Michal Kurka

Megan Kurka

Agenda

Today's Talk

H2O Overview

- What is H2O?
- H2O in R

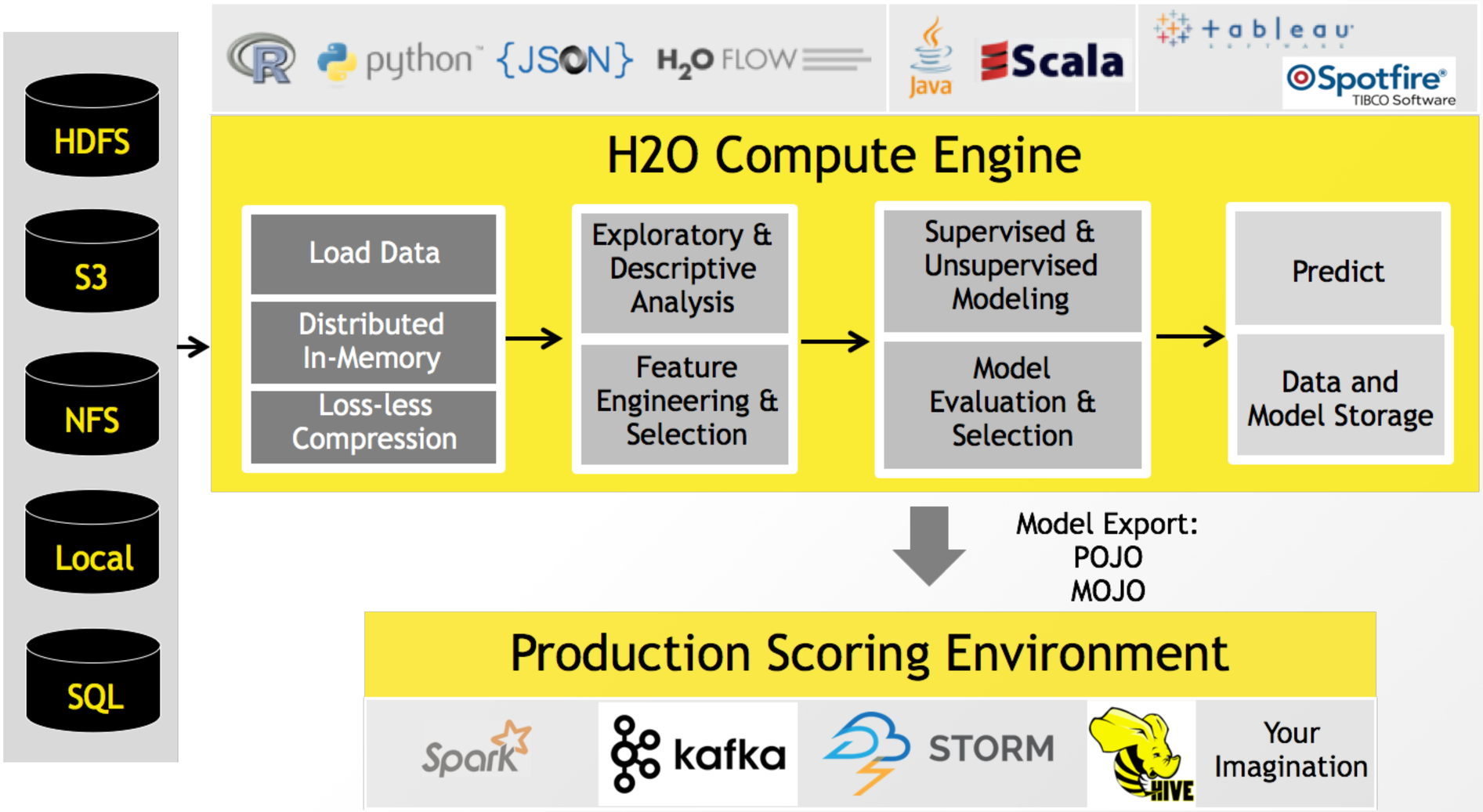
Supervised Learning

- Introduction to Machine Learning
- Overview of Supervised Learning Algorithms
- Training Models in R and Flow
- Hyper-Parameter Tuning

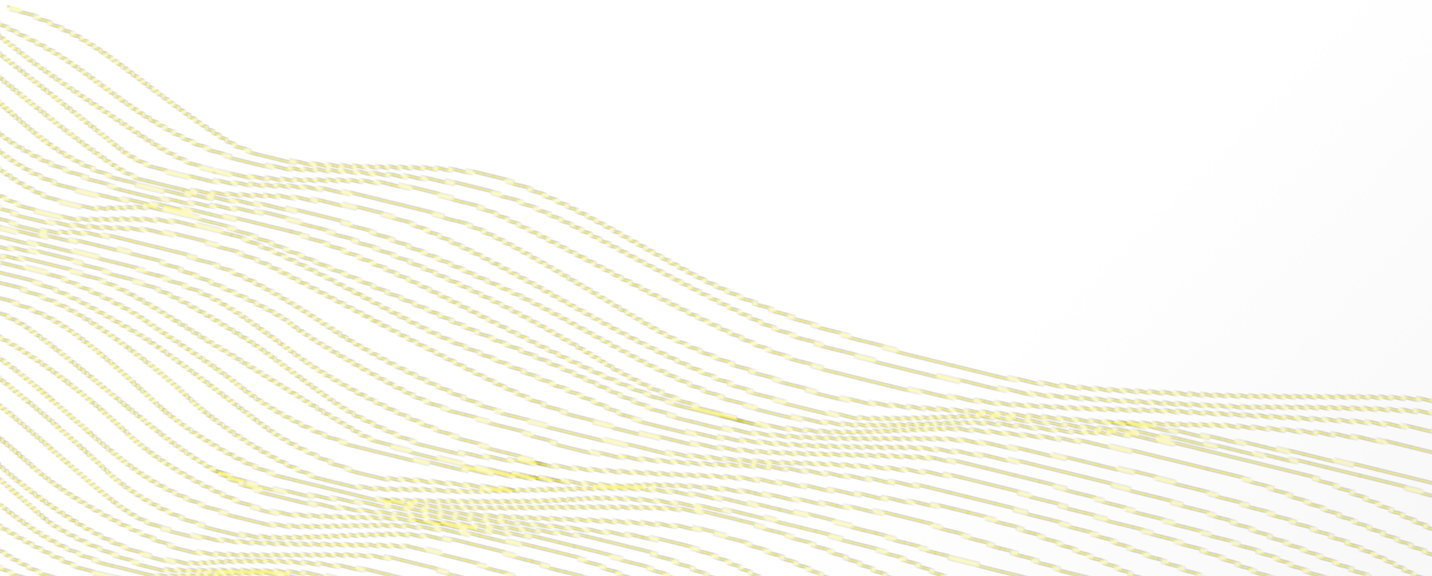
Natural Language Processing

- Word2Vec Algorithm
- Converting Text to Vectors: An Example

High Level Architecture



H_2O in R



Using H_2O with R

- Rest API drives H2O from R
- All computations are performed in performance-optimized Java code in the H2O cluster

