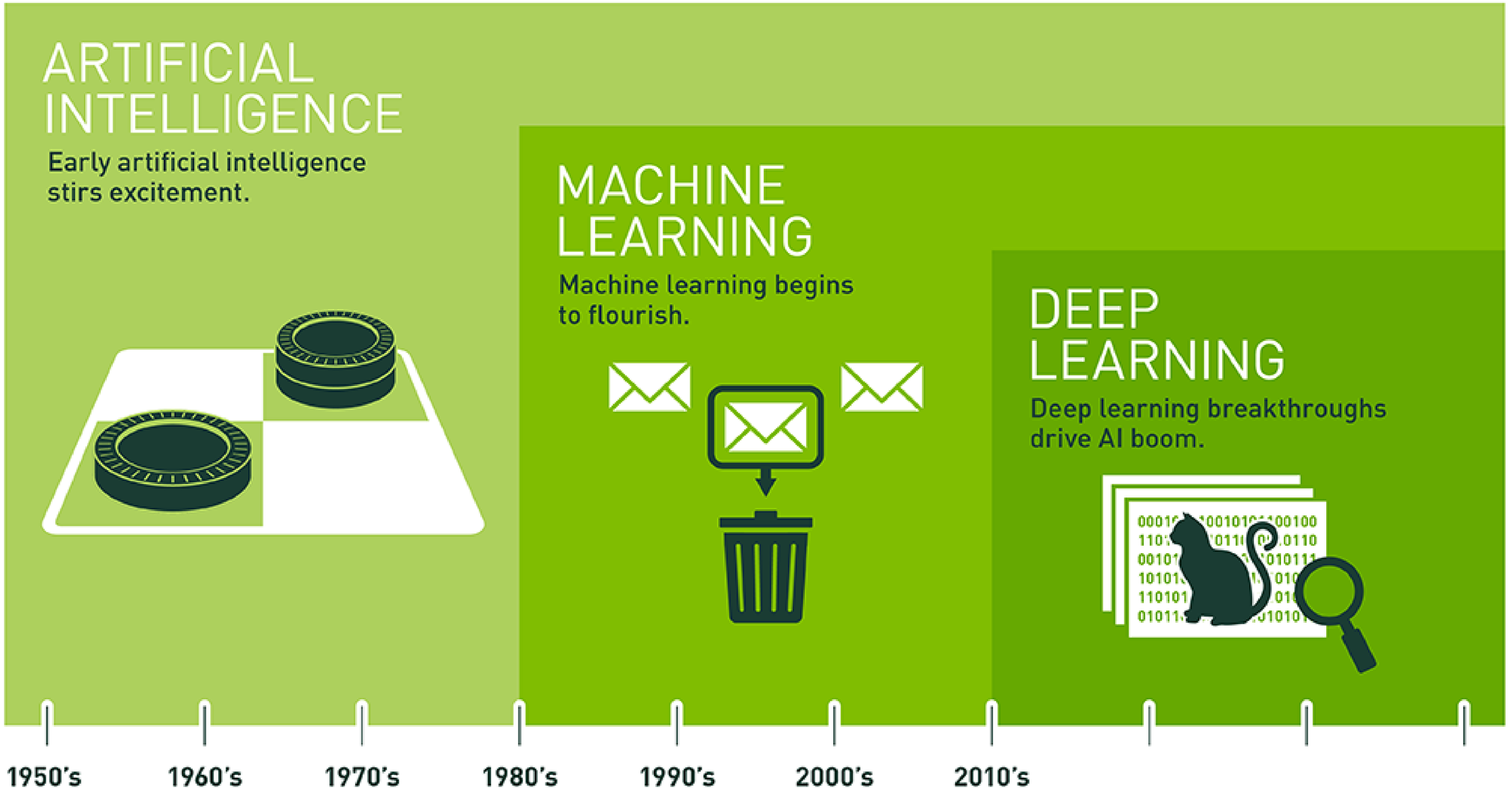
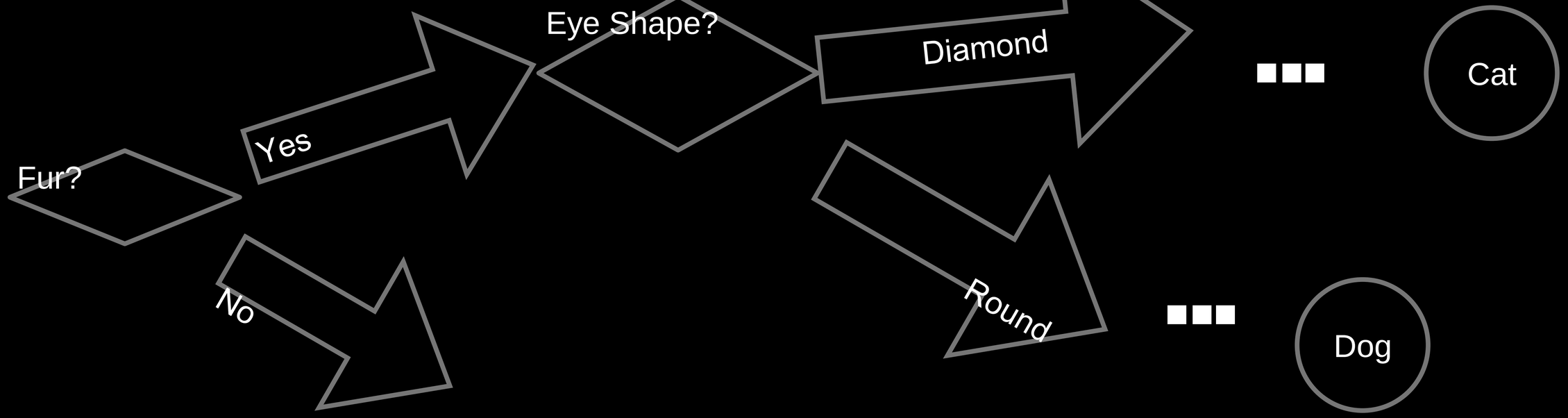
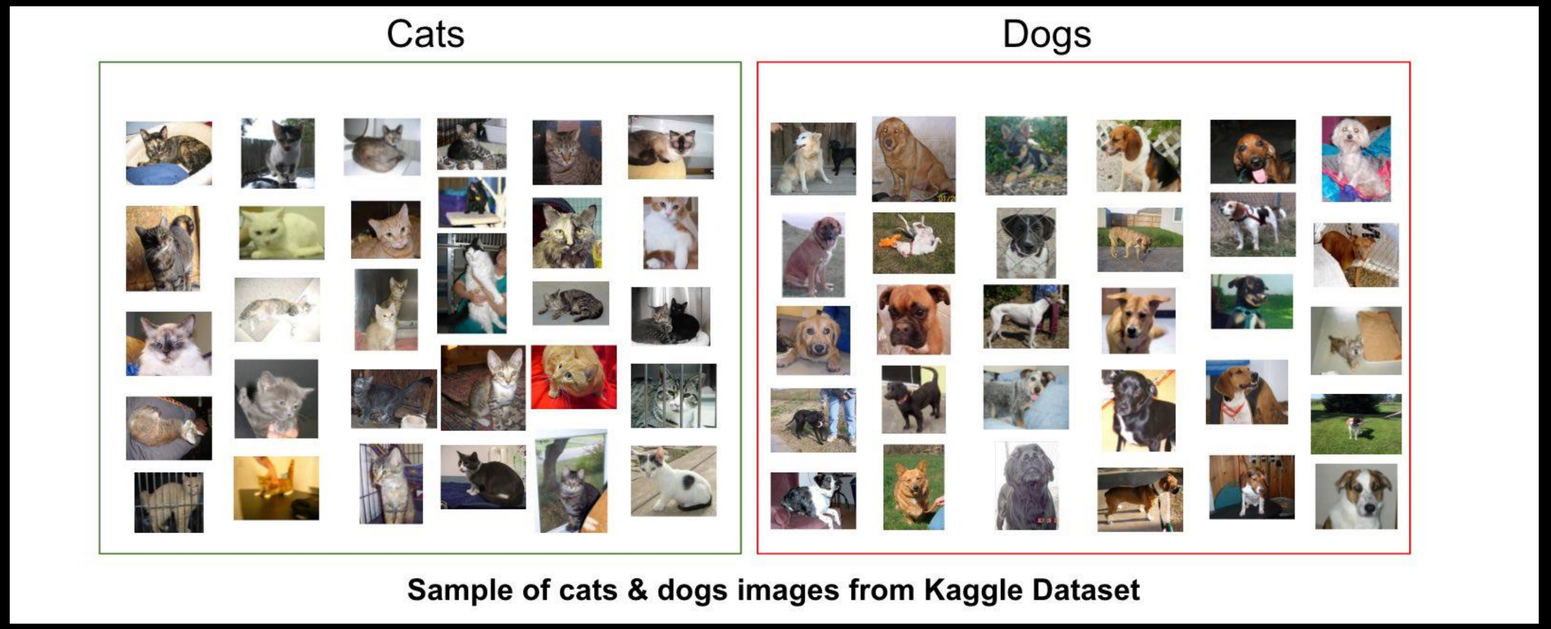
A Inteligência Artificial encontra a Meteorologia

