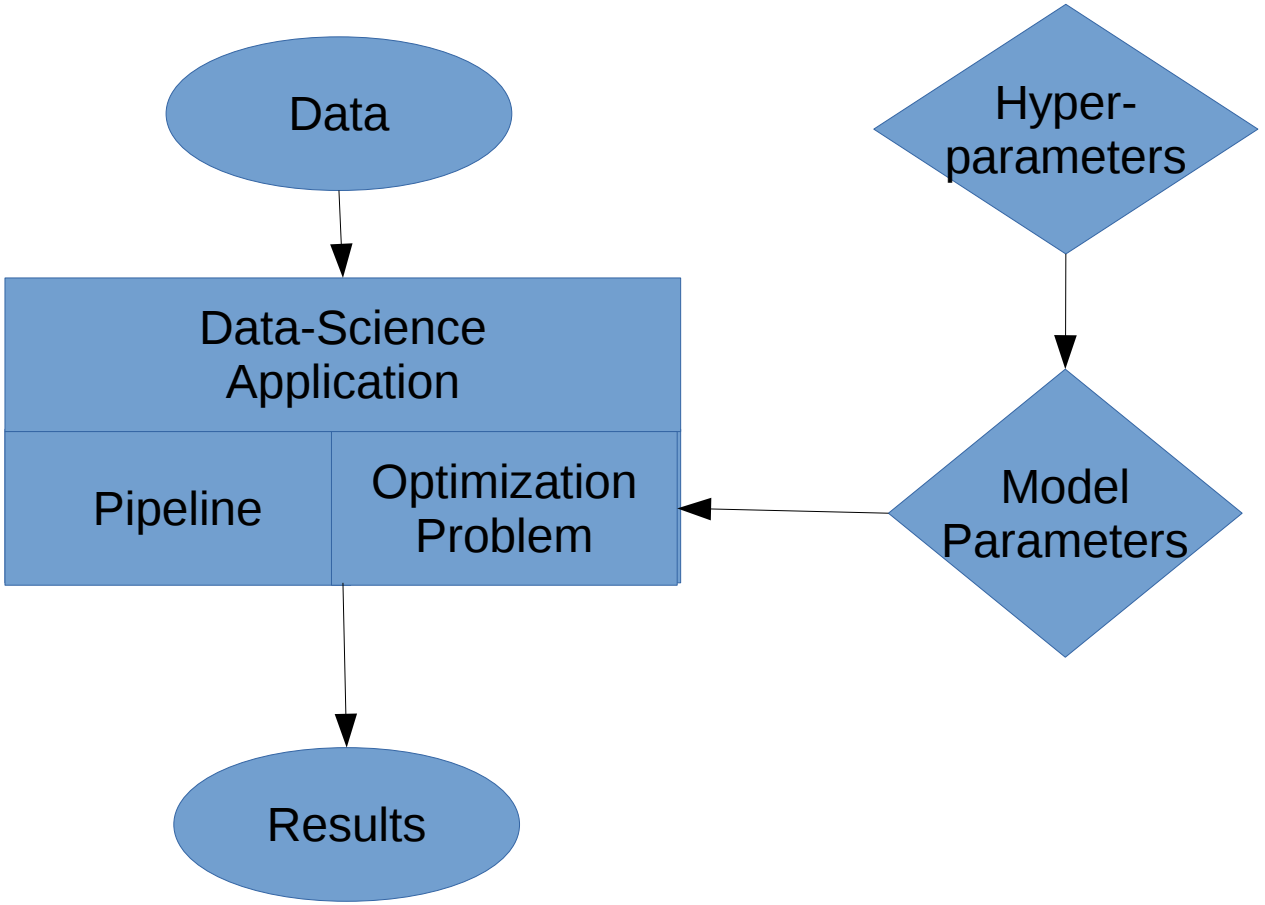
## What are tensors?

- Double: 3.14159.. ← Tensor of rank 0, shape []
- Vector: [1.0, 2.0, 3.0, 5.0, 8.0, 13.0, …]
  - e.g. time-series ← Tensor of rank 1, shape [n]
- Matrix: [ [1, 2], [1, 4], [1, 8], [1, 16], ... ]
  - e.g. images ← Tensor of rank 2, shape [n, m]
- Tensor: [ [matrix], [matrix], [matrix], [matrix] ]
  - e.g. list of images ← Tensor of rank 3, shape [n, m,k]
- Tensor of rank r, shape [d1, d2, …]
  - You get the idea

Tensors are defined by a shape and a type

Tensors are the main data-structure holding data in tensor flow

# Ingredients of data-science applications



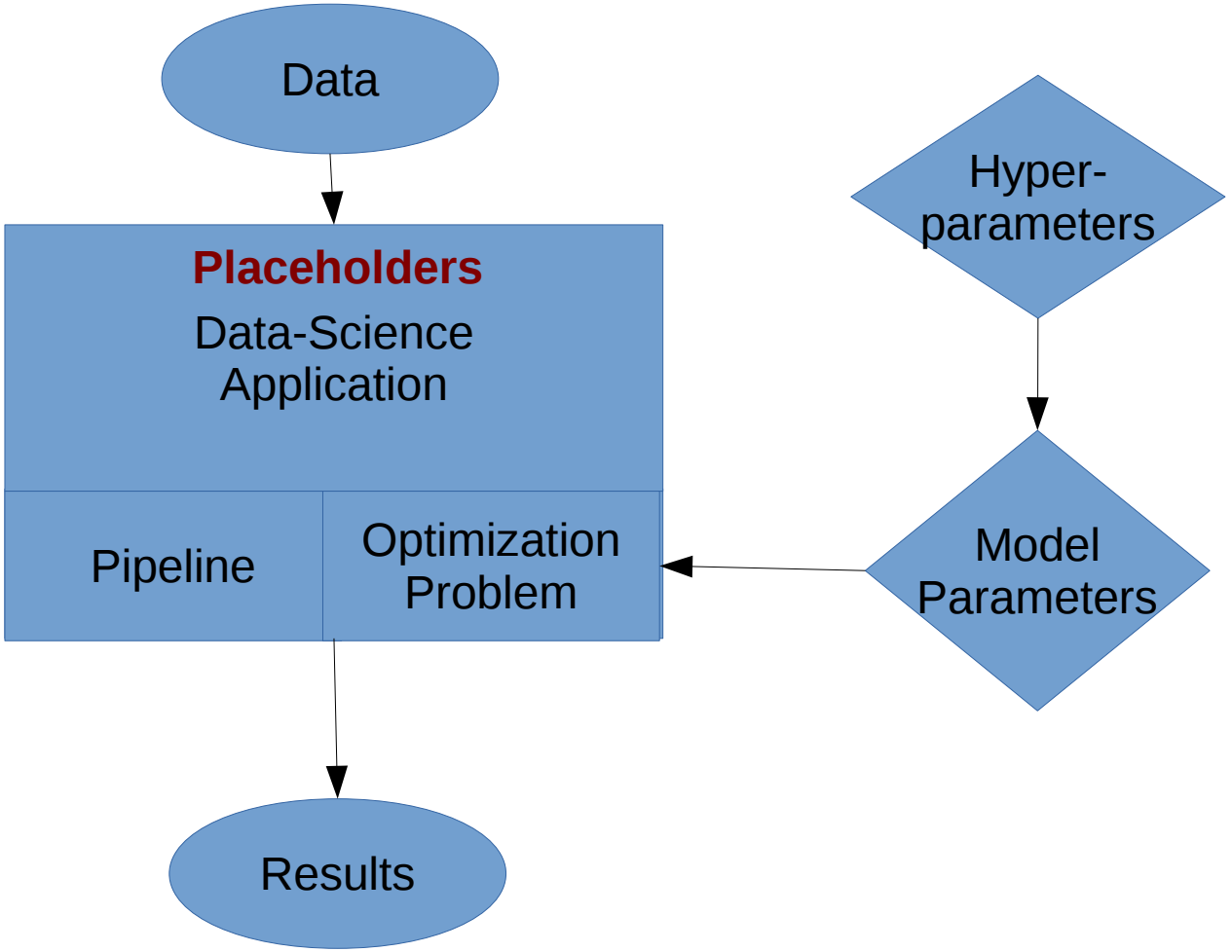
# Let's do it!