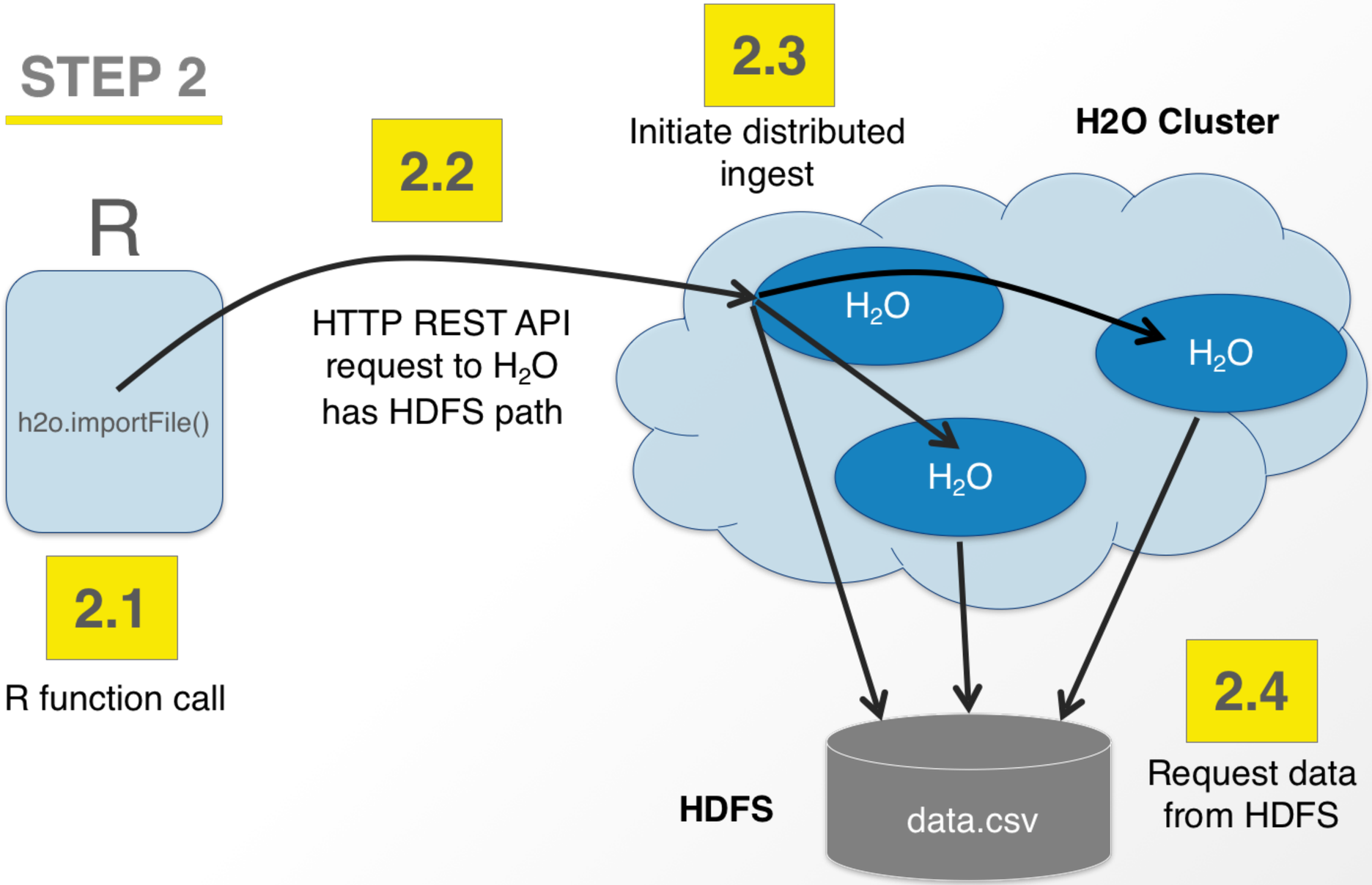
Reading Data into H2O with R

STEP 1

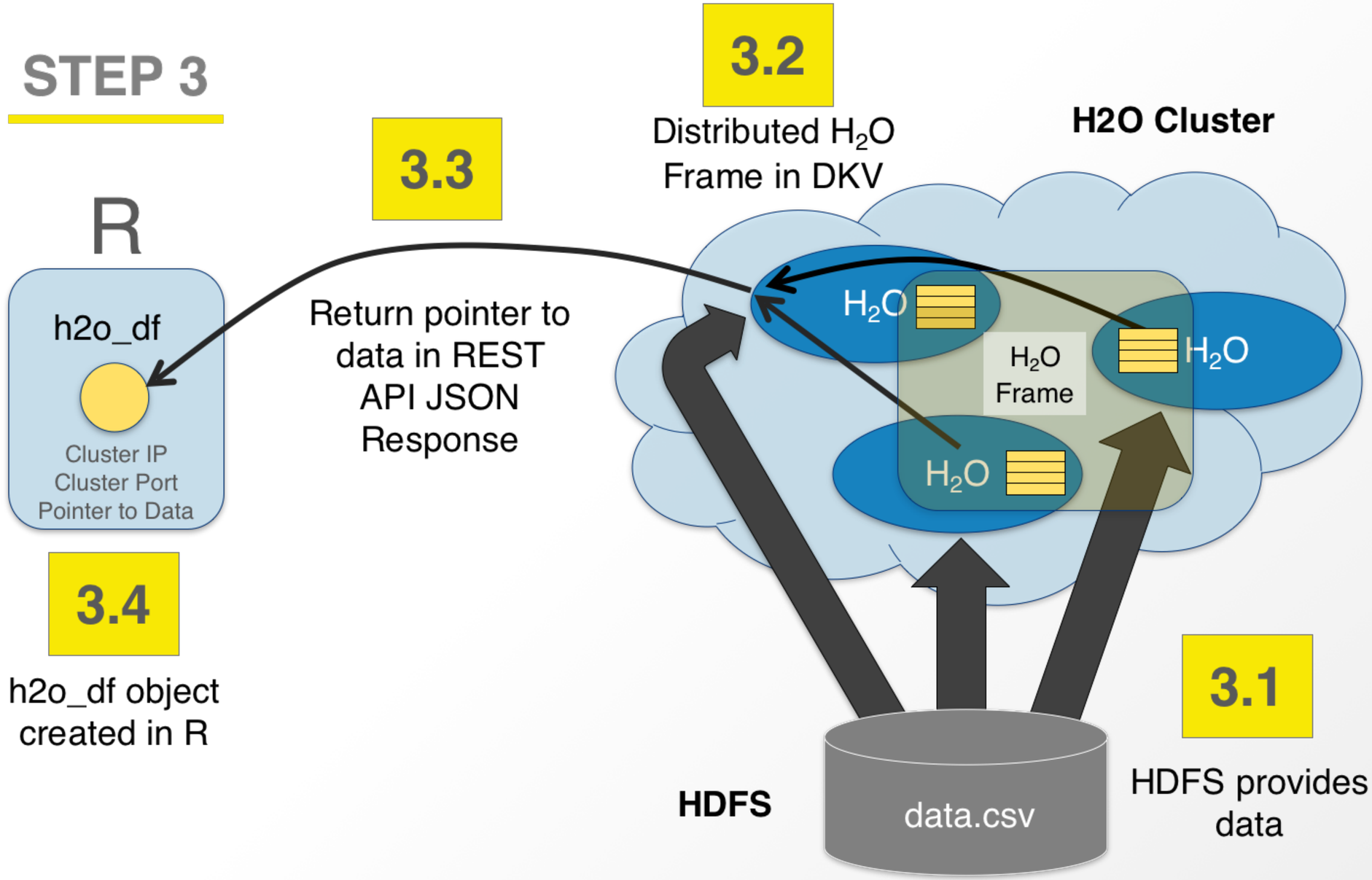
R user

```
h2o_df = h2o.importFile("../data/allyears2k.csv")
```

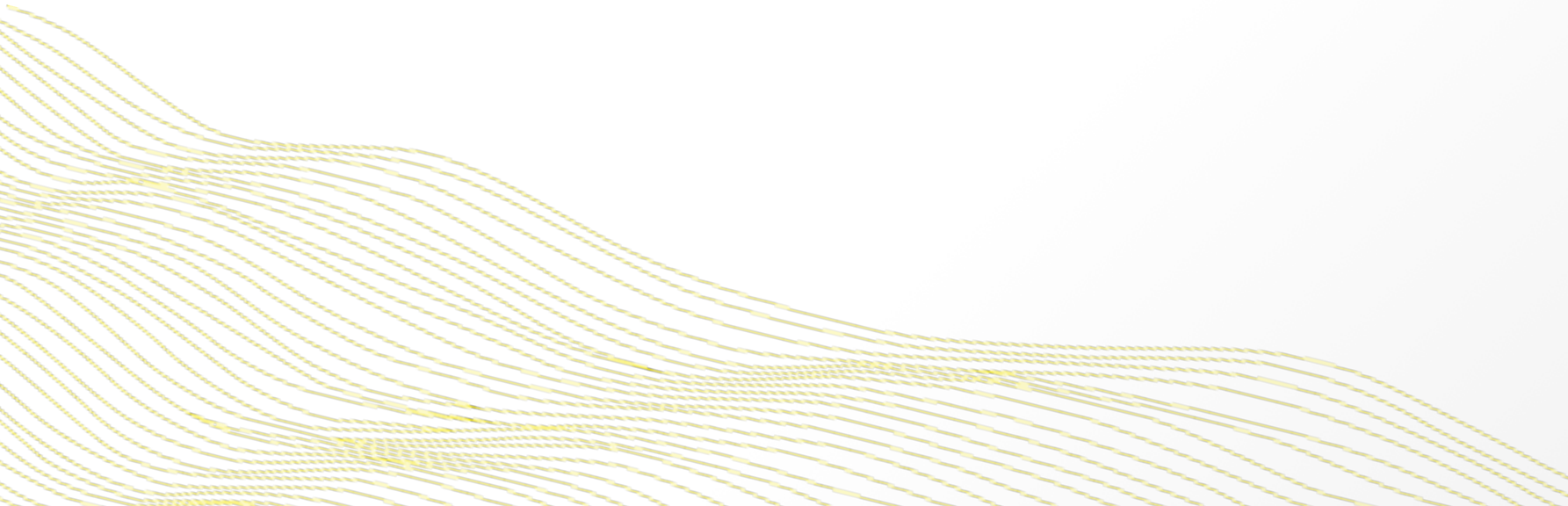
Reading Data from HDFS into H2O with R



Reading Data from HDFS into H2O with R



Natural Language Processing



Word2Vec

- Learns vector representations of words by analyzing large text corpus
  - Word Embeddings = mapping of words to vectors from a high dimensional space (100-1000)
  - Text sources: Google News, Wikipedia, Tweets, …

- Embeddings capture meaning of the word
  - Semantically similar words are close to each other
  - Can be used to find synonyms & analogies
  - Famous example:
    - » man is to woman as king is to *???* (queen)
    - » Vec(king) – Vec(man) + Vec(woman) = Vec(queen)