Igor Oliveira
IBM Research



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.



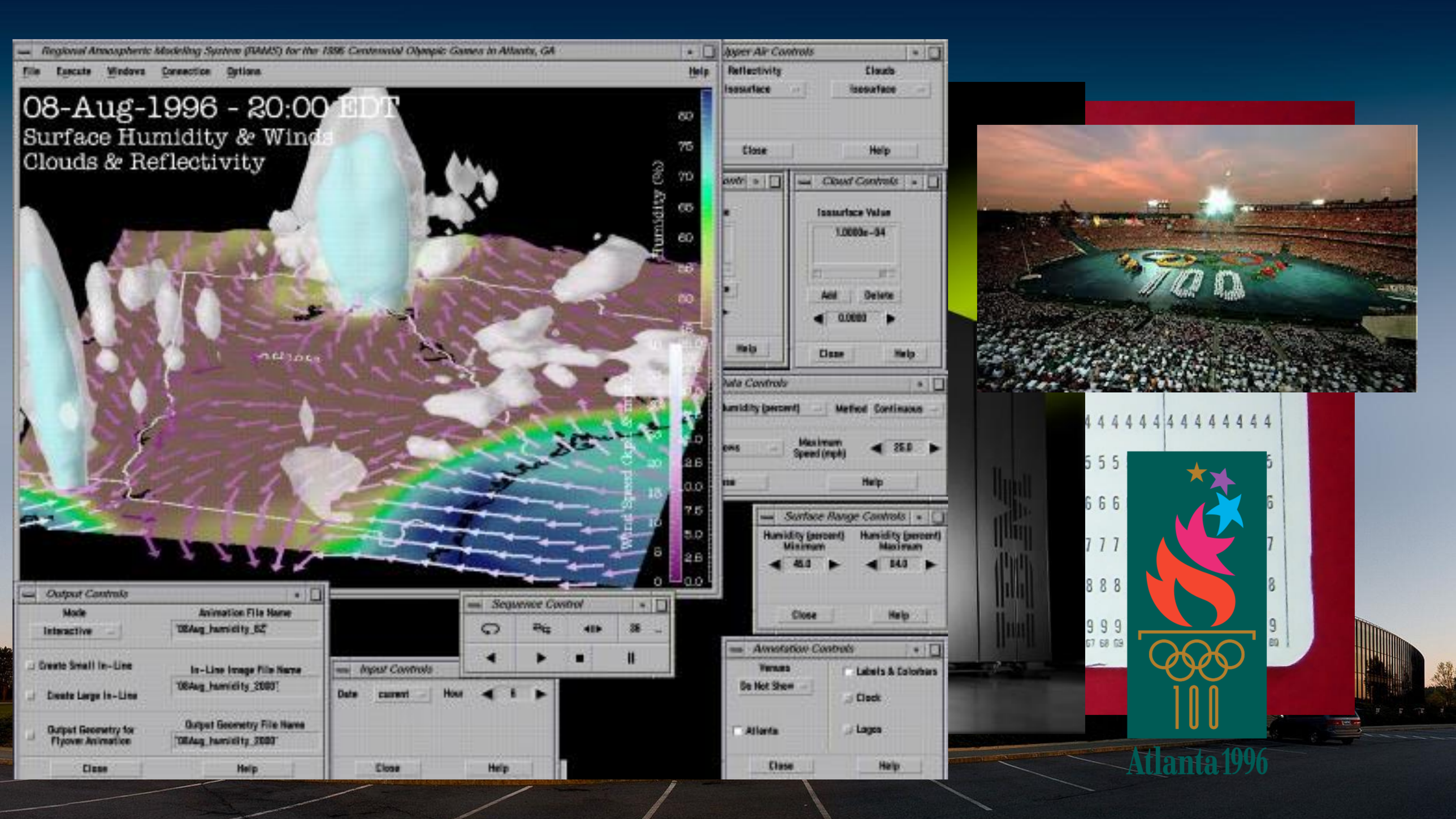


Sample of cats & dogs images from Kaggle Dataset

?

?


IBM
Tabulating
Systems Era
1900 – 1940s
Programmable
Systems Era
1950s – Present
Cognitive
Computing Era
2011 –
#CognitiveEra