Sessions + data-dictionary + placeholders

## “Hello data!” in tensorflow

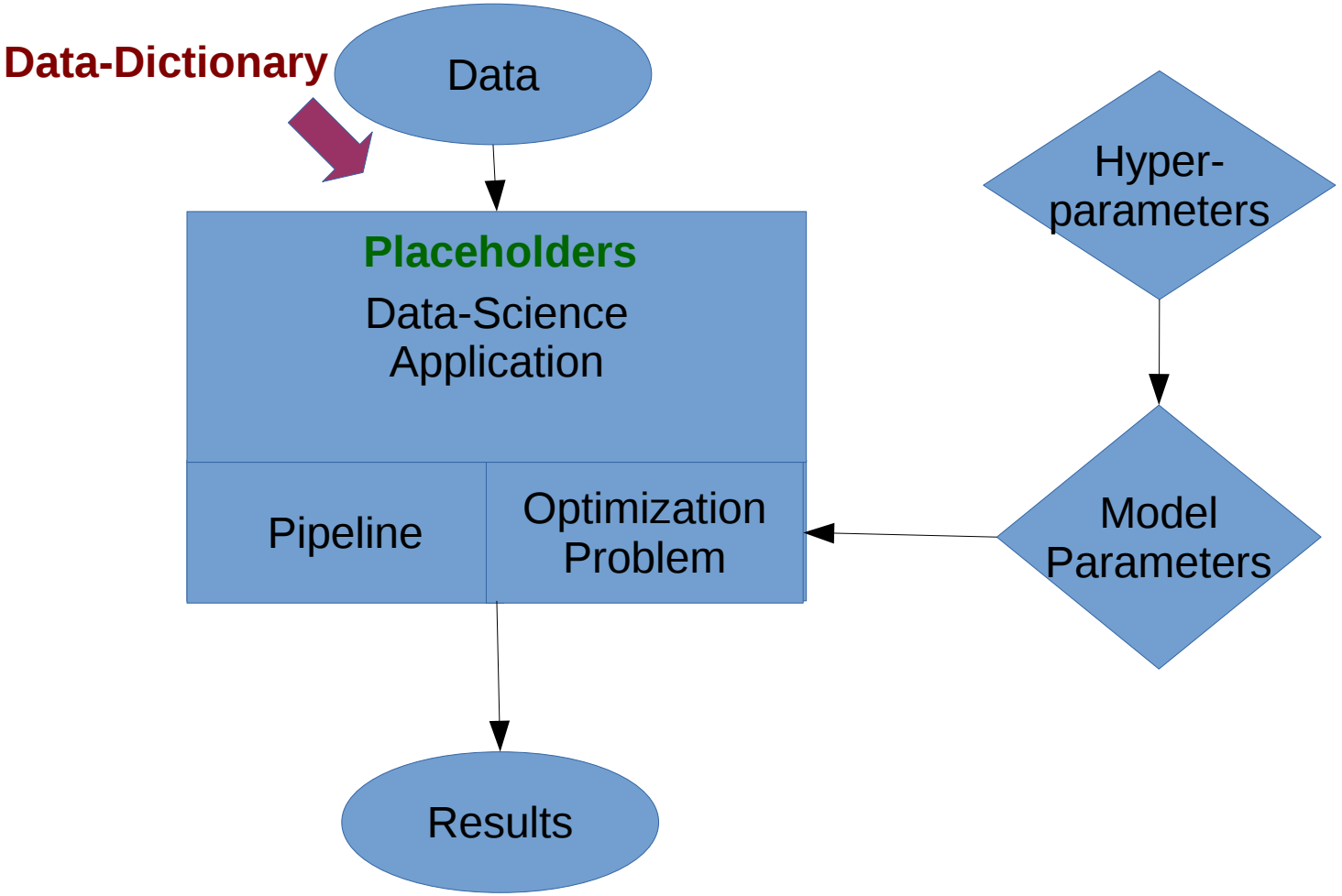
# Defining inputs in tensor flow

Placeholders = source of the flow




```
application_input = tf.placeholder( name='..', dtype=tf.int32, shape=(..) )
```