car

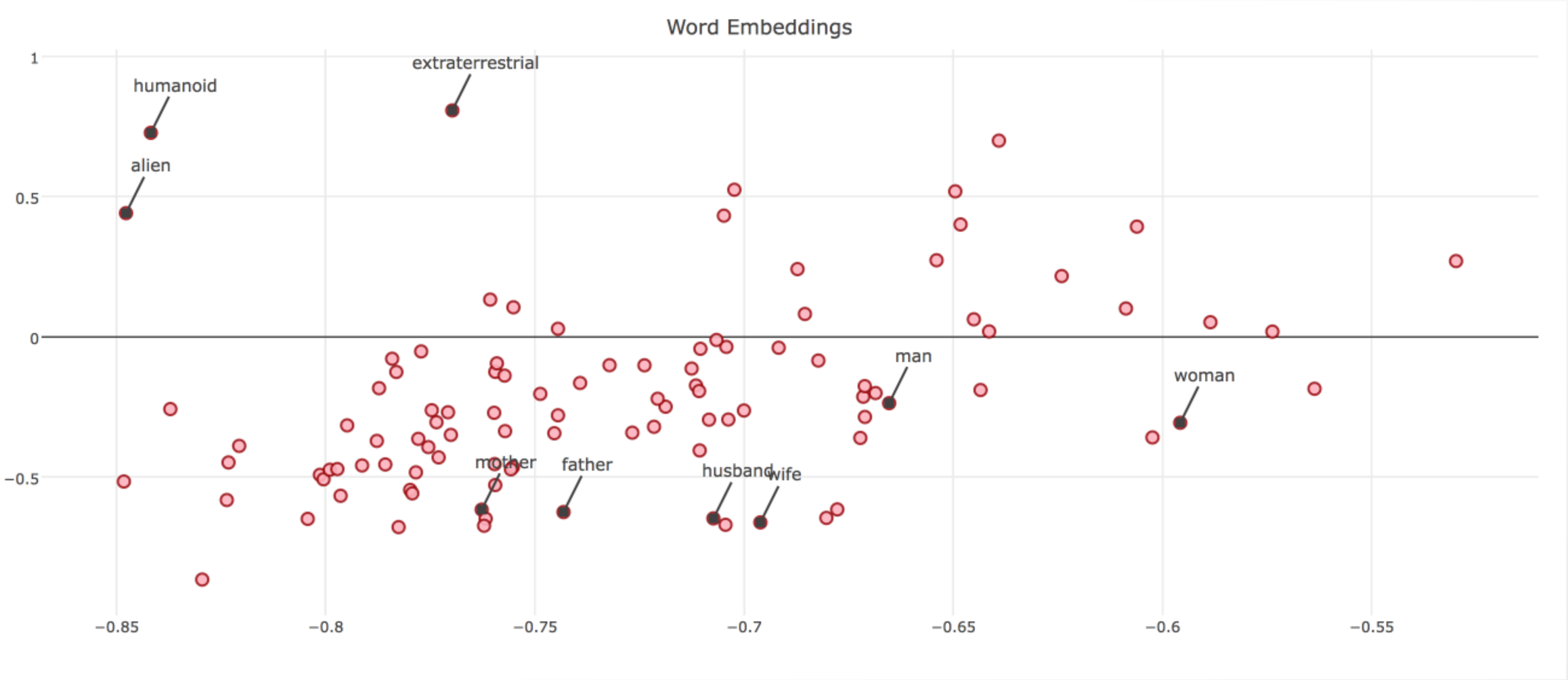
-0.09	-0.14	-0.12	-0.06	0.16

alien

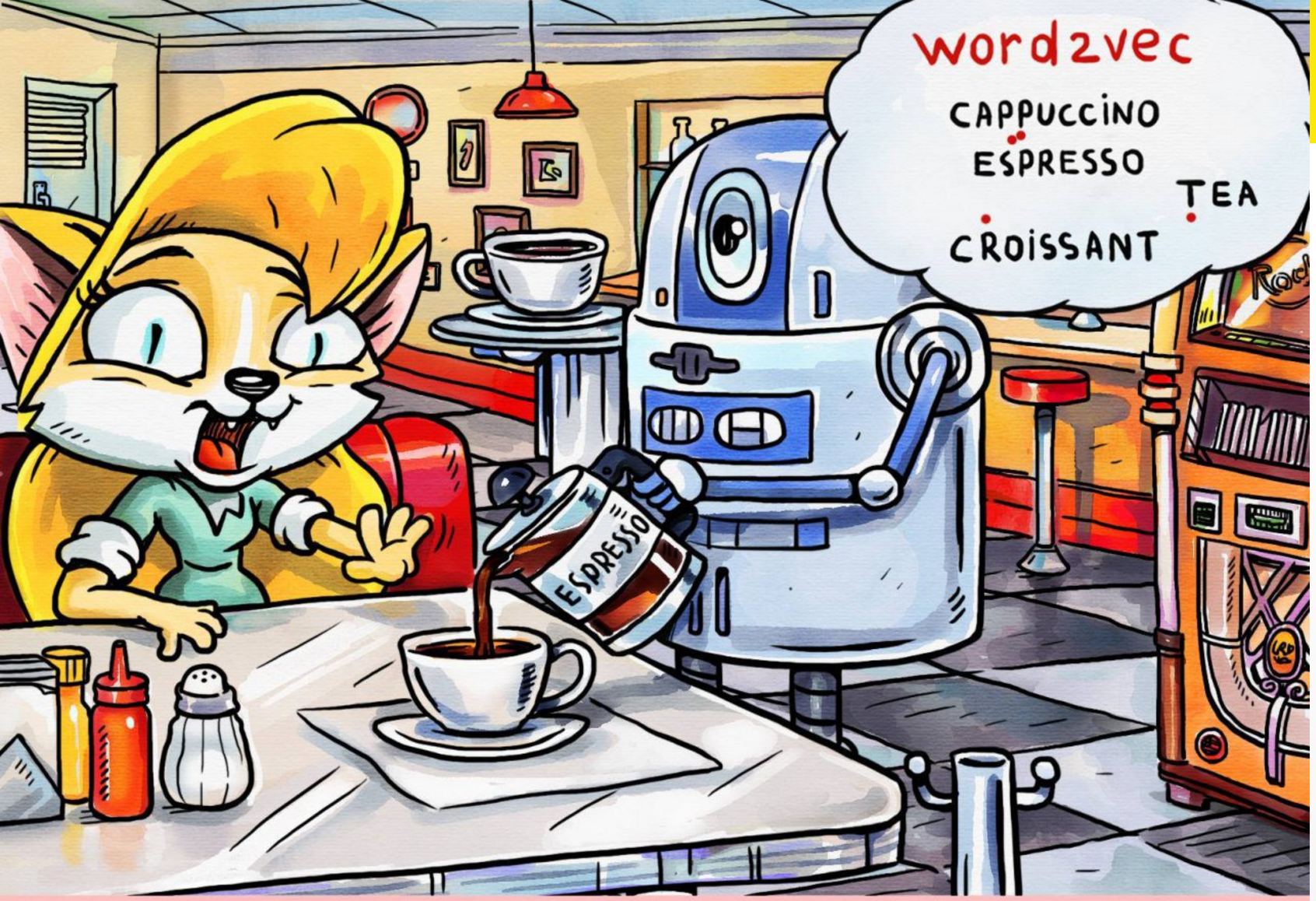
-0.38	-0.11	0.10	-0.26	-0.24

- Different variations of word2vec
  - Skip-Gram, CBOW
  - Hierarchical Softmax, Negative Sampling

Word2Vec


Word Embeddings
1
0.5
0
−0.5
humanoid
extraterrestrial
alien
man
woman
mother
father
husband
wife
−0.85
−0.8
−0.75
−0.7
−0.65
−0.6
−0.55

Word2Vec



- Espresso? But I ordered a cappuccino!
- Don't worry, the cosine distance between them is so small that they are almost the same thing.

Source:
twitter.com/DanilBaibak/status/844647217885581312

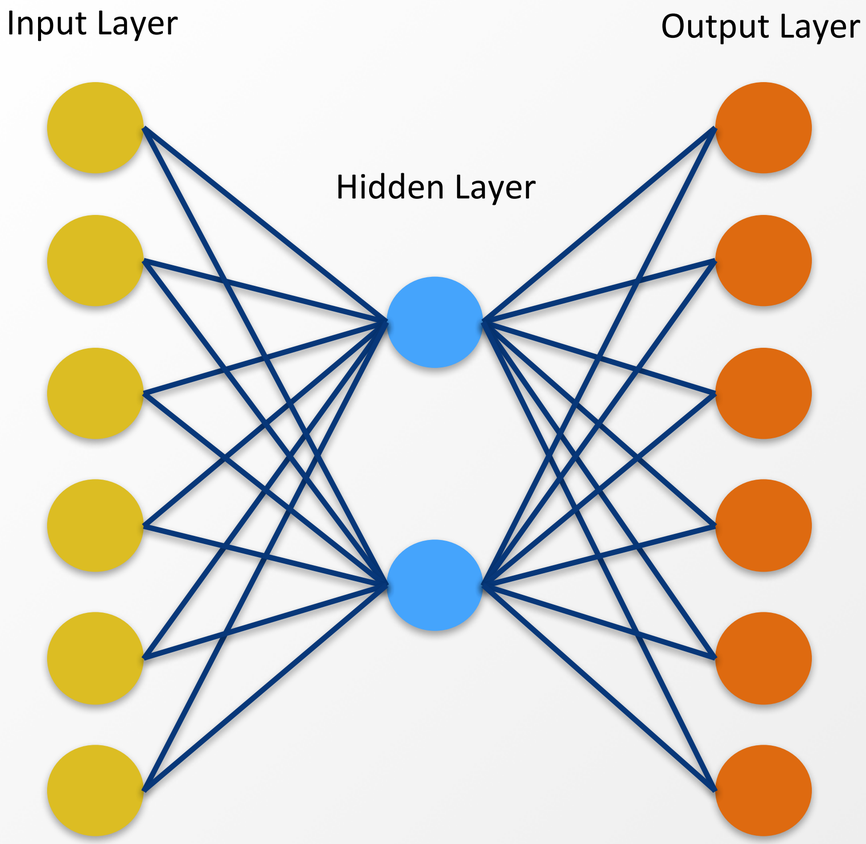
H2O.ai

Word2Vec Algorithm

- word2vec trains a Neural Network
  - Single hidden layer
  - Number of neurons in hidden layer = length of embeddings

- Uses a trick to formulate the problem as a supervised problem
  - The network is trained to predict target words based on given input words (window sliding over the text)
  - The weights on the edges connecting input layer to the hidden layer constitute the word embedding (weight matrix)
  - Similar approach to auto-encoders

- Optimizations
  - Hierarchical Softmax – binary tree to represent output probabilities (implemented in H2O)
  - Negative Sampling – sample the output vectors to be updated
  - HogWild! - lock-free parallelization (ignores conflicts)



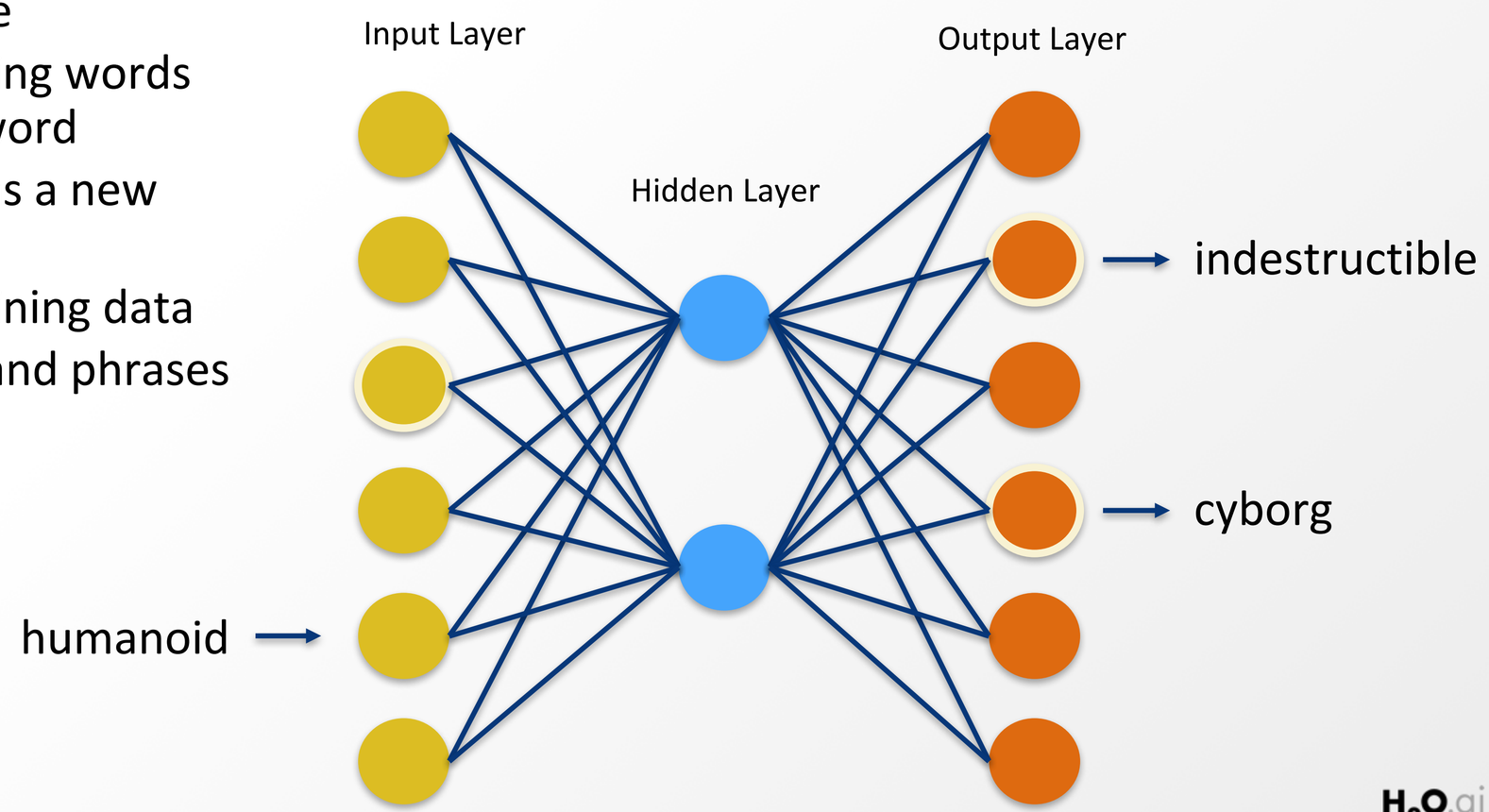
Word2Vec Algorithm

*”A seemingly **indestructible humanoid cyborg** is sent from 2029 to 1984 to assassinate a waitress, whose unborn son will lead humanity in a war against the machines, while a soldier from that war is sent to protect her at all costs.”*

