DATASTAX

Please Join our WiFi Network
WeWork Guest

Real-time AI

Out of the Lab, Into Prod

Introduction

Hilton Rosenfeld

- Developer / Architect
- Application Modernization
- Digital Transformation
- IT Operations Management
- CI / CD

Build your own NLP text classifier and expose it as an API

FastAPI

DATASTAX ASTRA DB

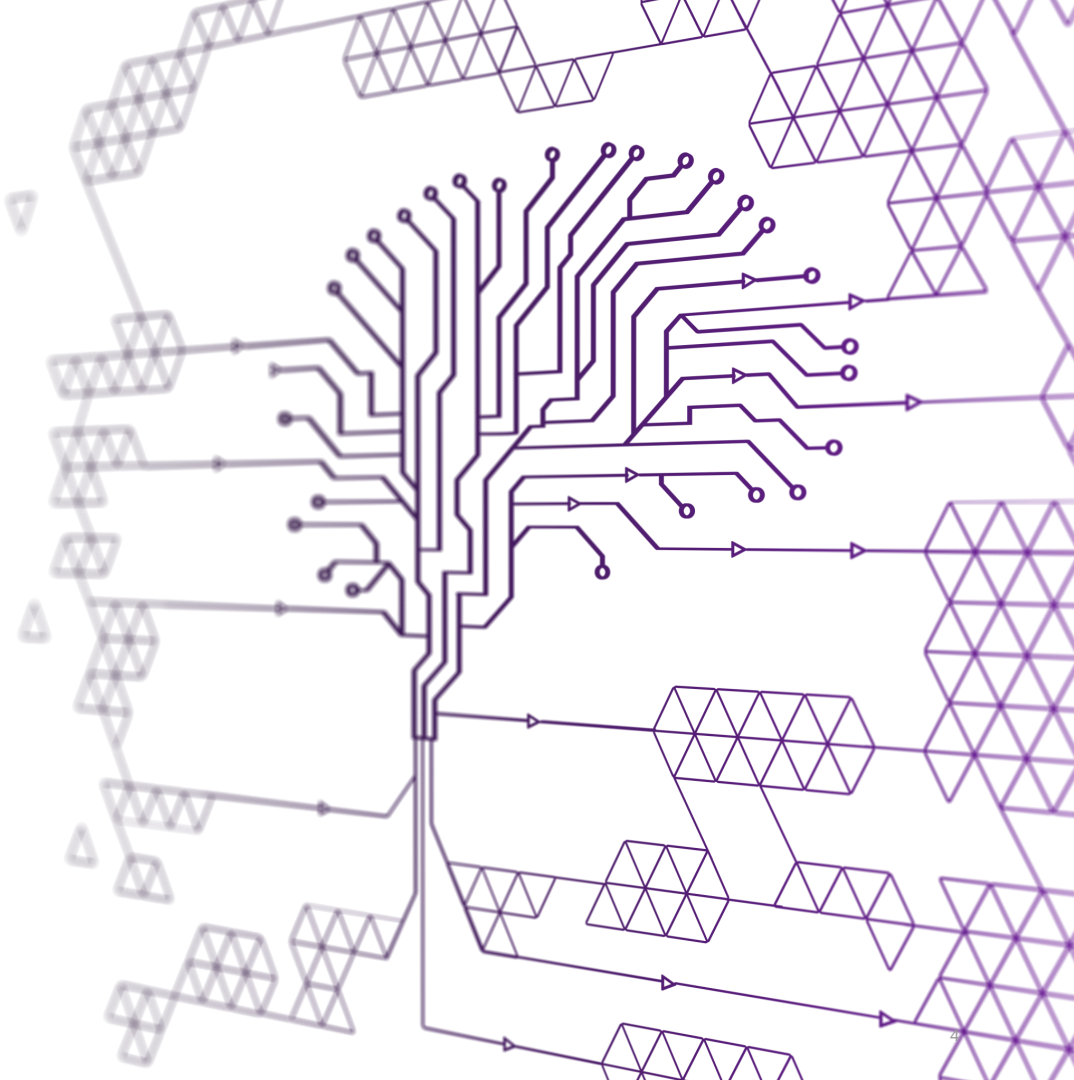
Housekeeping

- Break(s) will be provided.
- Refreshments will be served.