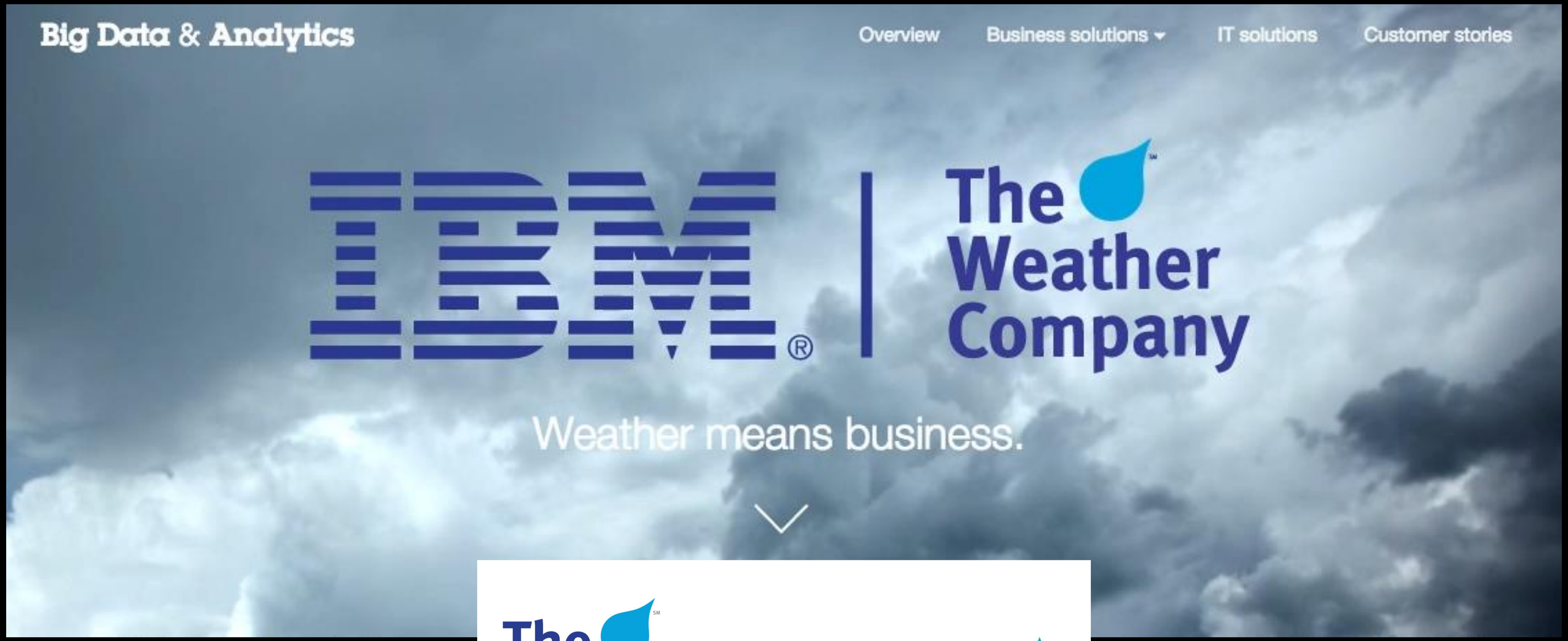

Regional Atmospheric Modeling System (RAMS) for the 1996 Centennial Olympic Games in Atlanta, GA
File Execute Windows Connection Options Help
08-Aug-1996 - 20:00 EDT
Surface Humidity & Winds
Clouds & Reflectivity
Humidity (%)
Wind Speed (kts)
Upper Air Controls
Reflectivity
Clouds
Isosurface
Close Help
Cloud Controls
Isosurface Value
Add Delete
Close Help
Data Controls
Method Continuous
Maximum Speed (mph)
Surface Range Controls
Humidity (percent) Minimum
Humidity (percent) Maximum
Output Controls
Mode
Interactive
Animation File Name
Create Small In-Line
Create Large In-Line
In-Line Image File Name
Output Geometry for Flyover Animation
Output Geometry File Name
Sequence Control
Input Controls
Date current
Hour
Annotation Controls
Versus
Do Not Show
Labels & Colorbars
Clock
Atlanta
Logos
Close Help
Atlanta 1996