- H2O supports Skip-Gram architecture
  - Network tries to predict surrounding words based (context) on a given input word
  - It treats each context-target pair as a new observation
- Skip-Gram works well with larger training data
  - Represents well even rare words and phrases
- CBOW support is planned

Given: **humanoid**

Predict: **indestructible** and **cyborg**



Word2Vec - Usage

- word2vec embeddings are typically used in a pre-processing step of another ML algo
- we need to aggregate the word embeddings for word sequences of variable length (sentences, paragraphs,...)
  - pooling
    - averaging (aka “mean pooling”)
    - (TF-IDF) weighted averaging
      - Not good for use cases when order of words is important (Sentiment Analysis)
      - http://jxieeducation.com/static/research/documentembedding_poster.pdf
    - PCA & Fisher Vectors: http://www.cs.tau.ac.il/~wolf/papers/qagg.pdf
  - simply concatenate embeddings
    - for short inputs, eg. job titles
- use an extension of word2vec
  - Paragraph Vectors/doc2vec
    - Adds “paragraph token” as an input of the training process, it acts as a memory that remembers what is missing from the current context – or the topic of the paragraph
    - https://cs.stanford.edu/~quocle/paragraph_vector.pdf

Word2Vec in an Example ML Workflow

Use Case: Predict movie genre from a movie synopsis/plot summary.

1. word2vec workflow
    1. Tokenize plots: break up plots into separate words
    2. Filter words: remove stop words like “and”, “the”, ”of”; remove too short words
    3. Train a word2vec model
    4. Sanity check: find synonyms based on the word embeddings
    5. (optional) Evaluate semantic accuracy of the model
        - But the best measure is always the performance of the overall problem (how well we predict the genres)
2. Use model to transform plots to vectors
3. Use plot vector representations with a supervised machine learning algorithm to predict the genre

Word2Vec in H2O – Key Functions

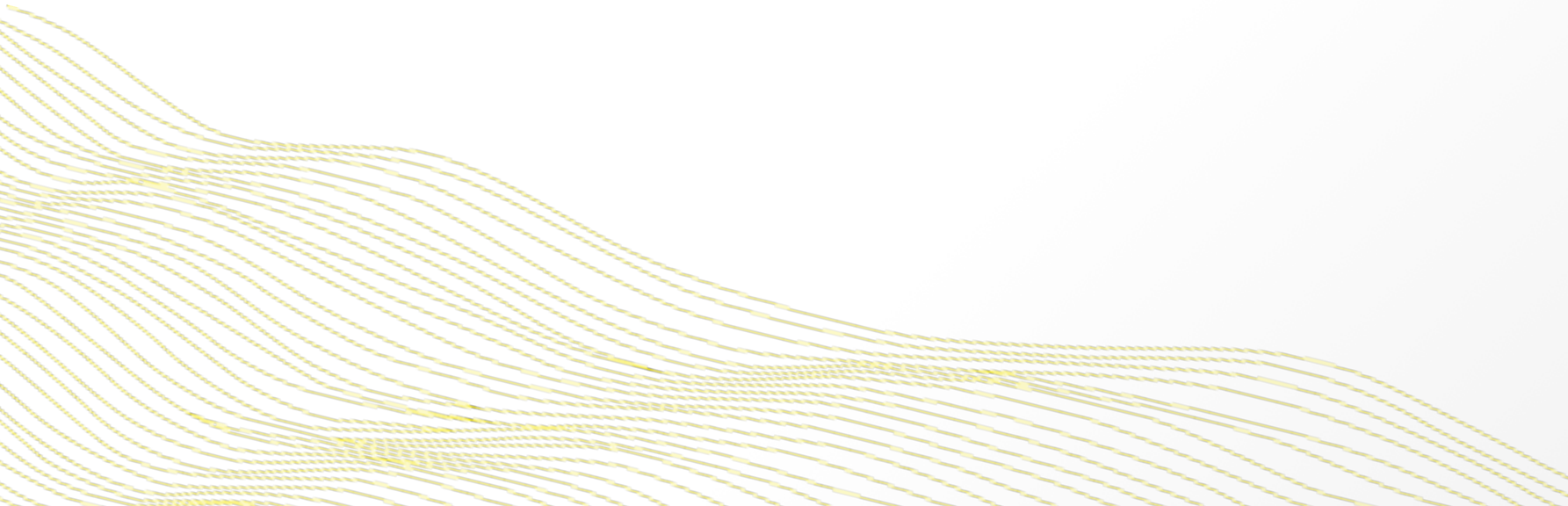
```
# Break job titles into sequence of words
words <- h2o.tokenize(movies$plot, “\\\\W+”)
```

```
# Build word2vec model
w2v_model <- h2o.word2vec(words, epochs = 10)
```

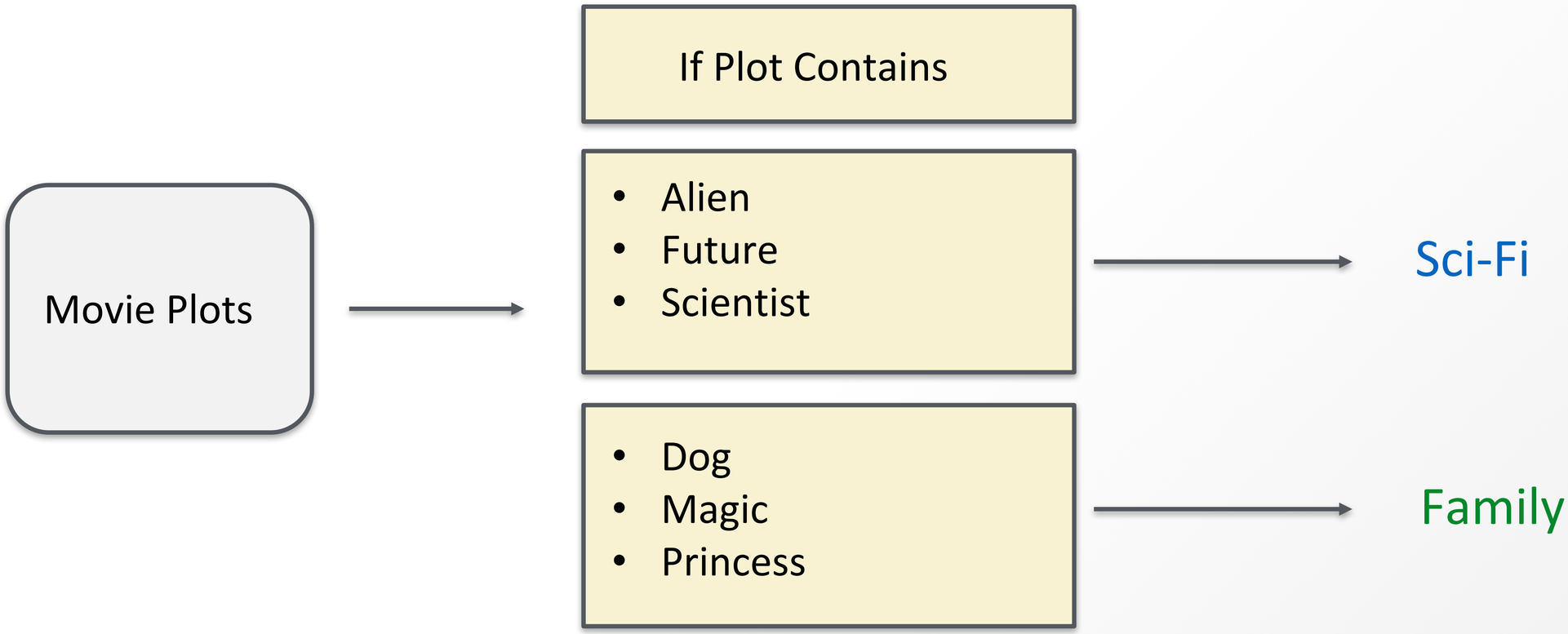
```
# Sanity check - find synonyms for the word 'teacher'
h2o.findSynonyms(w2v_model, "teacher", count = 5)
```

```
# Transform words into embeddings and aggregate for each plot
plot_vecs <- h2o.transform(w2v_model, words, aggregate = “AVERAGE”)
```