- Ask Questions !
- Scroll through exercises at your own pace.

- Connectivity Issues:
  - disable VPN and/or Firewall

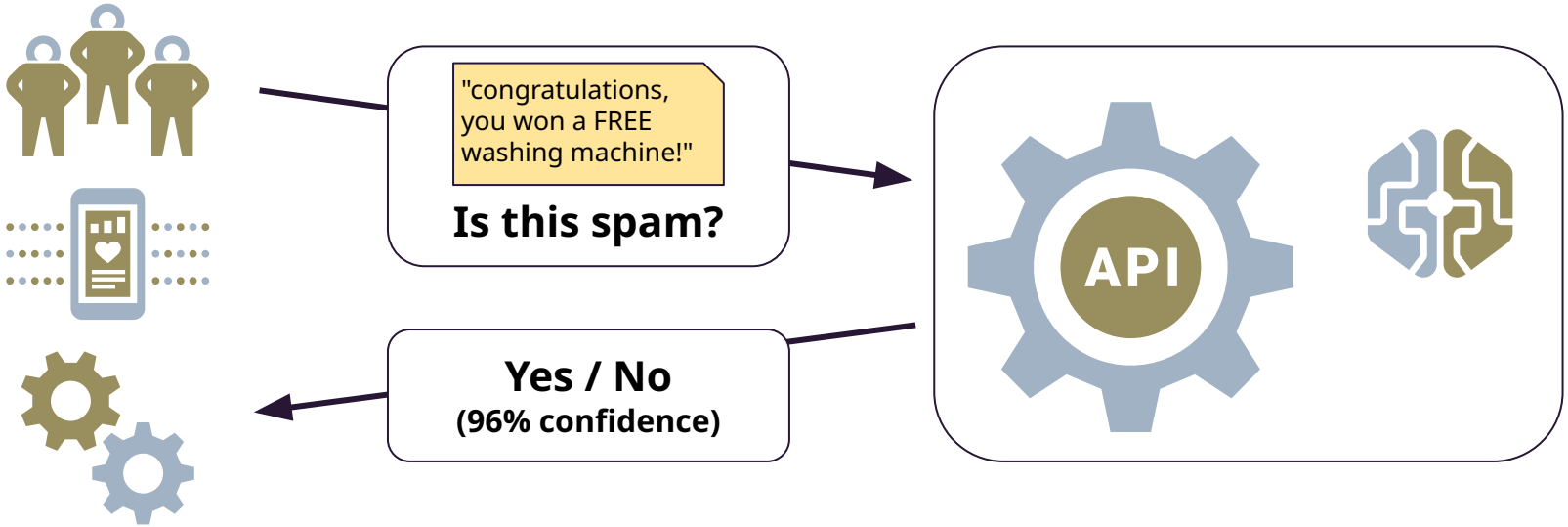
What we are building today



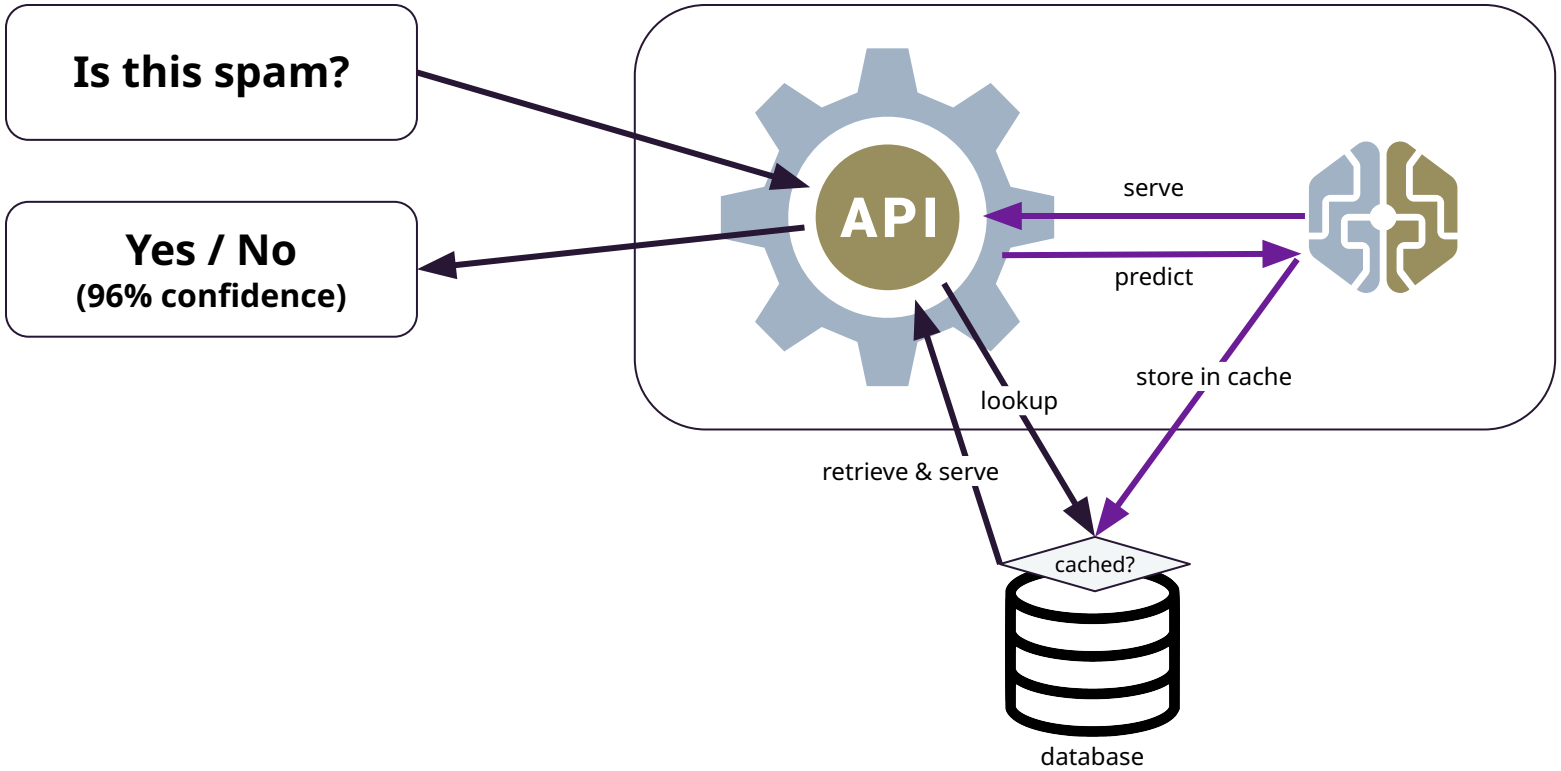
Spam

- 85% of emails are spam:
  122, 330, 000, 000 daily
  - Advertising - 36%
  - Adult-related content - 31.7%
  - Financial matters - 26.5%
  - Scams and fraud - 2.5%
- For every 12,500,000 emails sent, spammers receive one reply.
- Email spam costs businesses **$20.5 billion** annually.