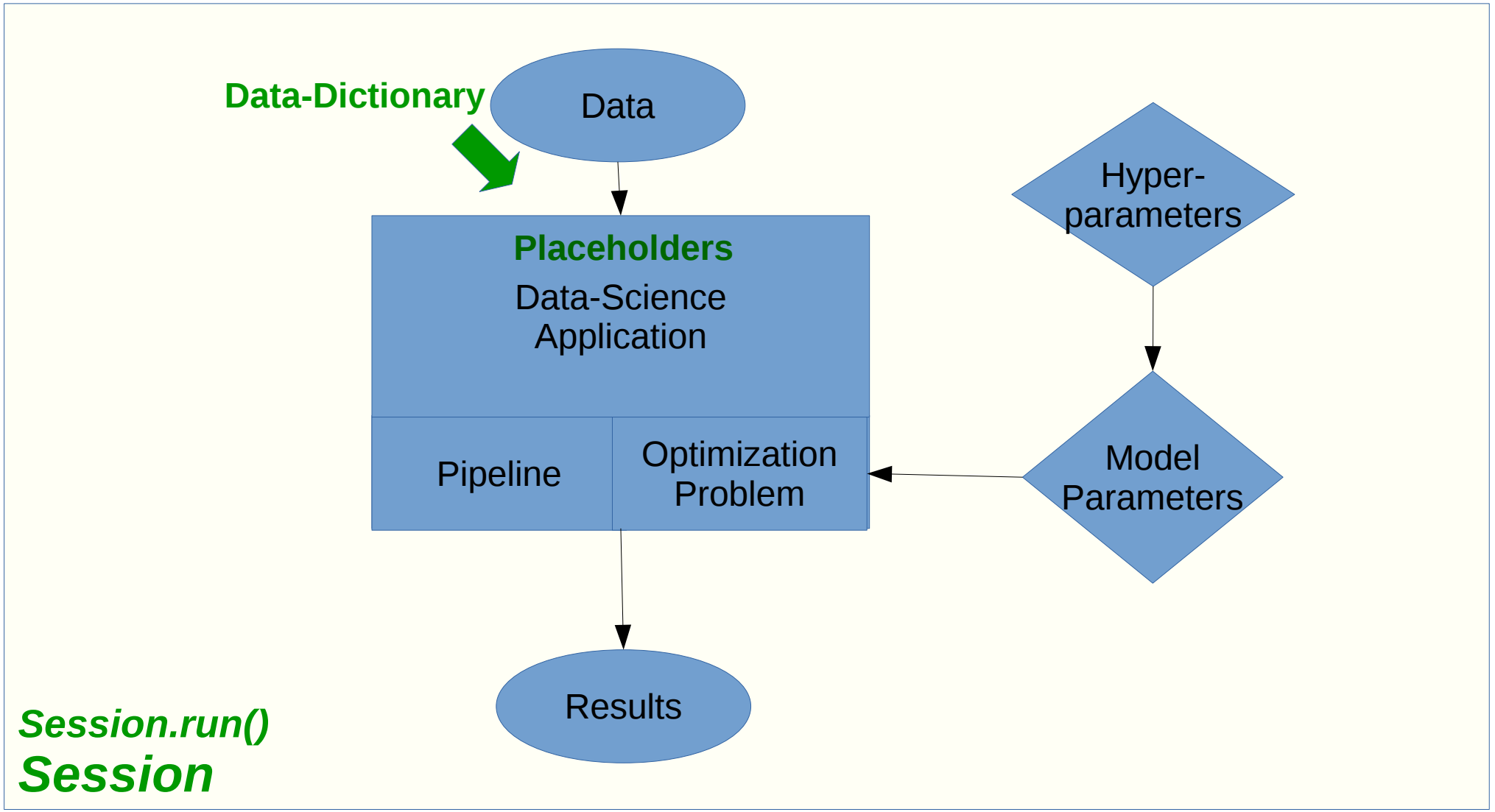
# Passing data to applications
# Data-dictionary




feed_dict = {placeholder_variable : values}

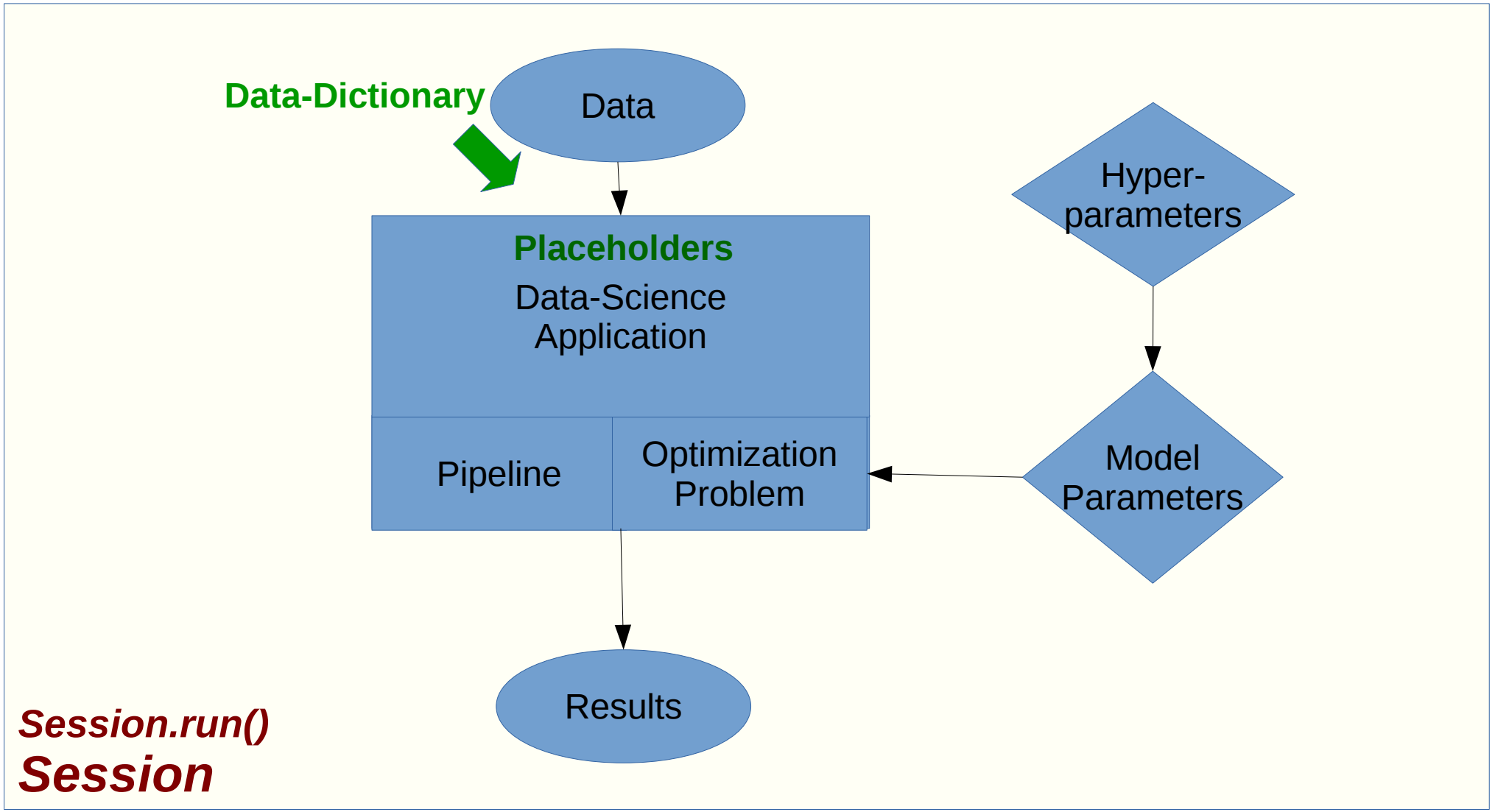
# Outputting data in tensor flow

Session.run() returns target tensors




retrieved_tensor = session.run(tensors_to_retrieve, feed_dict)