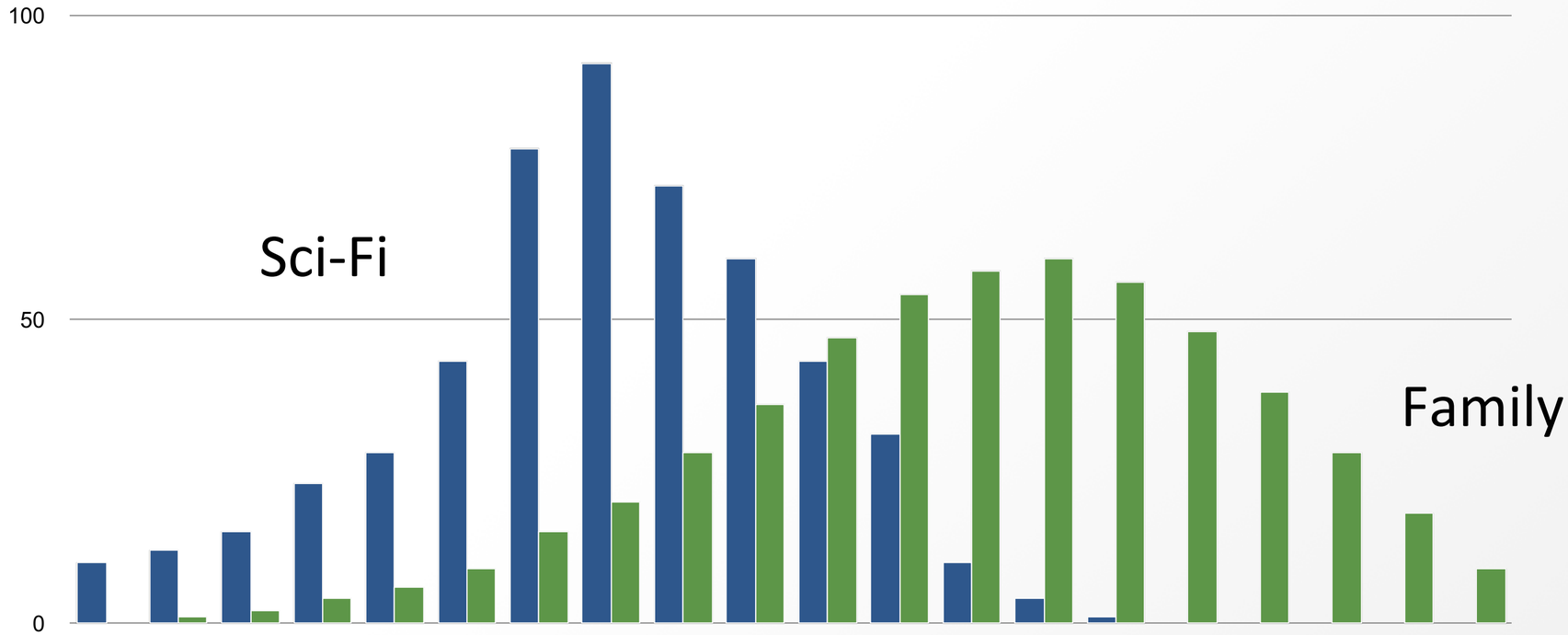
Machine Learning



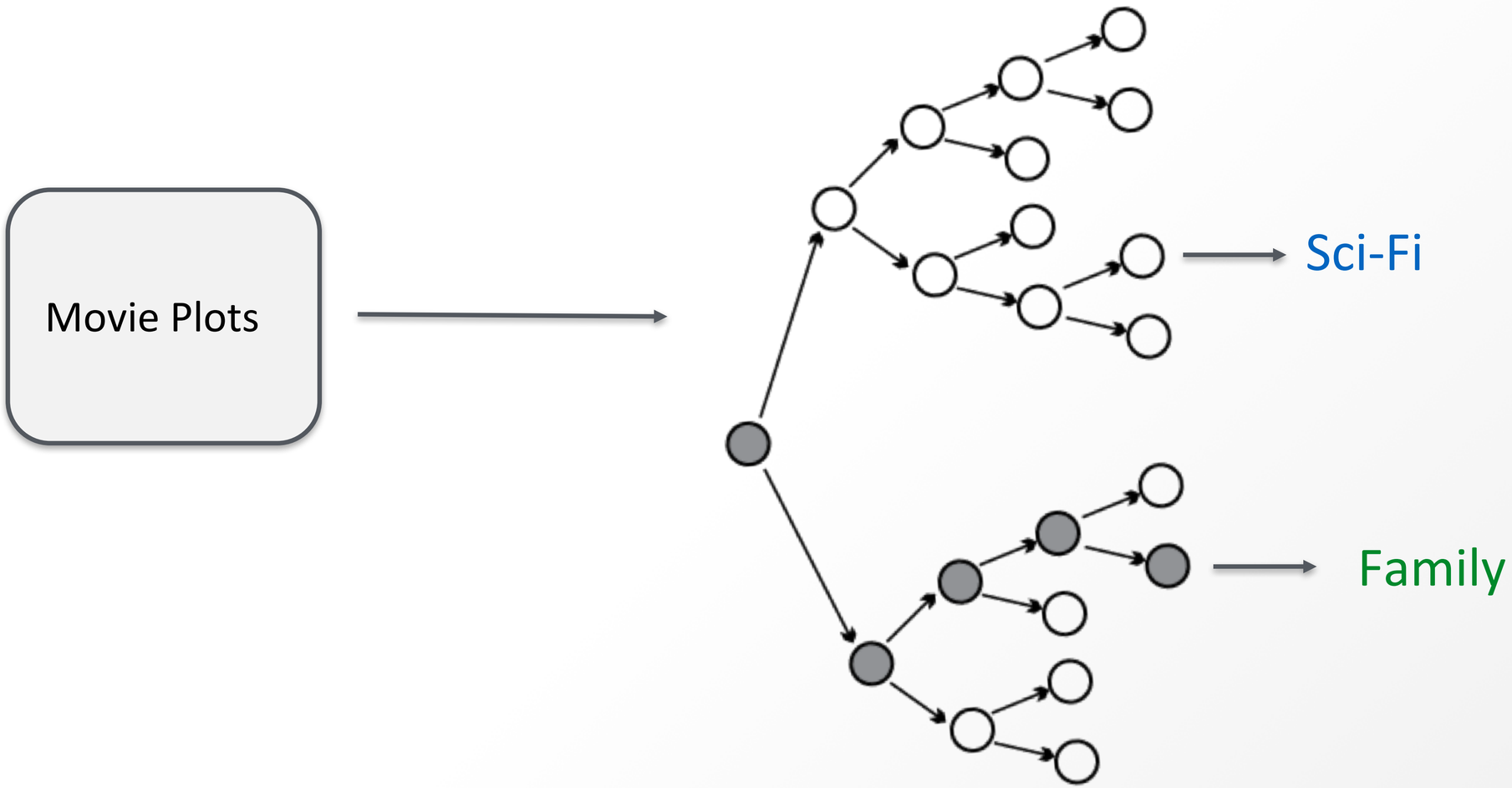
Rule Based Model



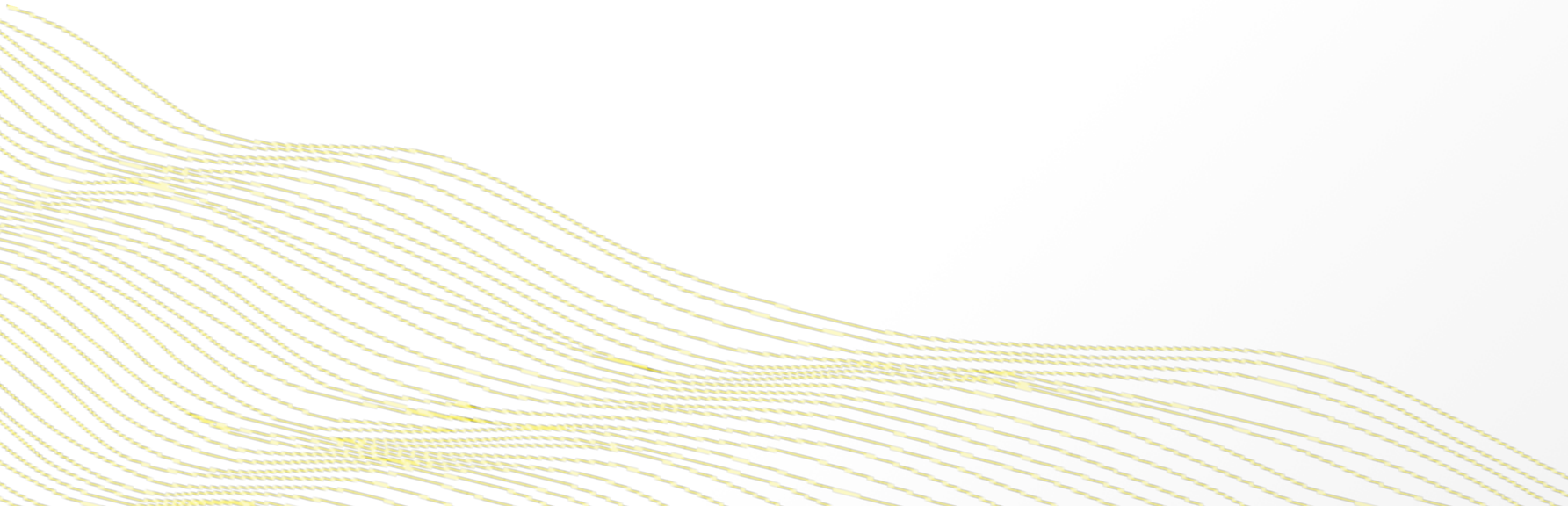
Machine Learning Model



Machine Learning Model

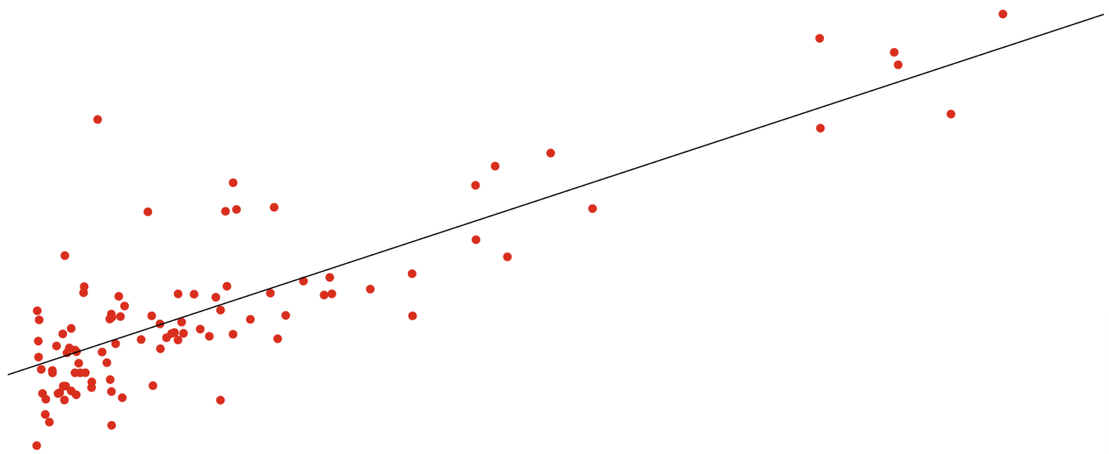


Supervised Learning



Supervised Learning

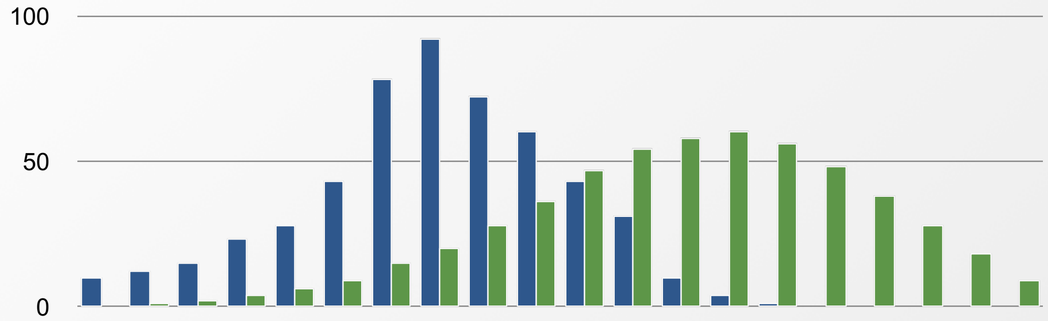
Regression



How much will a movie cost to make?