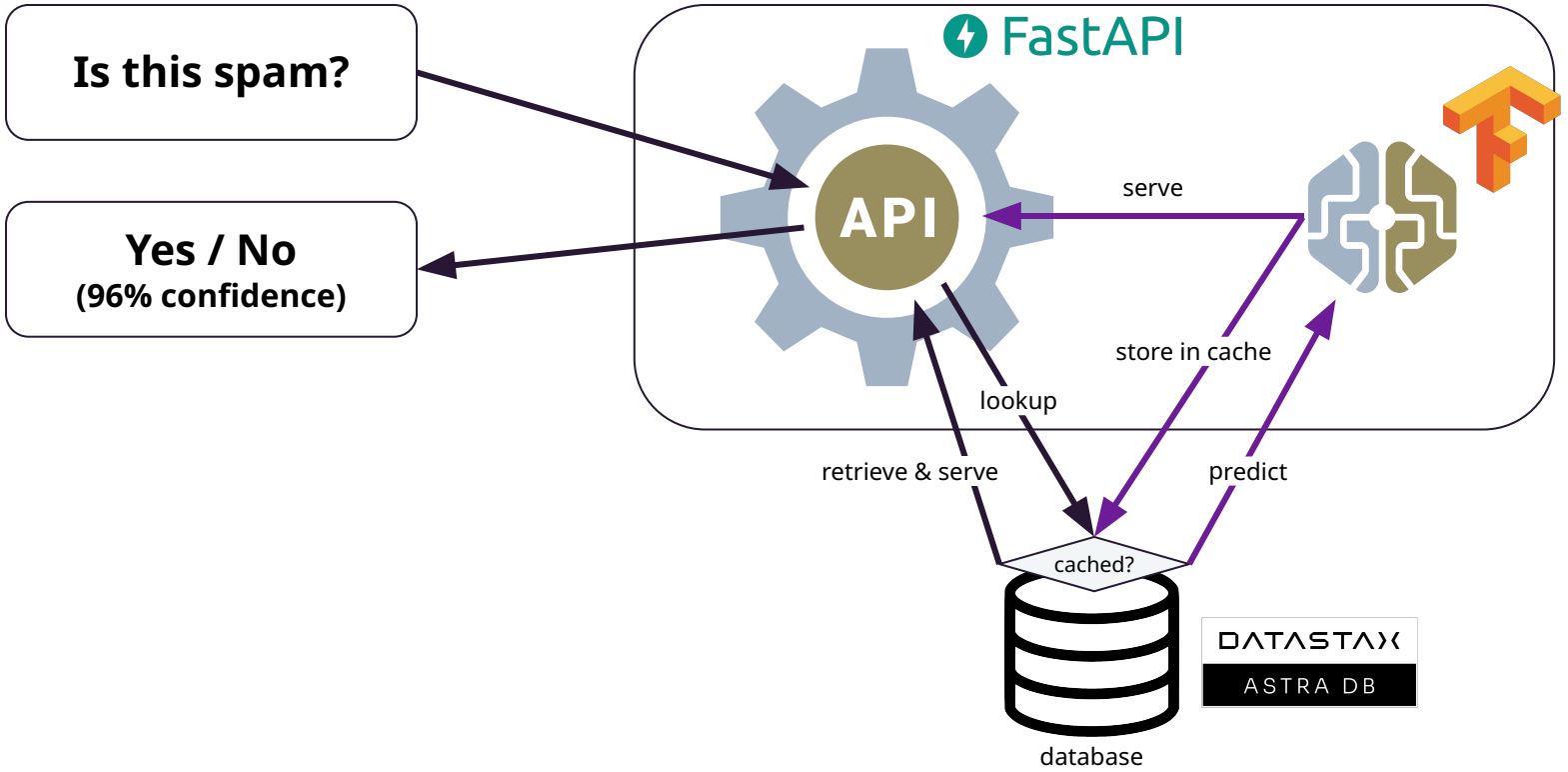
What we want



Architecture sketch



Tech stack

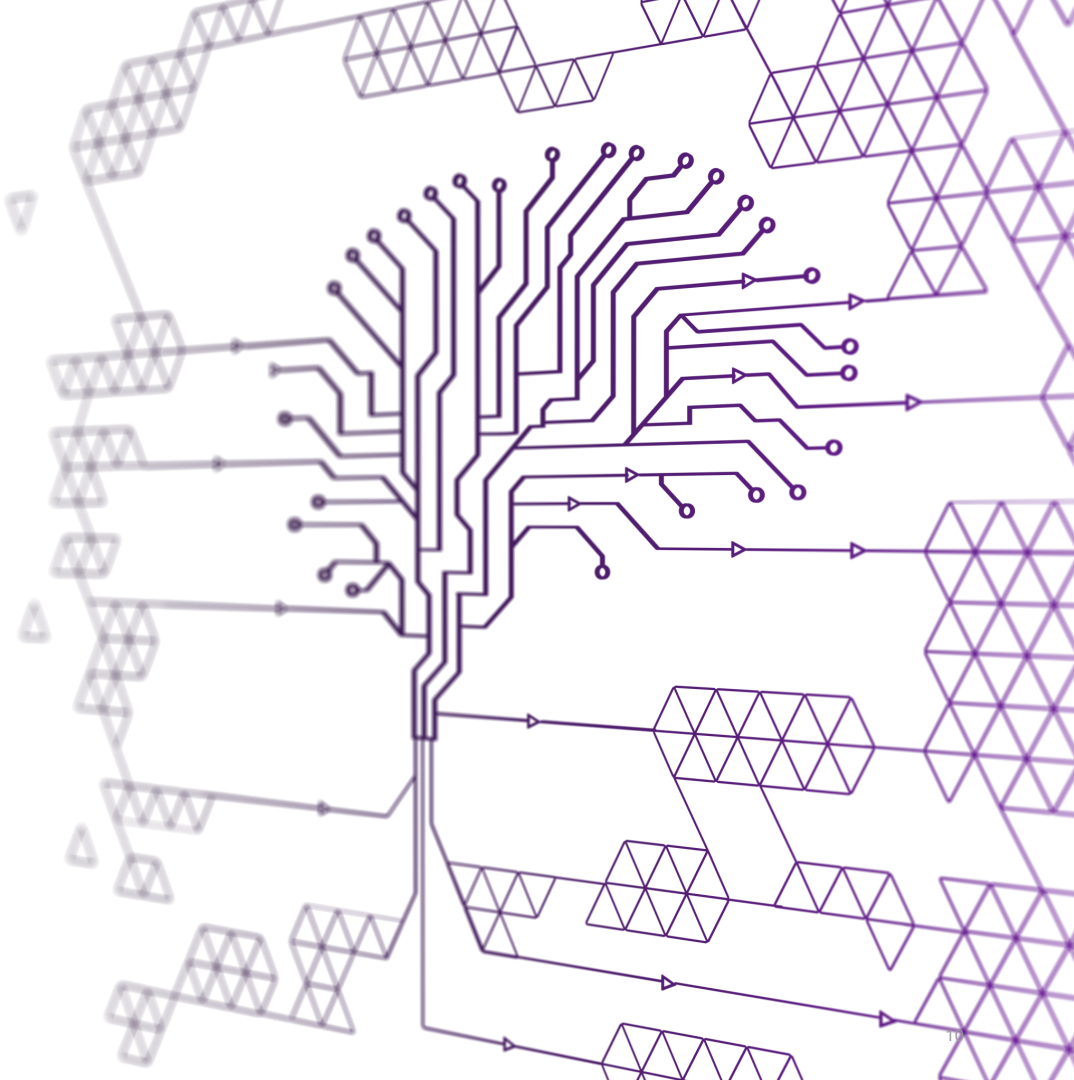


Applying Machine Learning to the problem of Spam filtering

1. Create a deep learning model from multiple spam datasets
2. Make that available as a REST API using Jupyter, TensorFlow, FastAPI and DataStax Astra DB.

- Combining 2 open source spam datasets curated by The University of California, Irvine (UCI).
- Preparing a new dataset to be fully ready for training a model.
- Building and training an LSTM model leveraging Keras and TensorFLow.