# Running applications in tensorflow: Session



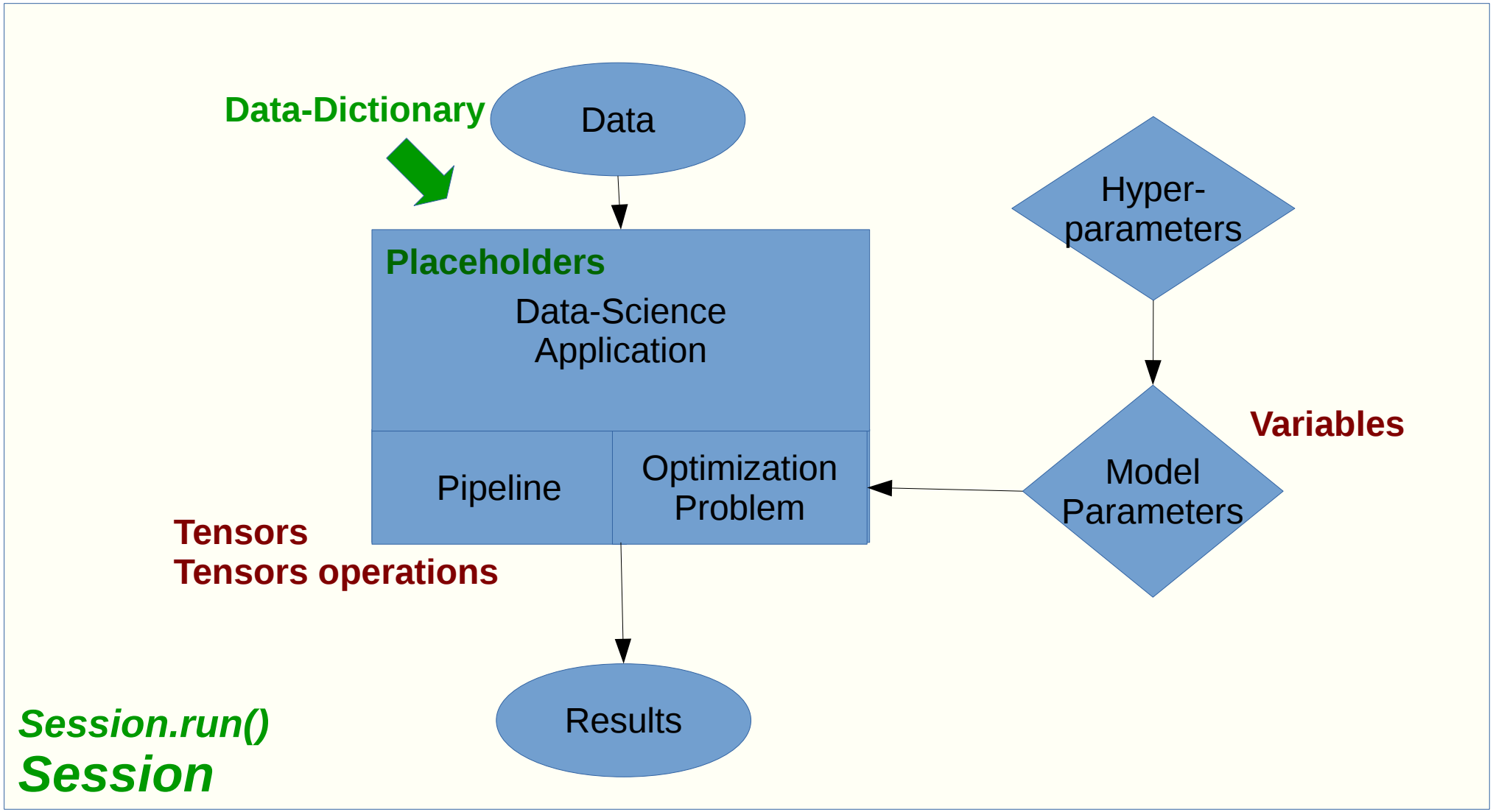

A defined application is run in a session object

# Let's do it!

Sessions + data-dictionary + placeholders

## “Hello data!” in tensorflow

# Defining processing pipelines




The application graph is where intermediary data (tensors) intertwines with model variables

# Defining variables in tensorflow

- Warnings:
  - tensorflow picky about variables types
  - Variables are attached to a session
  - Variables MUST be initialized before starting using
    - init = tf.initialize_all_variables()
    - session.run(init)

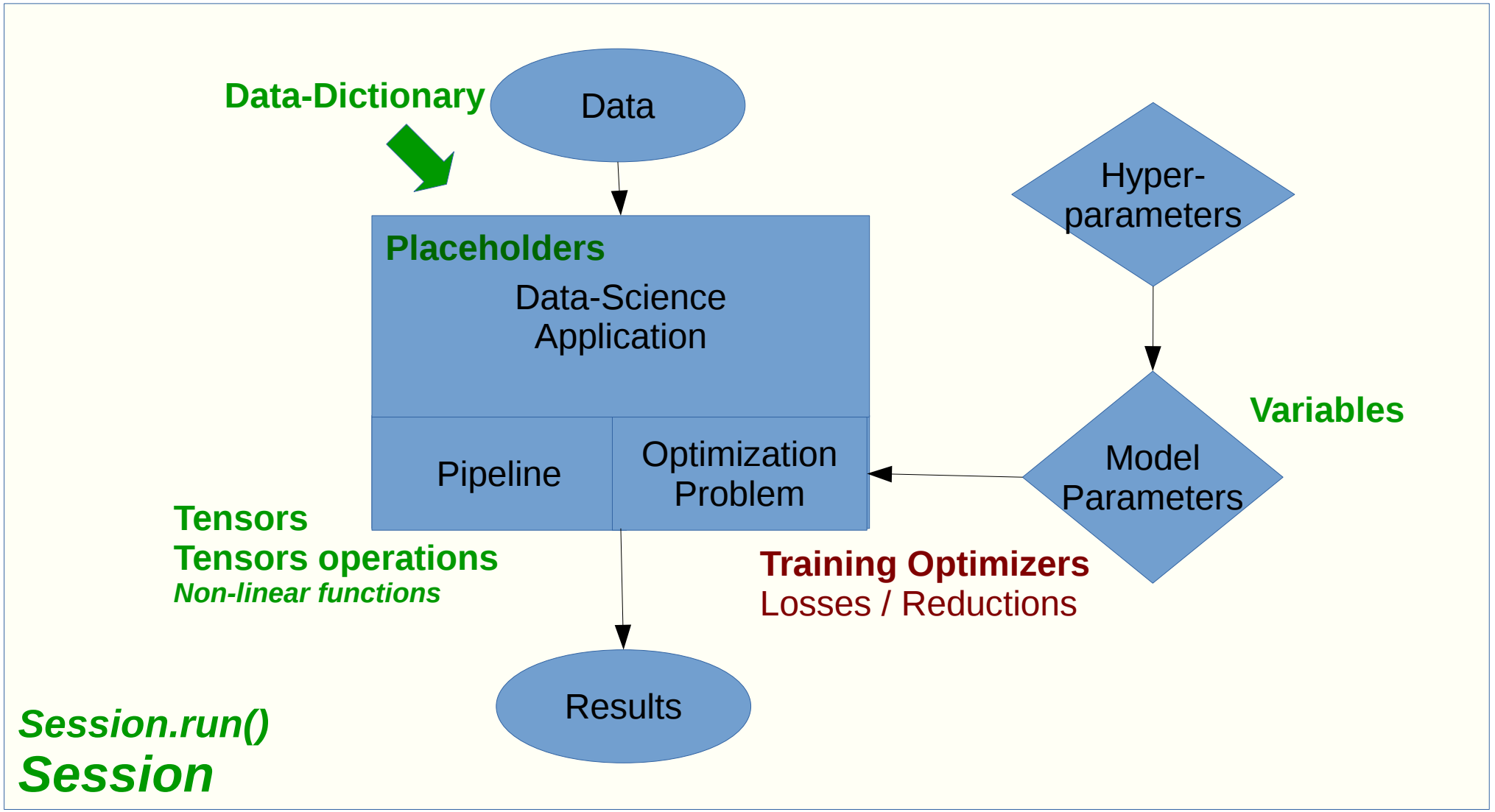
# Let's do it!