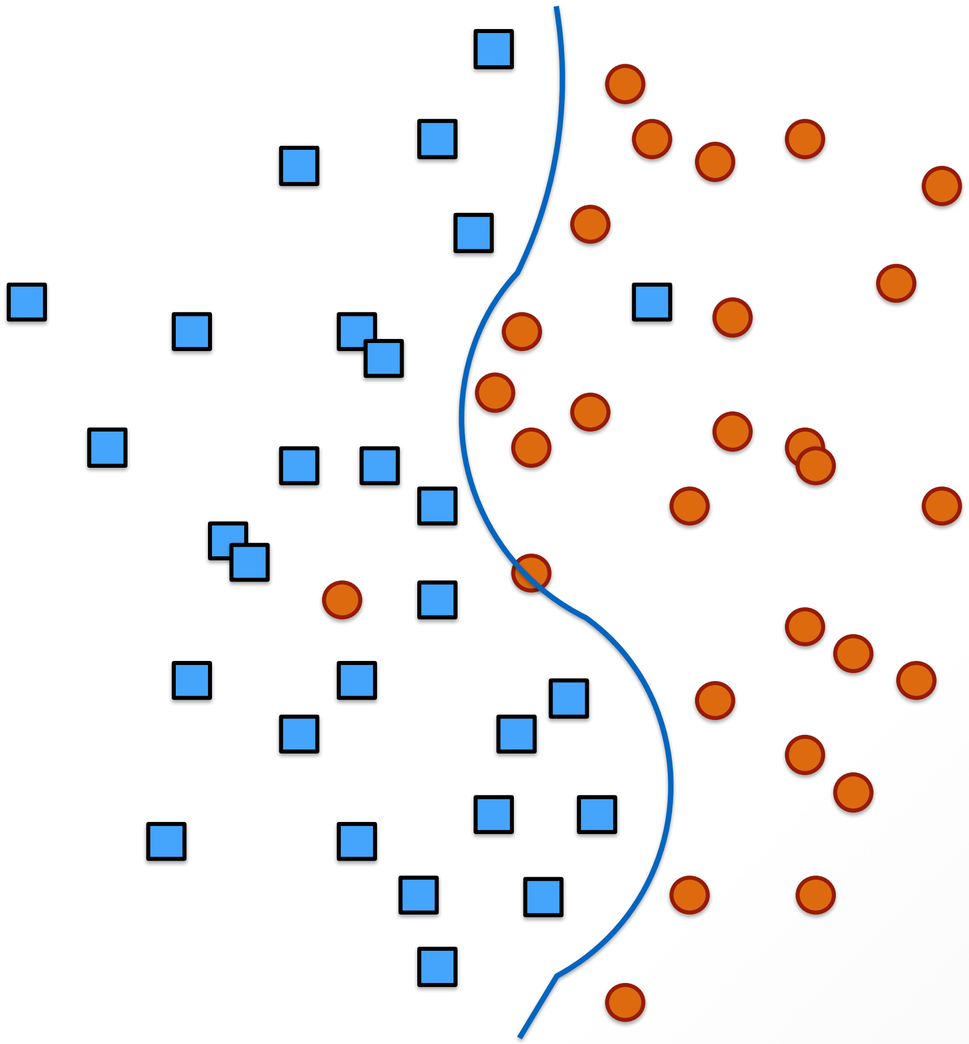
What is the genre of a movie? Is it a Comedy? Drama? Action?