Kasparov

1997

Watson Jeopardy

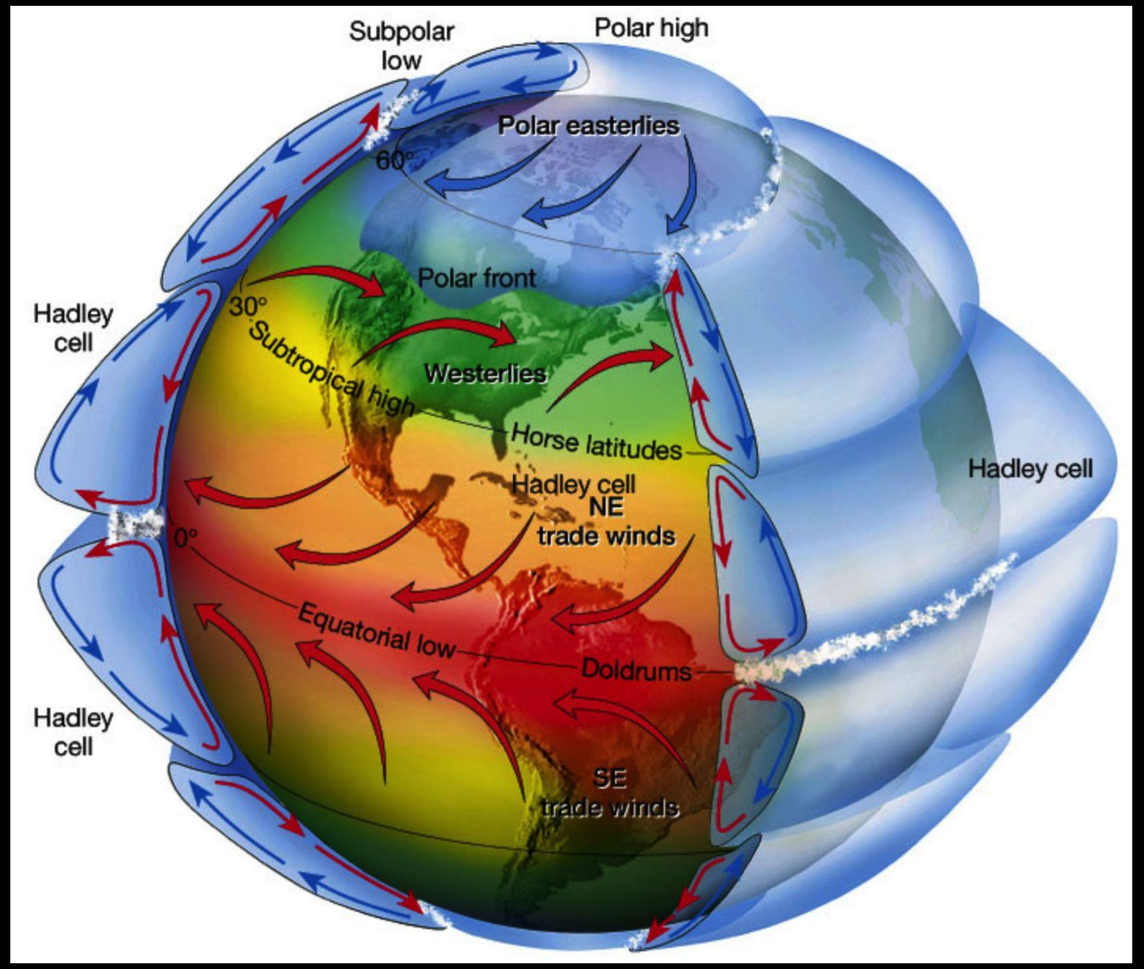
2011

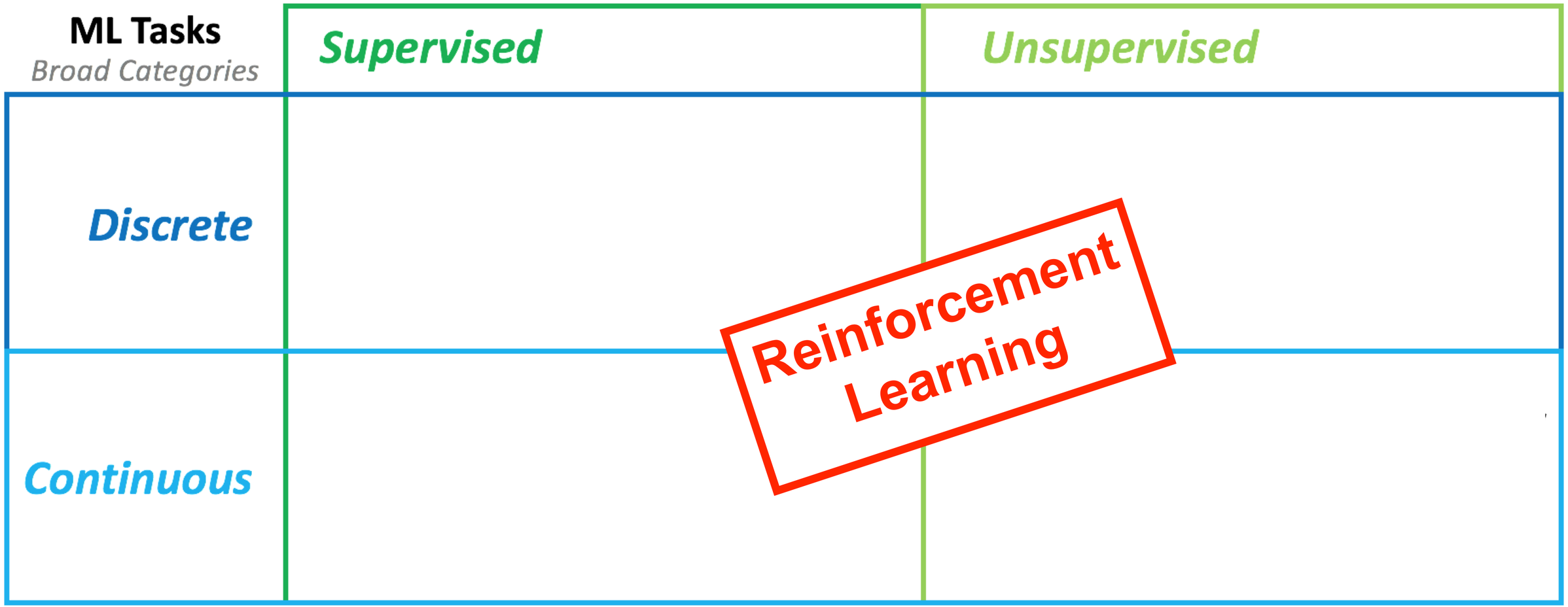


The Weather Company
An IBM Business

The Weather Channel

WEATHER UNDERGROUND

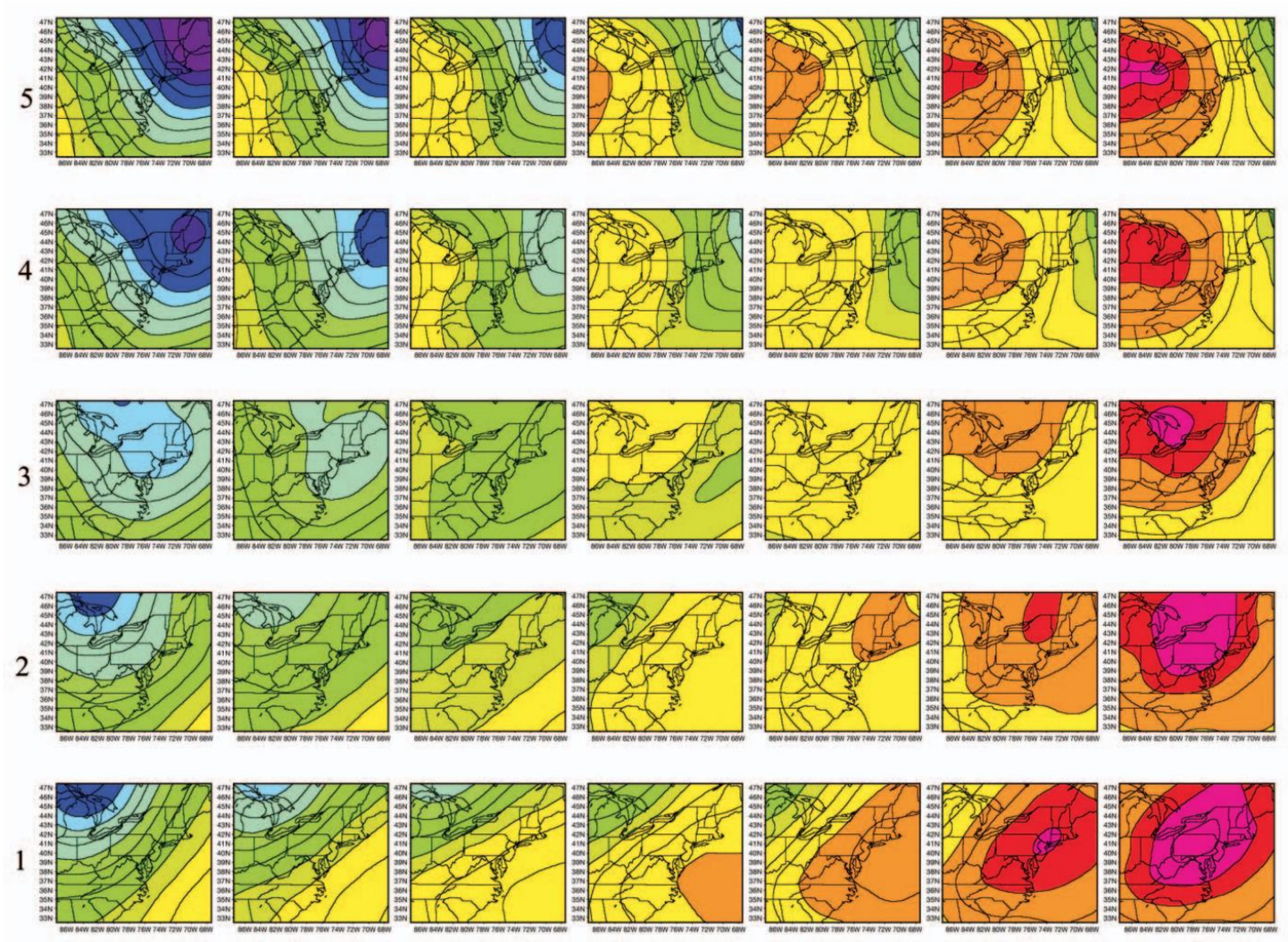

Subpolar low
Polar high
Polar easterlies
60°
Polar front
Hadley cell
30°
Subtropical high
Westerlies
Horse latitudes
Hadley cell
NE trade winds
Hadley cell
0°
Equatorial low
Doldrums
Hadley cell
SE trade winds


ML Tasks
Broad Categories
Supervised
Unsupervised
Discrete
Continuous
Reinforcement Learning

Starting out - 10 minutes of training

The algorithm tries to hit the ball back, but
it is yet too clumsy to manage.