Sessions + data-dictionary + placeholders
+
Tensors + variables

## Logistic-regression in tensorflow

# Actually learning something

## Defining loss and optimizer




Defining a loss and adding a training operation enables to define an optimization problem

# Let's run it!

Sessions + data-dictionary + placeholders
&
Tensors + variables
&
Loss + optimizer

## Logistic-regression in tensorflow

# Comparison point
# Tensorflow & scikit-learn

## Tensorflow

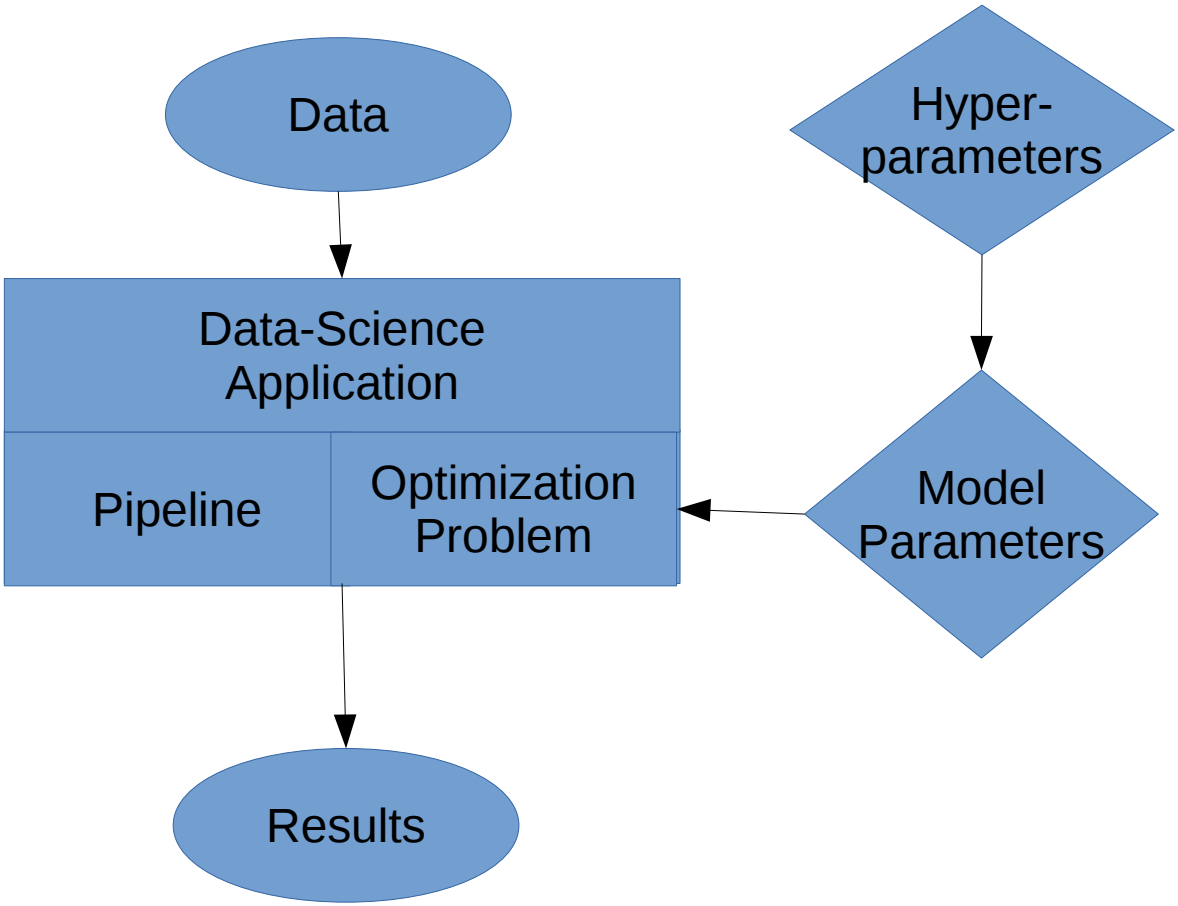
- Full fledged optimization routine
- SGD based
- Requires definition of optimization problem
- Designed to scale

## Scickit-learn

- Out-of-box ML library
- Dedicated optimizer
- Just use the *fit()* and *predict()* methods
- Designed to be user-friendly

# Phases of data-science applications

- Definition
- Training
- Scoring



# Shall I use deep-learning / tensorflow?

- Application is complex
- Data is
  - Highly dimensional
  - Plenty
- Feature space is:
  - Complex
  - Huge