Classification

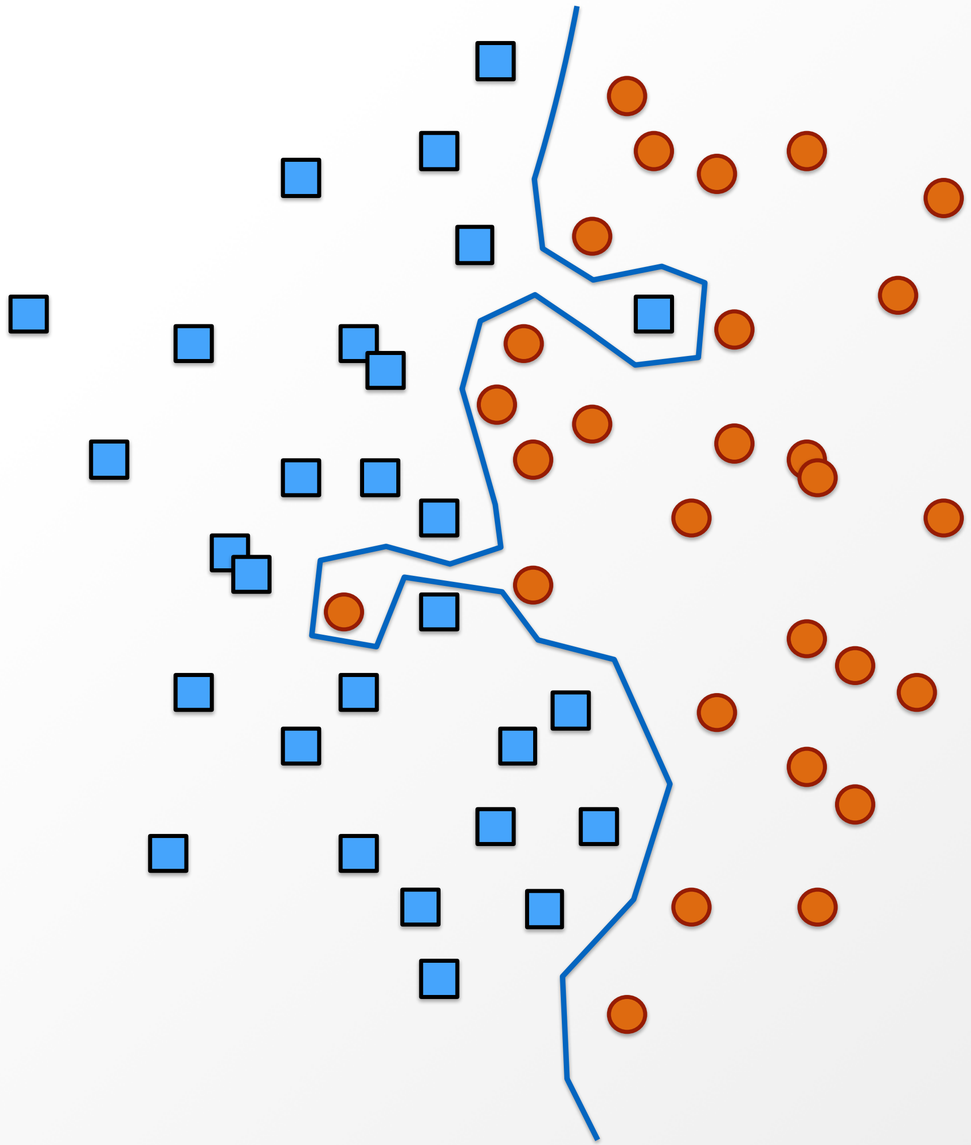


Overfitting

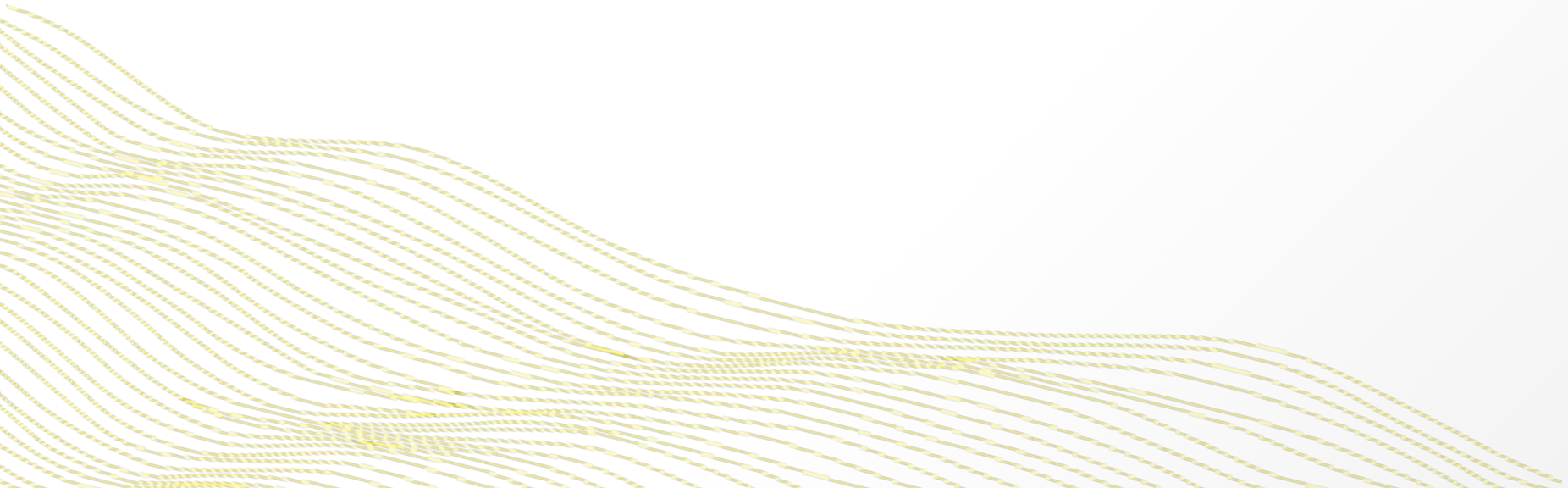
Finding the Signal in the Data



Memorizing the Data



Demo Time



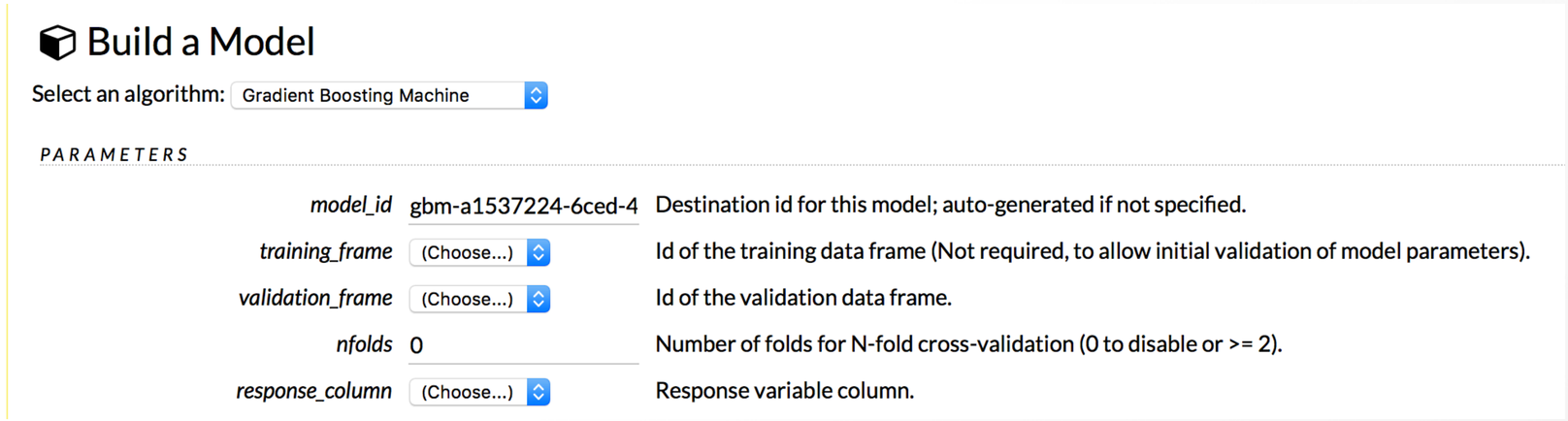
Establish a Baseline

In R:

```
print(”Build Gradient Boosted Machine with default parameters")

gbm_model <- h2o.gbm(x = myX, y = "genre",
                     training_frame = train, validation_frame = test)
```

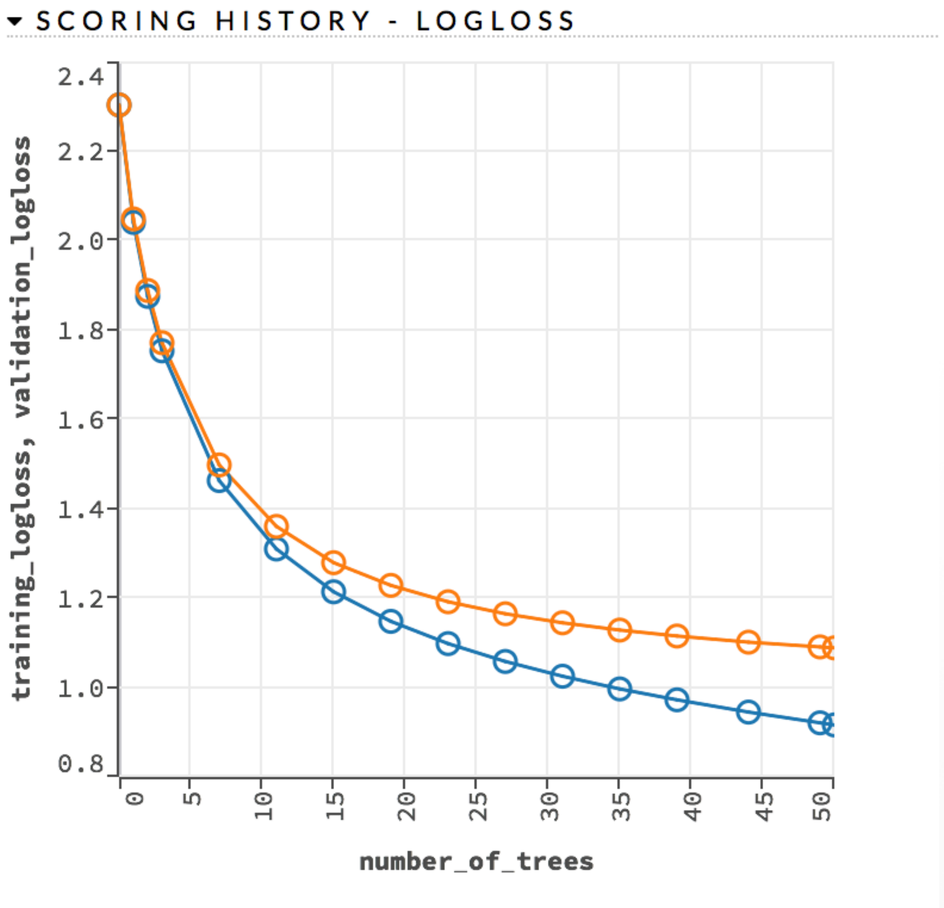
In Flow:



H2O.ai

Performance Metrics in Flow

Scoring History



Hit Ratio

k	hit_ratio
1	0.6182
2	0.8247
3	0.9027
4	0.9416
5	0.9651
6	0.9793
7	0.9885
8	0.9951
9	0.9986
10	1.0

Performance Metrics in Flow

Confusion Matrix on Validation Data

	Action	Comedy	Documentary	Drama	Family	Horror	Romance	Sci-Fi	Thriller	Western	Error	Rate
Action	161	96	82	371	5	17	2	8	13	18	**0.7917**	**612 / 773**
Comedy	38	1980	351	1529	31	39	10	7	15	15	**0.5068**	**2,035 / 4,015**
Documentary	14	217	5330	842	13	12	1	13	10	3	**0.1743**	**1,125 / 6,455**
Drama	54	838	1027	5676	17	56	13	13	50	43	**0.2711**	**2,111 / 7,787**
Family	10	147	87	186	95	13	3	11	3	2	**0.8294**	**462 / 557**
Horror	12	128	104	406	6	270	0	3	23	8	**0.7188**	**690 / 960**
Romance	3	60	17	250	4	2	13	2	3	0	**0.9633**	**341 / 354**
Sci-Fi	11	19	69	100	2	16	0	79	5	1	**0.7384**	**223 / 302**
Thriller	18	119	82	512	2	42	1	5	56	3	**0.9333**	**784 / 840**
Western	16	63	11	158	1	2	0	0	3	322	**0.4410**	**254 / 576**
Total	**337**	**3667**	**7160**	**10030**	**176**	**469**	**43**	**141**	**181**	**415**	**0.3818**	**8,637 / 22,619**

Examining Results

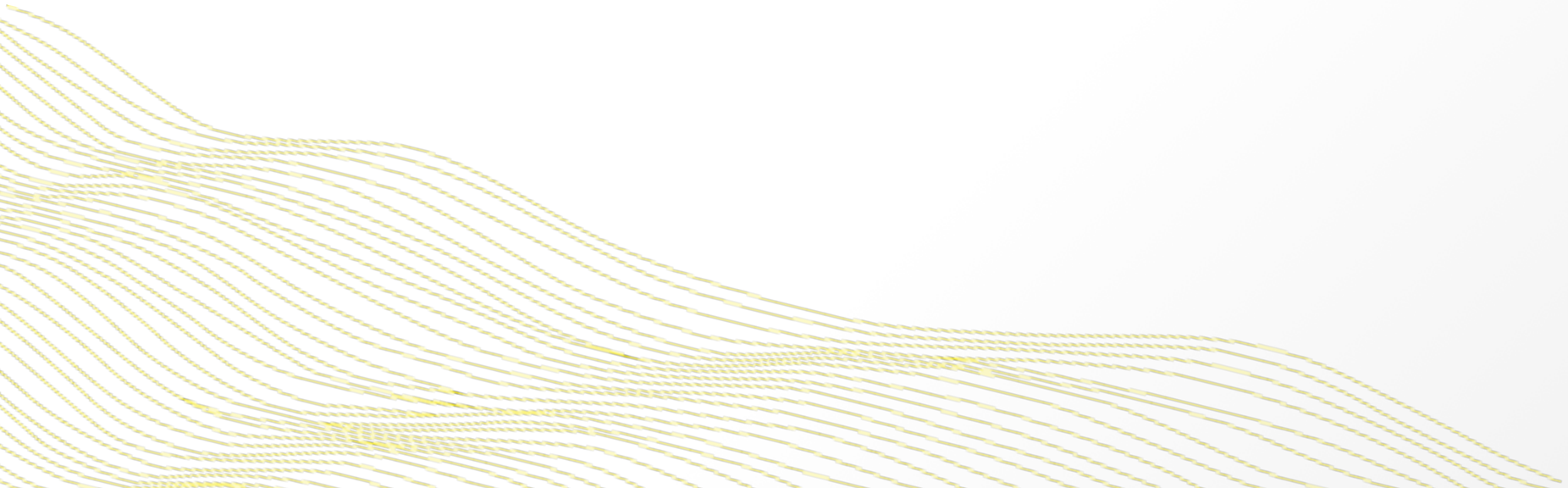
We see that the model predicted a lot of movies that were Thriller as Drama.

Why?

Plot of a Drama from the Training Data:

"Cordelia Gray is the reluctant owner of a ramshackle investigation agency following the suicide of her boss. Watching over her as she hunts down clues in the murky and sinister world of crime, is her straight-laced and intuitive office assistant Edith Sparshott."

Demo Time

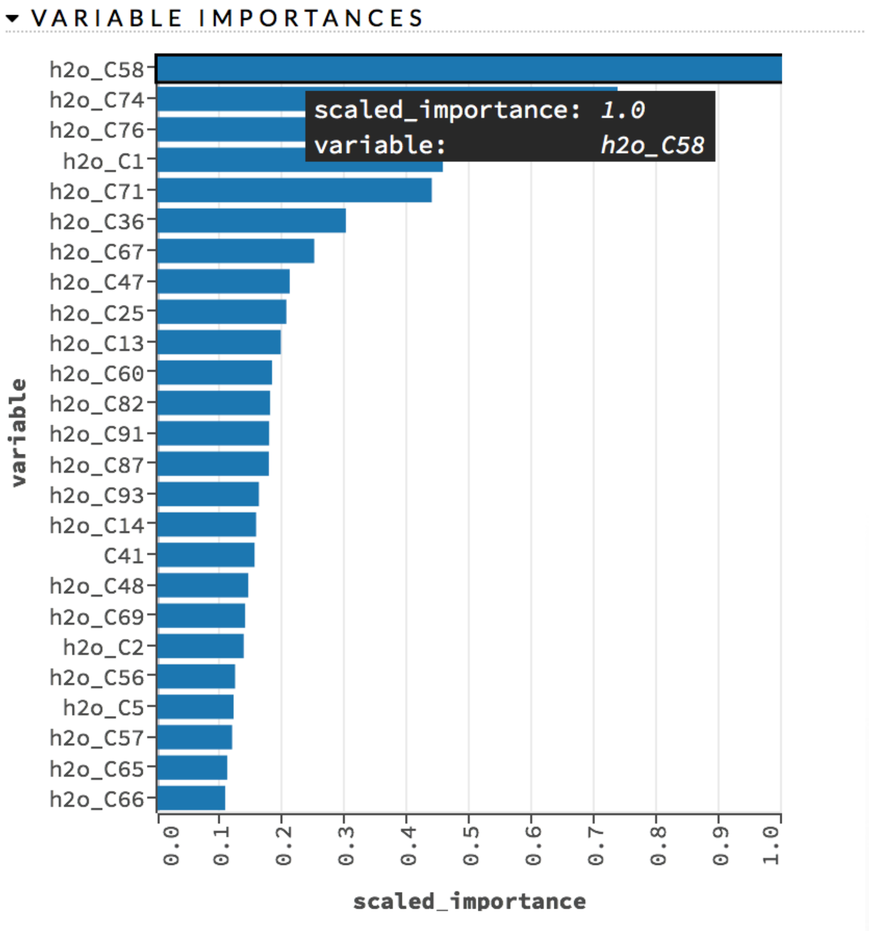


Adding New Features

Features	Mean Per Class Error
• Pre-trained Word Embeddings	0.637
• Pre-trained Word Embeddings • H2O Word Embeddings	0.542