Unsupervised Methods



Supervised Methods

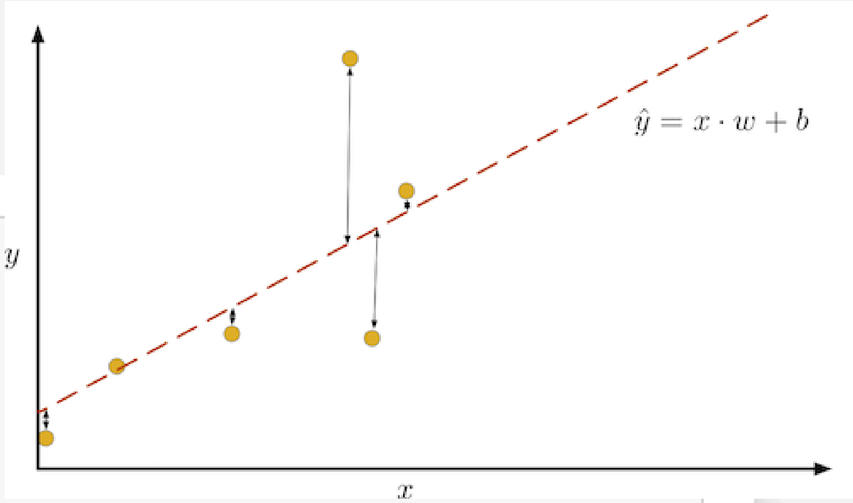
$$y = A * x$$

Labels
Target

Operator

Predictor
Features
Inputs

Boston Housing Dataset



In [1]:
```
from sklearn.datasets import load_boston
import pandas as pd

boston = load_boston()
df = pd.DataFrame(boston.data, columns=boston.feature_names)
df['Value']=boston.target
df.head()
```

Out[1]:

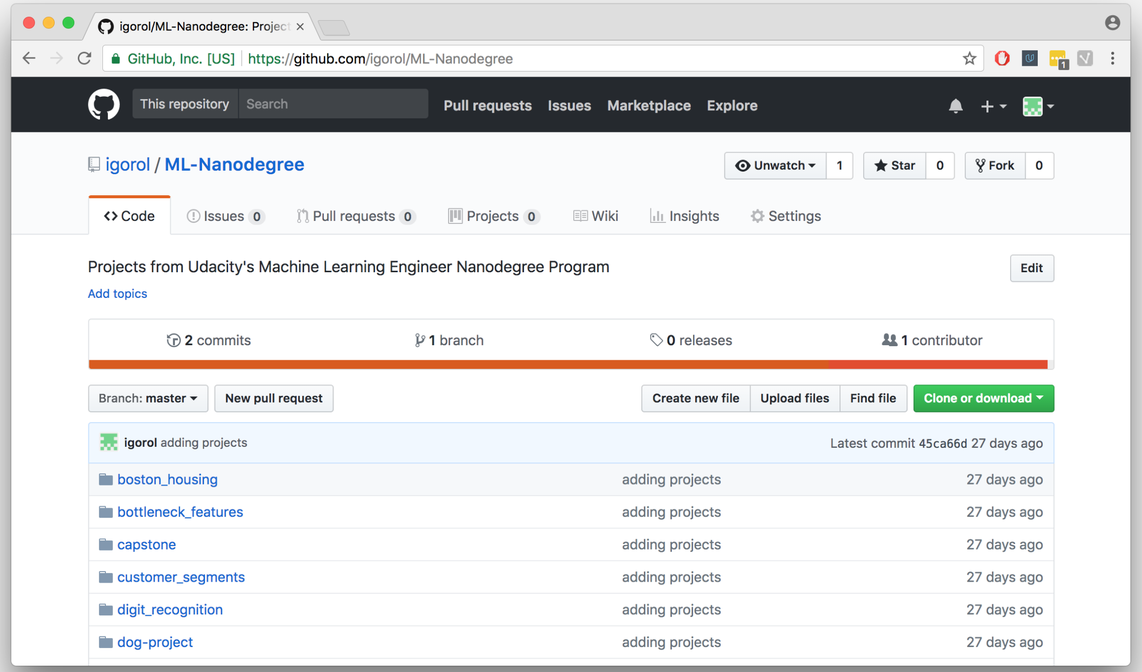
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Value
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

Value = ß1* LSTAT + ß2* B + ß3* TAX + ß4* LSTAT + ß5* RAD + ….. + E

Value = ß* LSTAT + E

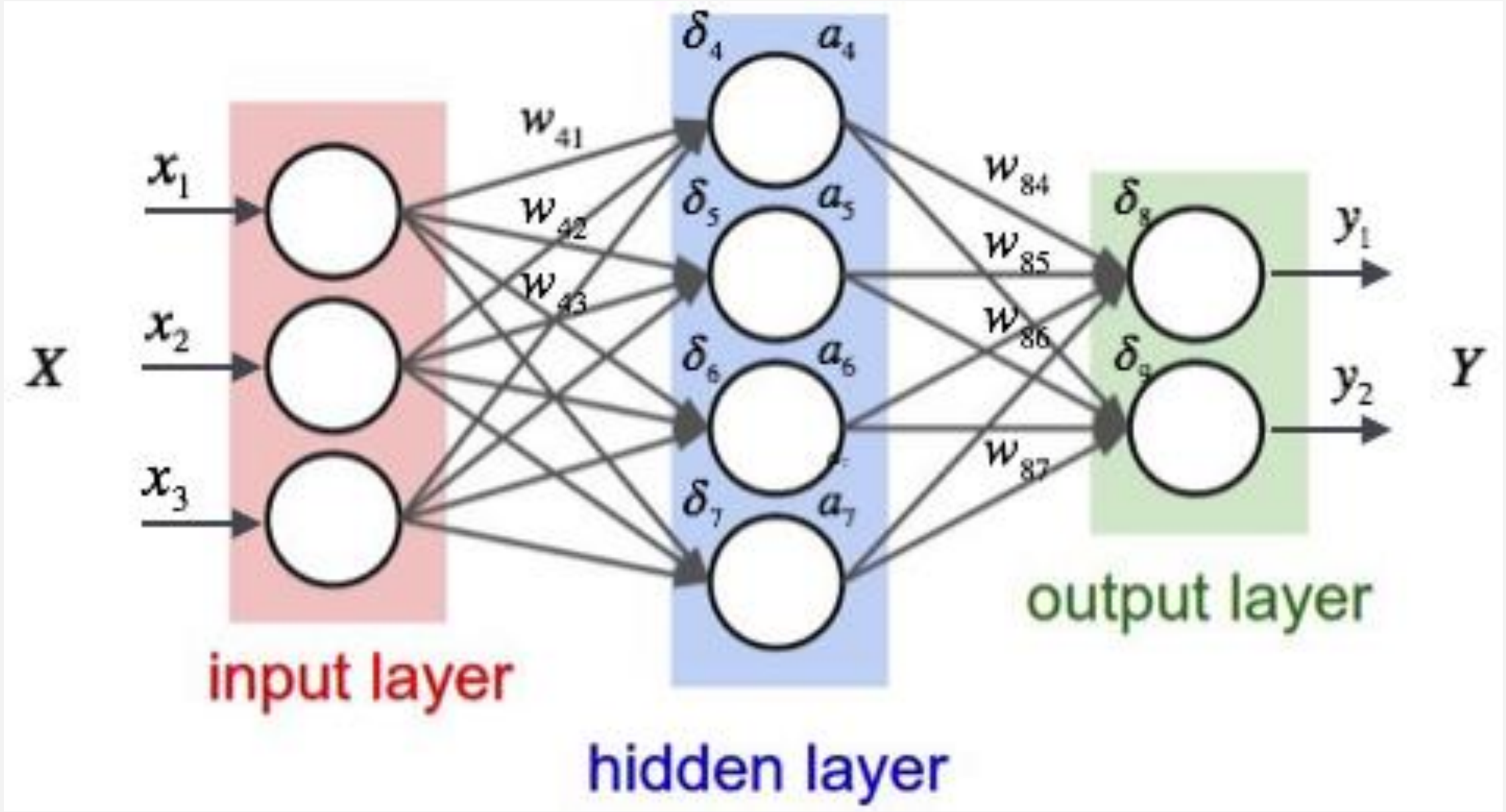
WX	TEMP(T-3)	RELH(T-3)	UWIND(T-3)	VWIND(T-3)
FG	12	99	1	2
CAVOK	23	40	4	5
CAVOK	20	60	4	6
CAVOK	28	50	3	2