***YES!***

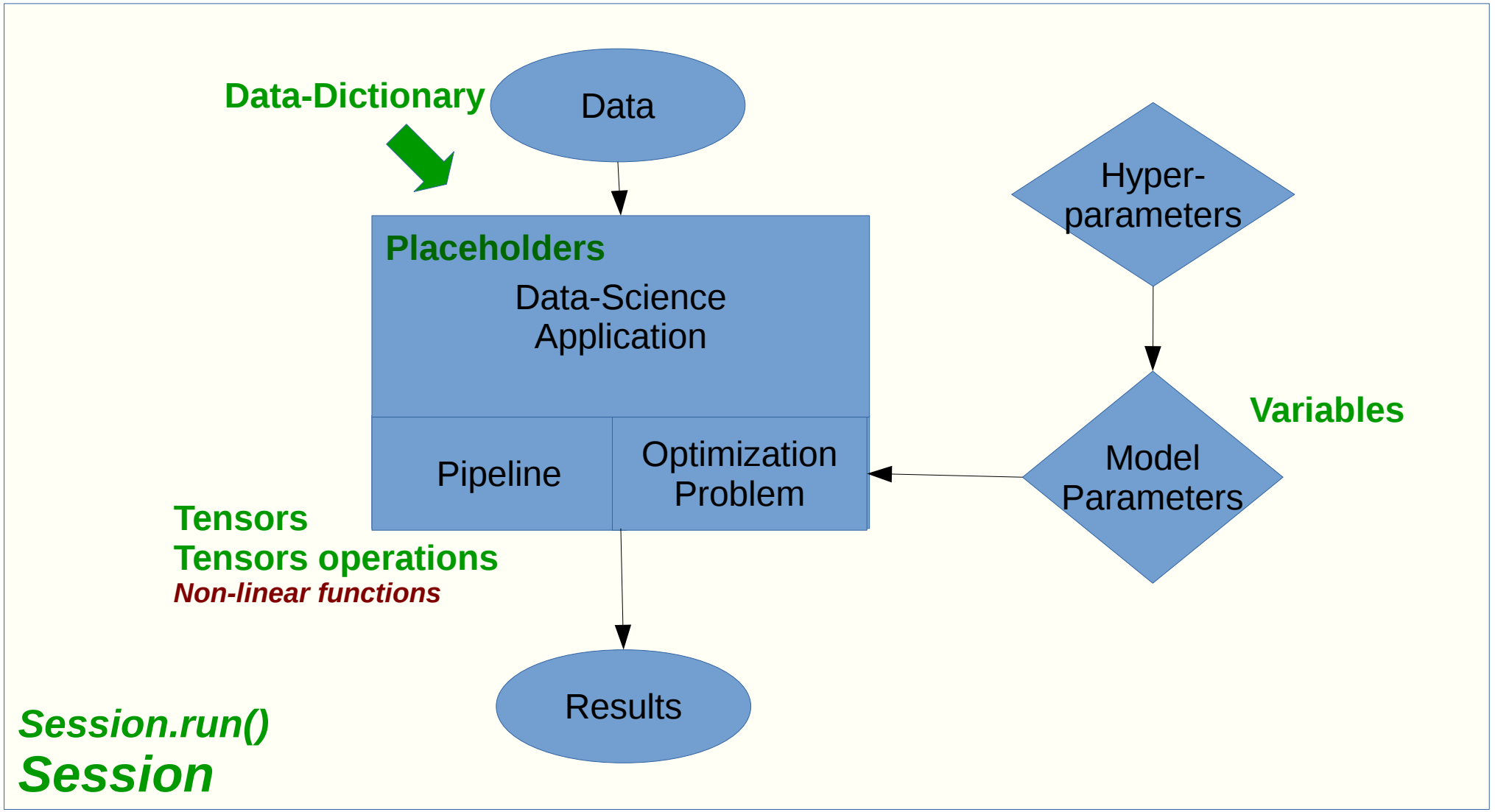
- Application is simple
- Data is
  - Concise
  - Small
- Feature space is:
  - Intuitive
  - Simple

***NO!***

Don't use a sledgehammer to kill a fly.