Examine Results

Variable Importance



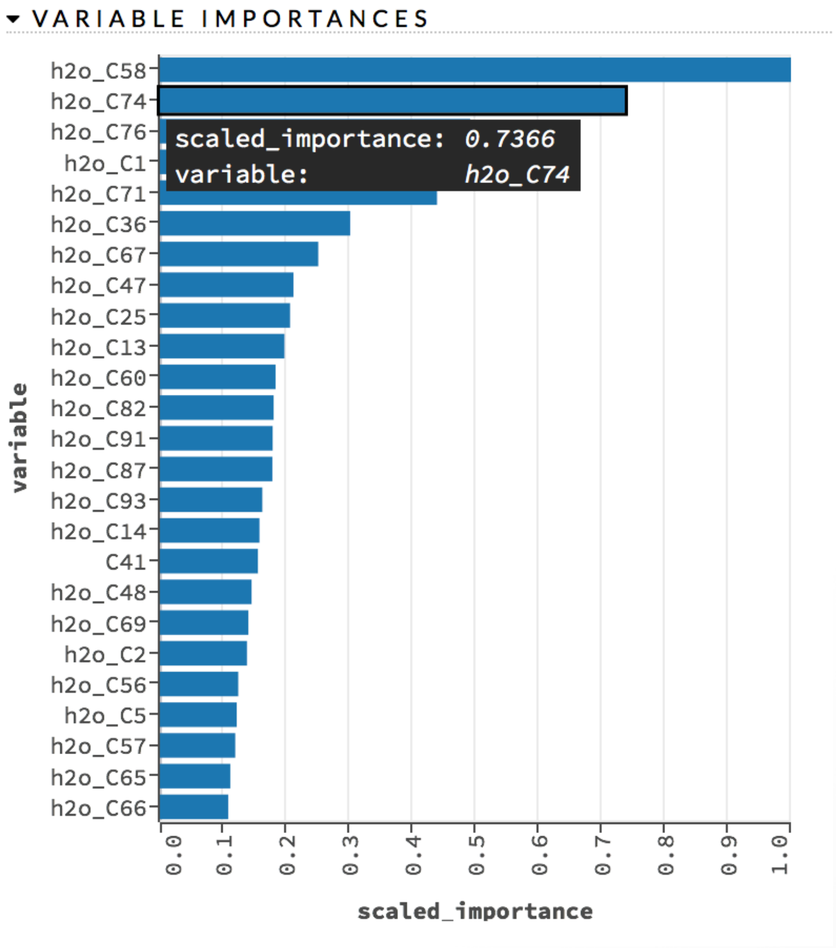
Words with High C58

- spirituals
- percussion
- philharmonic
- orchestral
- concerto
- sonata
- hadyn

H2O.ai

Examine Results

Variable Importance



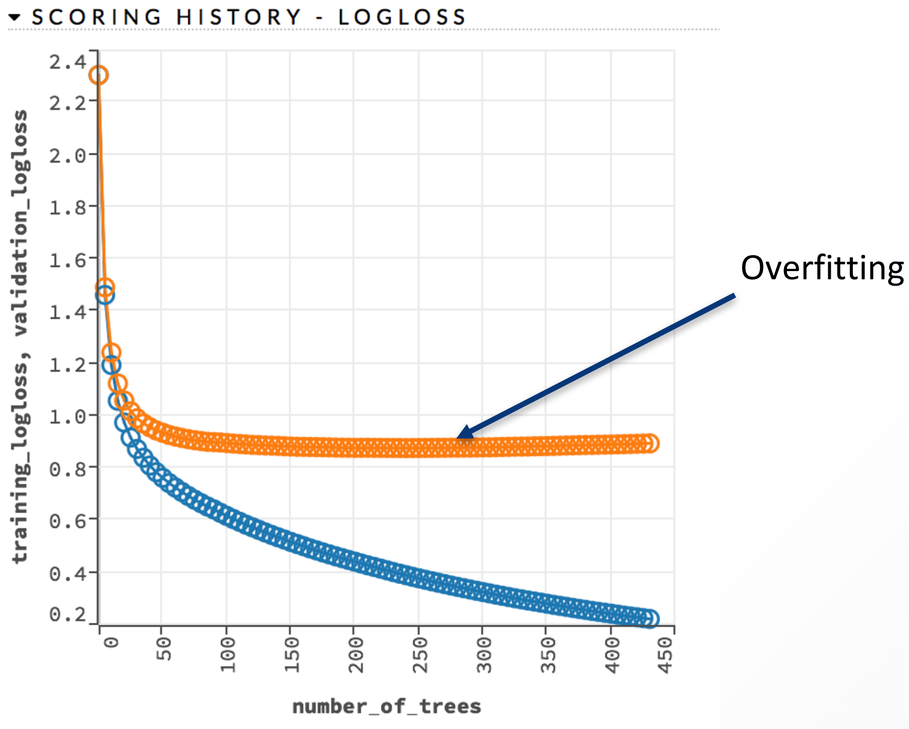
Words with High C74

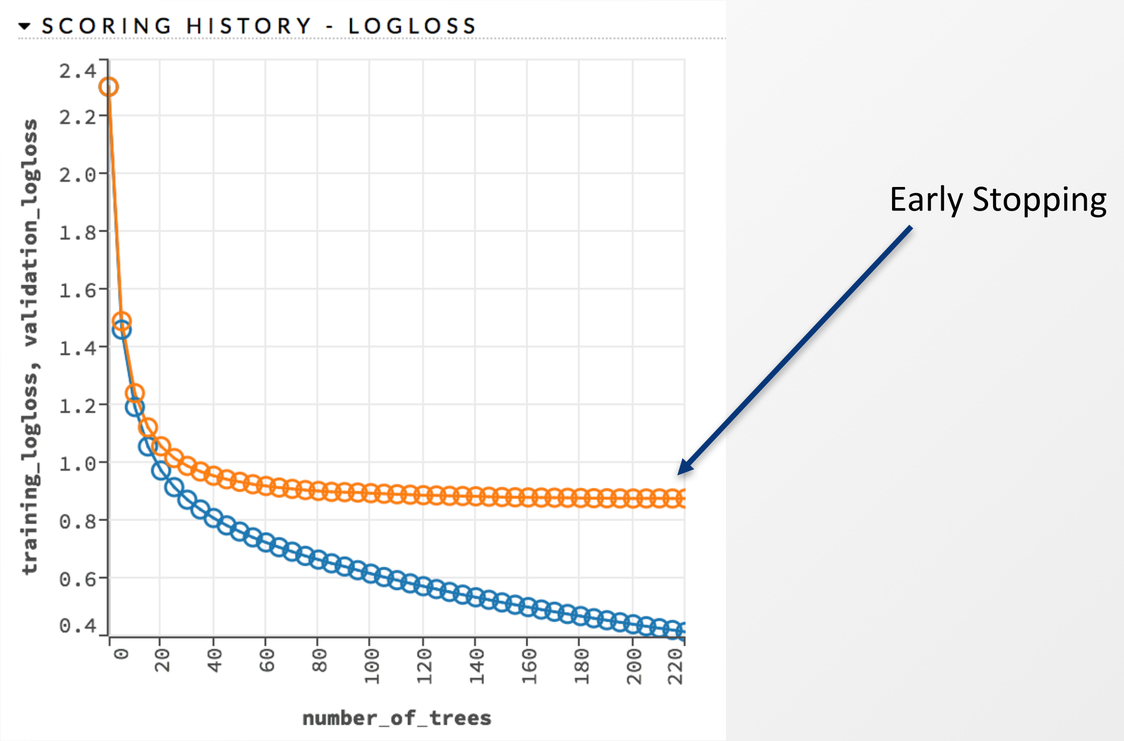
- einsteins
- patrolmen
- upperclassman
- speakeasies
- razzle

Early Stopping

Early stopping once the validation mean per class error doesn't improve by at least 0.01% for 5 consecutive scoring events

- `stopping_rounds = 5`
- `stopping_tolerance = 1e-4`
- `stopping_metric = "mean_per_class_error"`





Examine Results

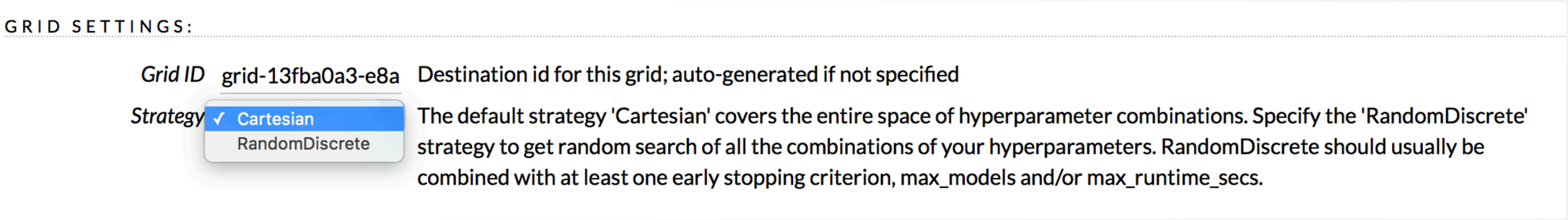
Model	Mean Per Class Error
• Pre-trained Word Embeddings	0.637
• Pre-trained Word Embeddings • H2O Word Embeddings	0.542
• Pre-trained Word Embeddings • H2O Word Embeddings • Early Stopping	0.503

Grid Search

In R:

```
grid <- h2o.grid(hyper_params = list(max_depth = c(1:20)),
                 search_criteria = list(strategy = "RandomDiscrete", max_runtime_secs = 3600),
                 algorithm = "gbm“, ..)
```

In Flow:



Questions?