https://github.com/igorol/ML-Nanodegree

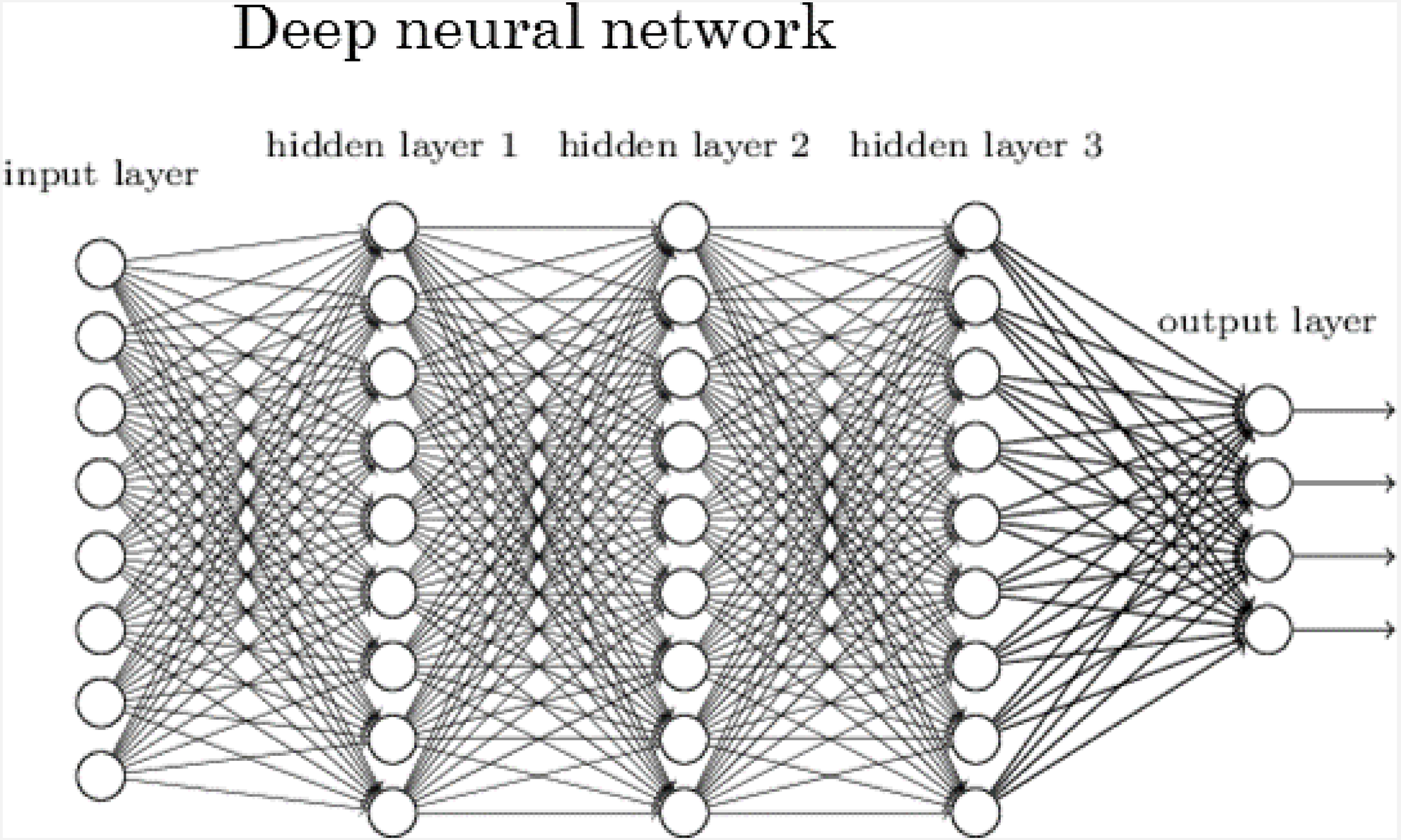


Redes Neurais

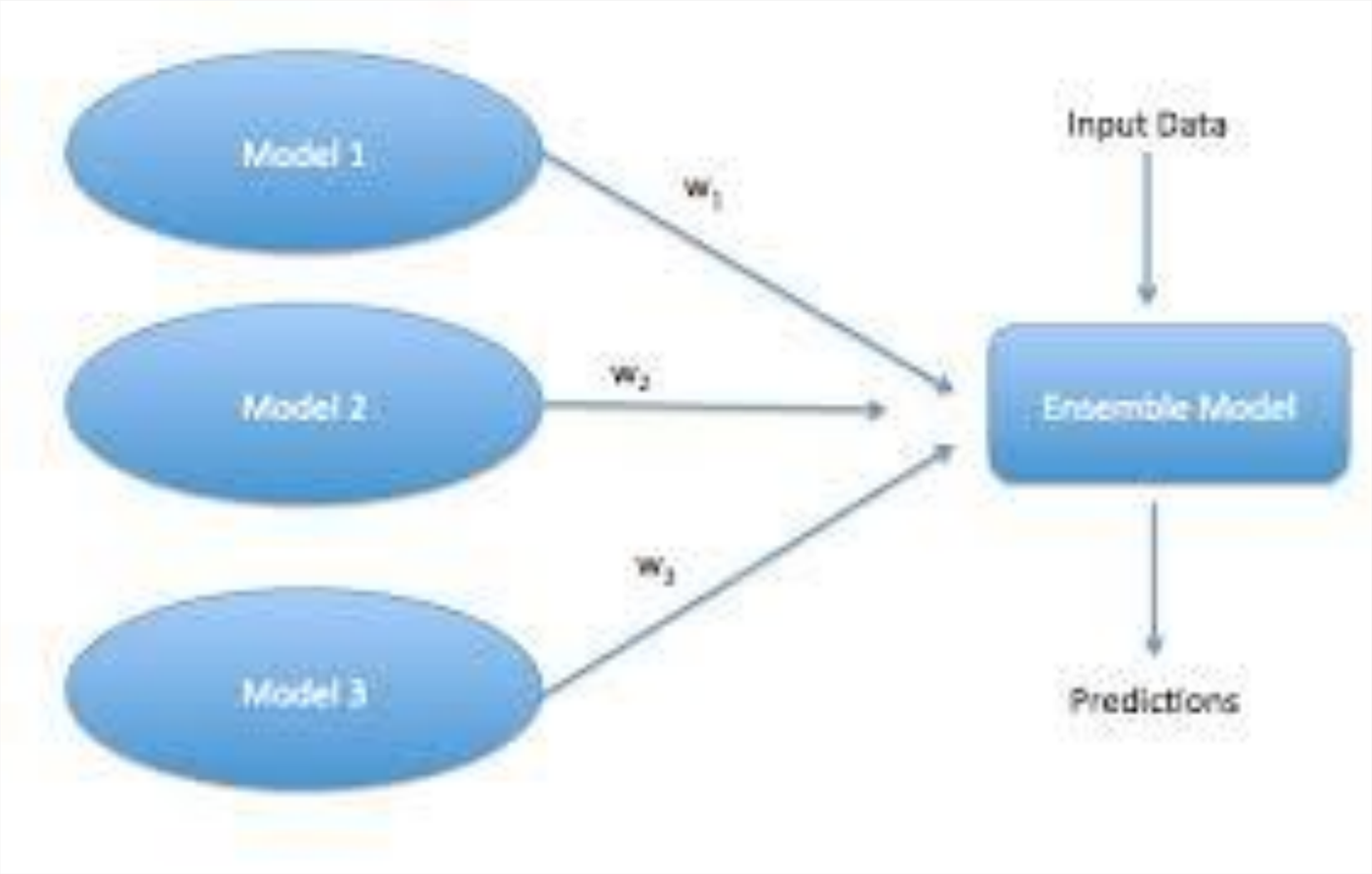
Smallest biocomputer existing

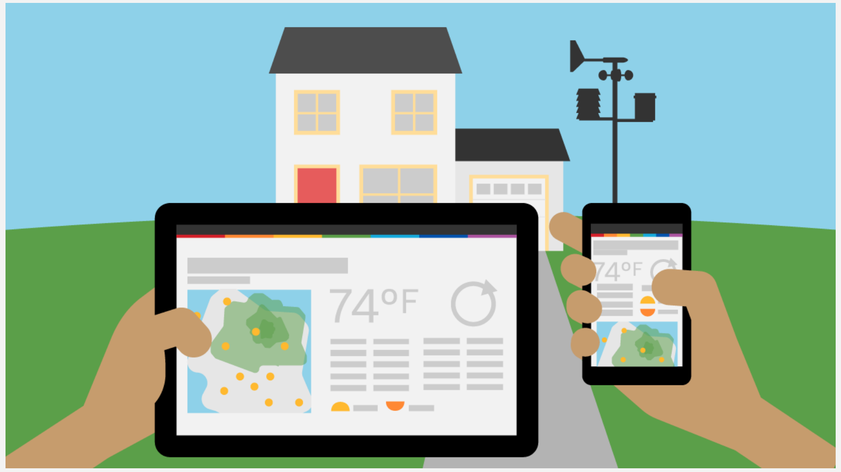

X
x_1
x_2
x_3
w_{41}
w_{42}
w_{43}
δ_4 a_4
δ_5 a_5
δ_6 a_6
δ_7 a_7
w_{84}
w_{85}
w_{86}
w_{87}
δ_8
δ_9
y_1
y_2
Y
input layer
hidden layer