# Defining a neural network

Our basic lego block

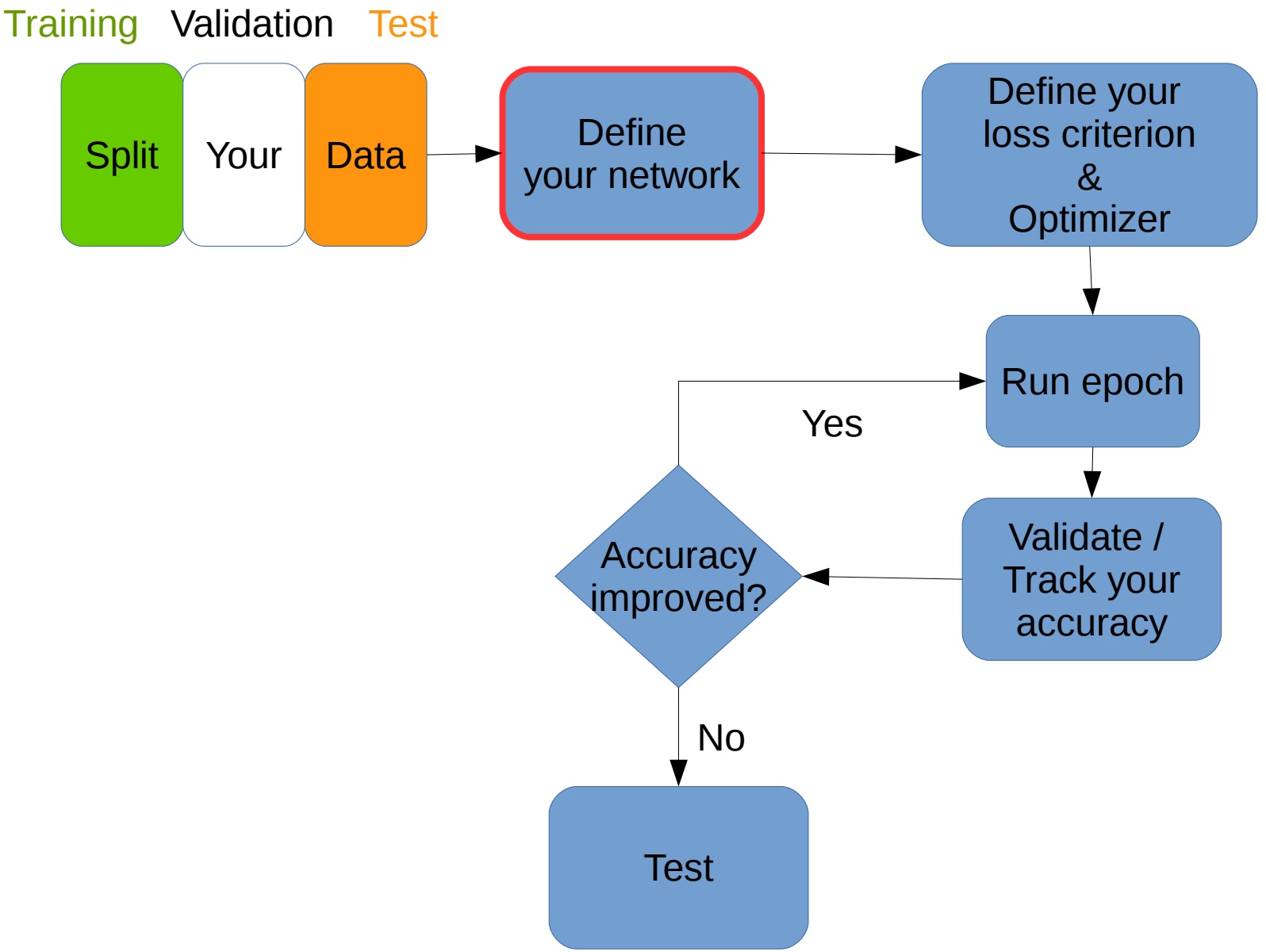


# Let's do it!

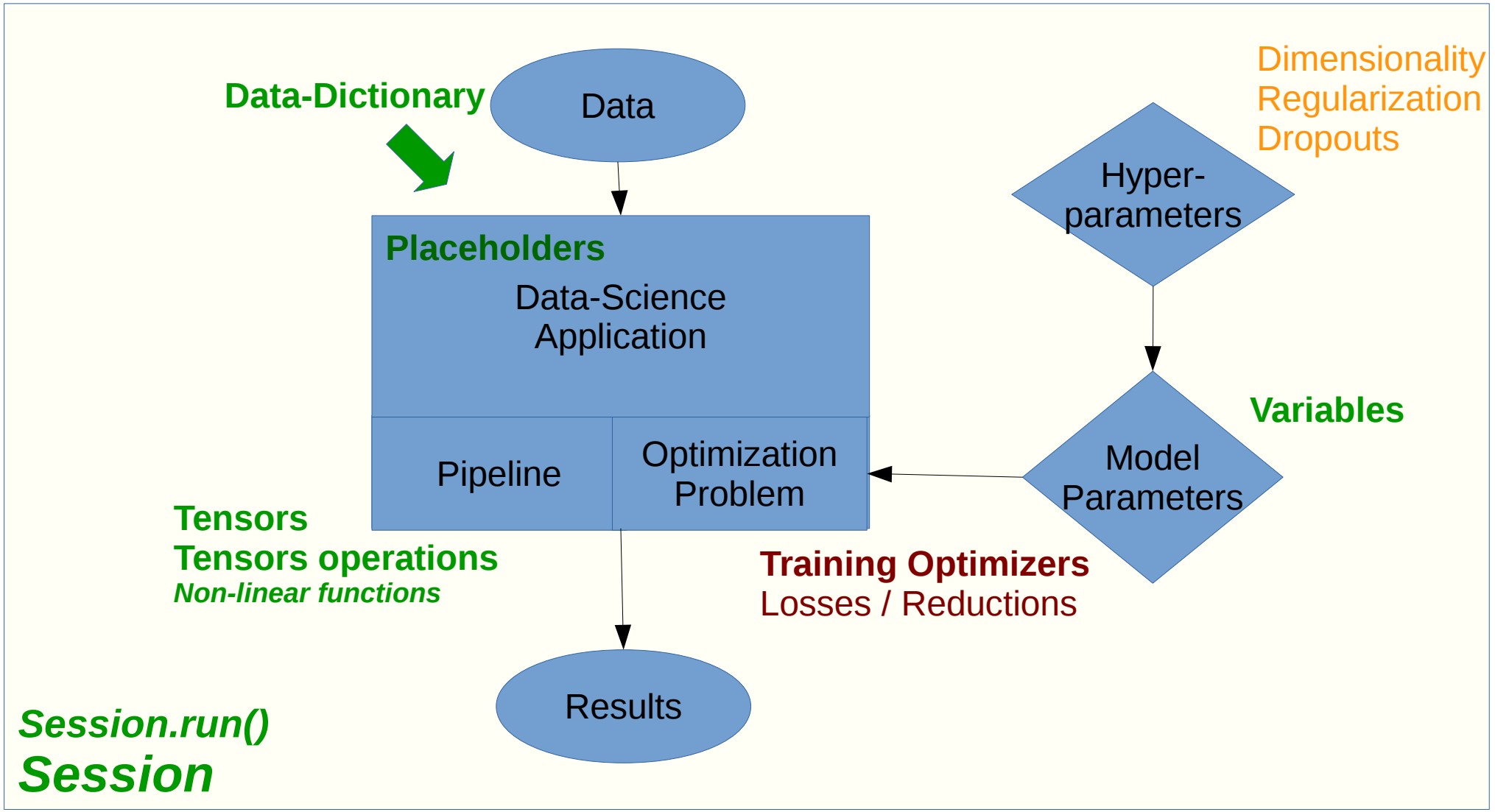
Sessions + data-dictionary + placeholders
+
Tensors + variables + NN operations

## Mnist in tensorflow

# The process of deep-learning



# Hyper-parameters tunning



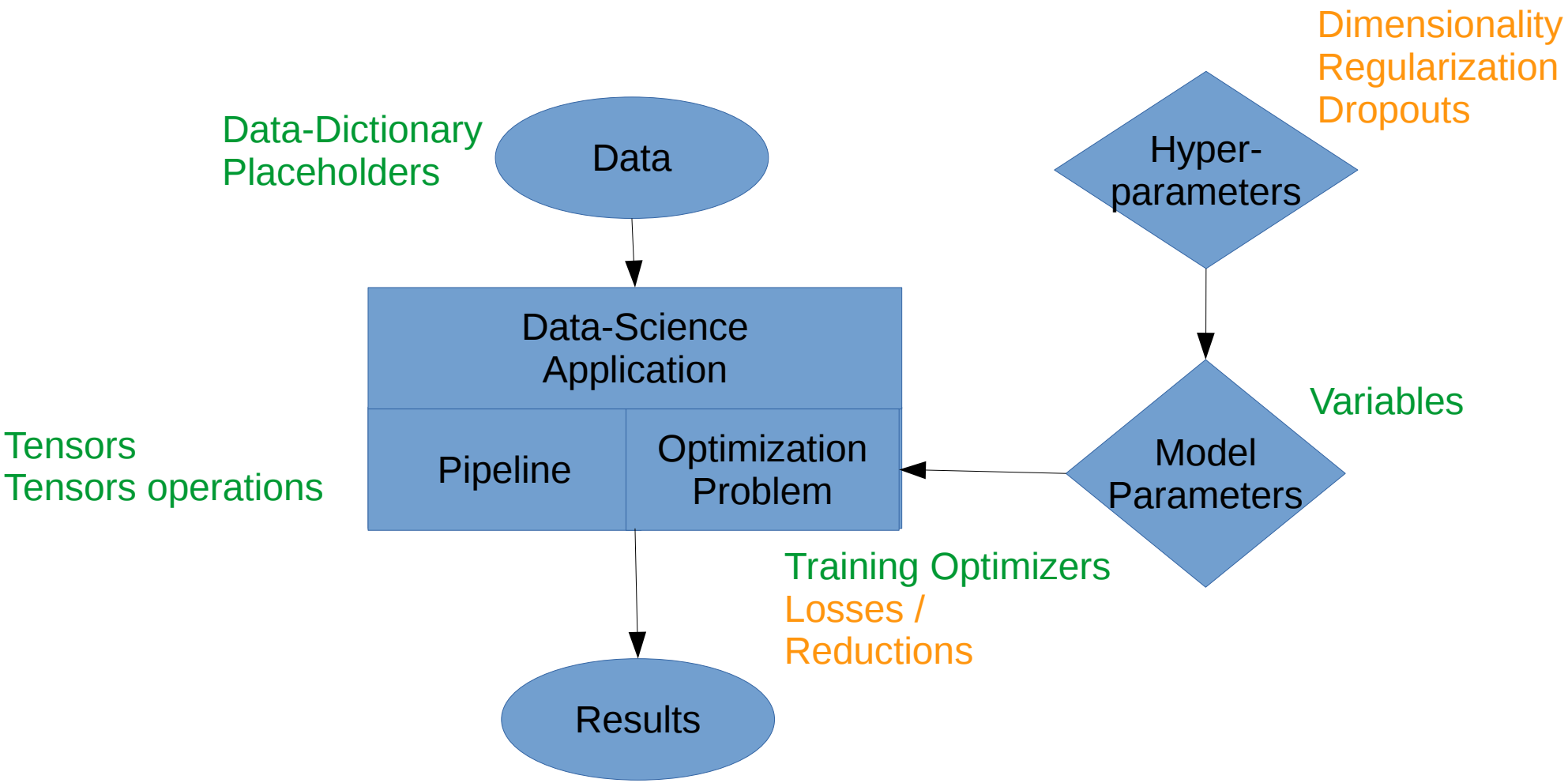

Defining a loss and adding a training operation enables to define an optimization problem

# Hyper-parameters tuning

It's a trap!

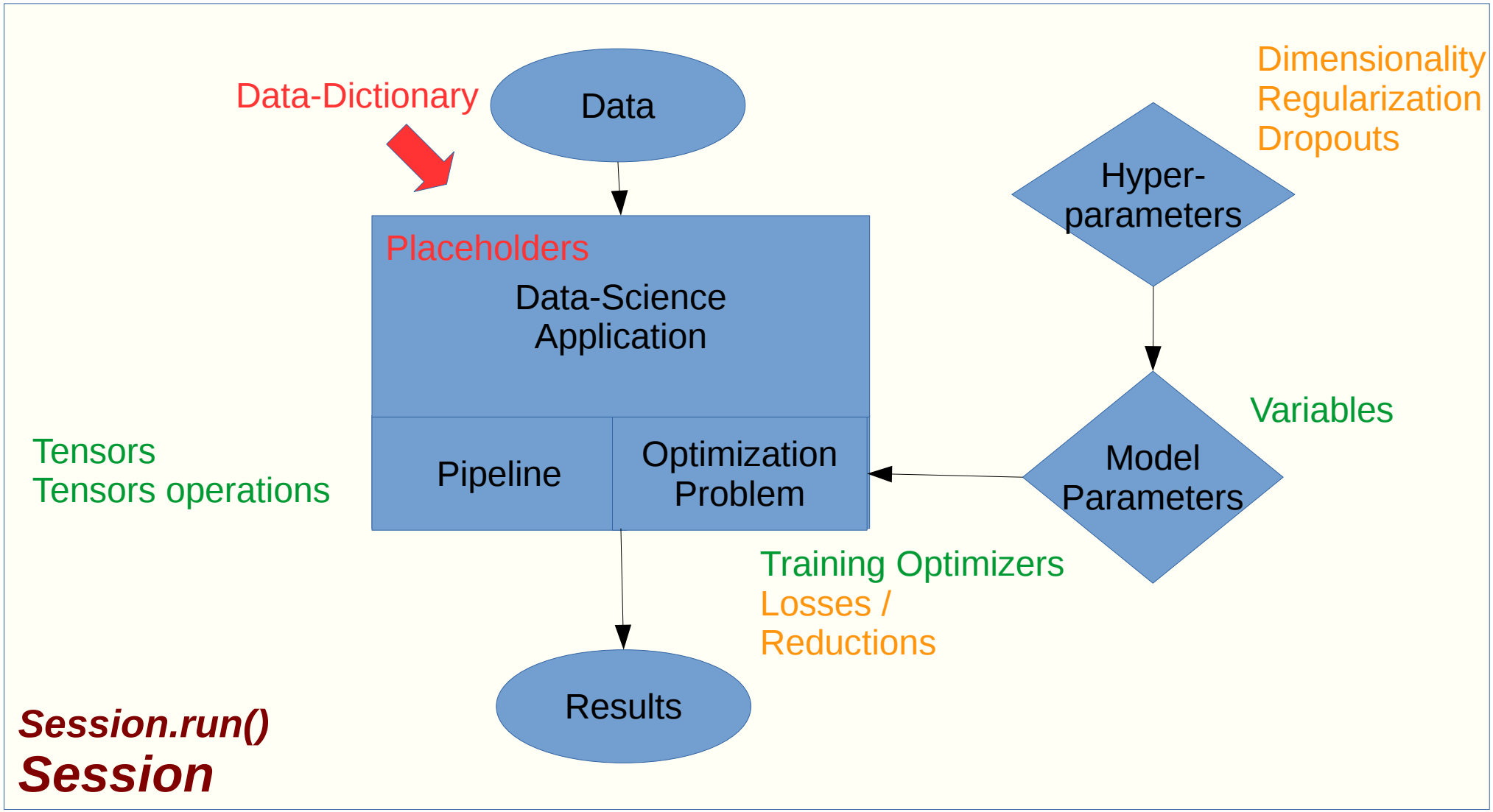
# Recommended structure of a deep-learning application

# Ingredients of data-science applications



# Outputting data in tensor flow

Session.run() returns target tensors



# Why would google open-source TensorFlow?

One Framework to rule them all, One Framework to find them,
One Framework to bring them all and in deep-learning bind them

End of presentation