output layer

Deep neural network

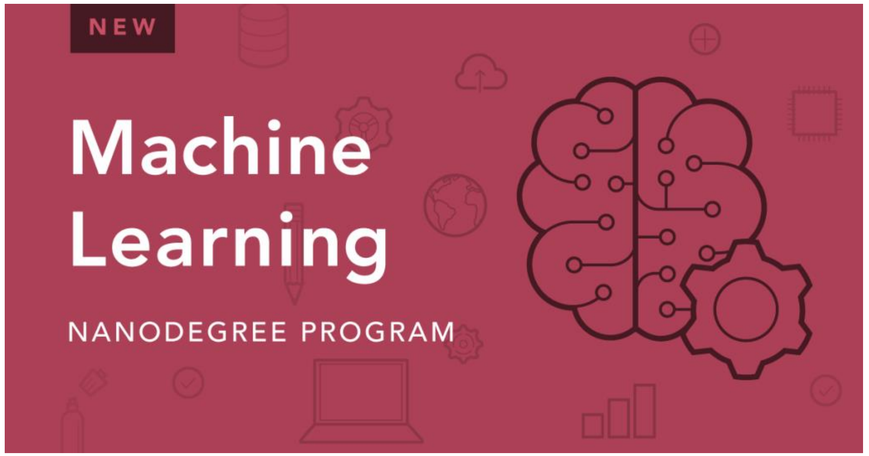


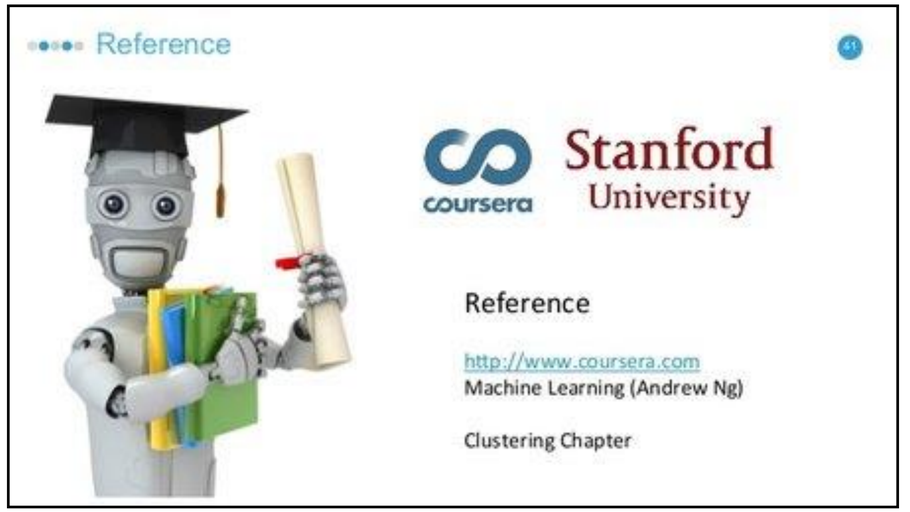
TWC Model Blending

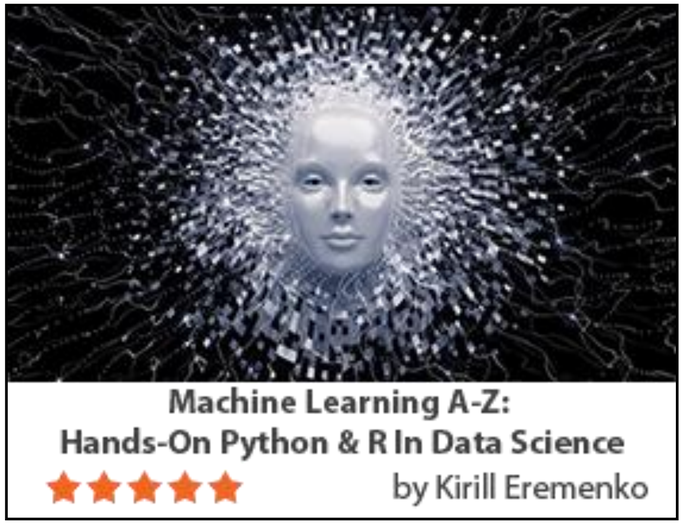




Resources


NEW
Machine Learning
NANODEGREE PROGRAM


Reference
coursera
Stanford University
Reference
http://www.coursera.com
Machine Learning (Andrew Ng)
Clustering Chapter

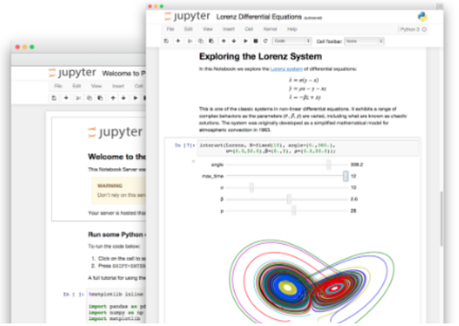

Machine Learning A-Z:
Hands-On Python & R In Data Science
by Kirill Eremenko


fast.ai
Making neural nets
uncool again

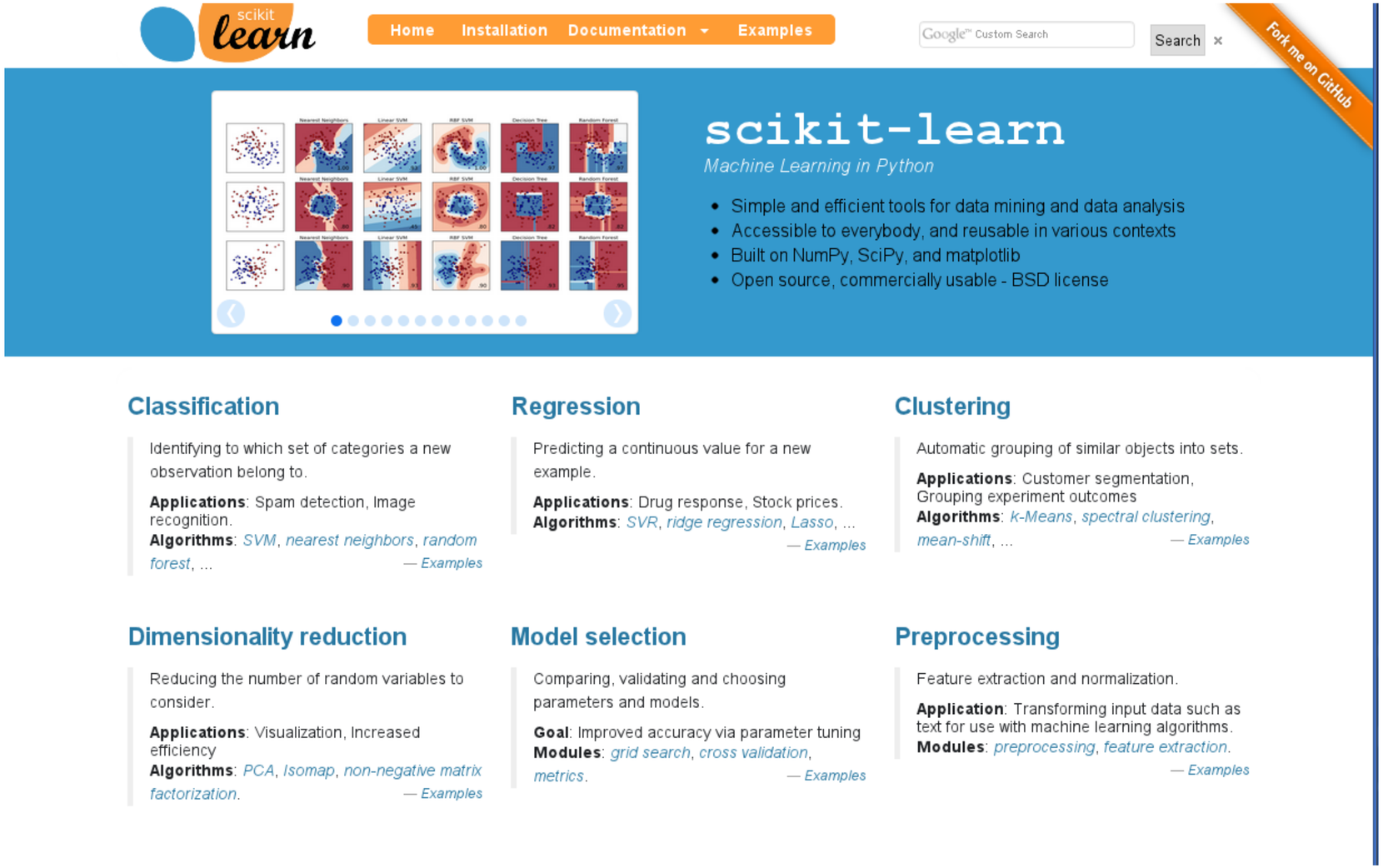
Resources

Jupyter Notebook

The Jupyter Notebook is a web-based interactive computing platform that allows users to author data- and code-driven narratives that combine live code, equations, narrative text, visualizations, interactive dashboards and other media.



Resources


scikit learn
Home
Installation
Documentation
Examples
Google™ Custom Search
Search
Fork me on GitHub
scikit-learn
Machine Learning in Python
Simple and efficient tools for data mining and data analysis
Accessible to everybody, and reusable in various contexts
Built on NumPy, SciPy, and matplotlib
Open source, commercially usable - BSD license
Classification
Identifying to which set of categories a new observation belong to.
Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, ...
— Examples
Regression
Predicting a continuous value for a new example.
Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, ...
— Examples
Clustering
Automatic grouping of similar objects into sets.
Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, ...
— Examples
Dimensionality reduction
Reducing the number of random variables to consider.
Applications: Visualization, Increased efficiency
Algorithms: PCA, Isomap, non-negative matrix factorization.
— Examples
Model selection
Comparing, validating and choosing parameters and models.
Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics.
— Examples
Preprocessing
Feature extraction and normalization.
Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction.
— Examples

Resources

Here's the `Sequential` model:

```
from keras.models import Sequential

model = Sequential()
```

Stacking layers is as easy as `.add()`:

```
from keras.layers import Dense, Activation

model.add(Dense(output_dim=64, input_dim=100))
model.add(Activation("relu"))
model.add(Dense(output_dim=10))
model.add(Activation("softmax"))
```

Once your model looks good, configure its learning process with `.compile()`:

```
model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
```

If you need to, you can further configure your optimizer. A core principle of Keras is to make things reasonably simple, while allowing the user to be fully in control when they need to (the ultimate control being the easy extensibility of the source code).

Resources

Deep Learning with PyTorch

Resources