- Creating an API to expose the model for use leveraging FastAPI
- Caching predictions and inference data

The AI



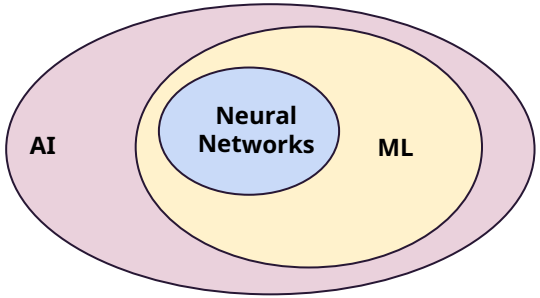
AI and ML

"ML: LSTM RNN for NLP"

Machine Learning = *algorithms that improve by being fed data, without explicit instructions what to do.*

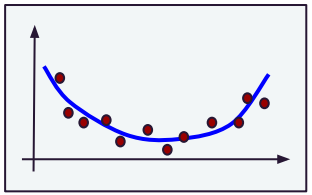
It's essentially statistical inference *(with superpowers)*.

Lots of math involved (*linear algebra, calculus, probability/statistics*).
Nowadays accessible as neatly-packaged tools (good for us!)

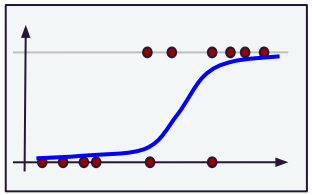


Simple examples of ML:

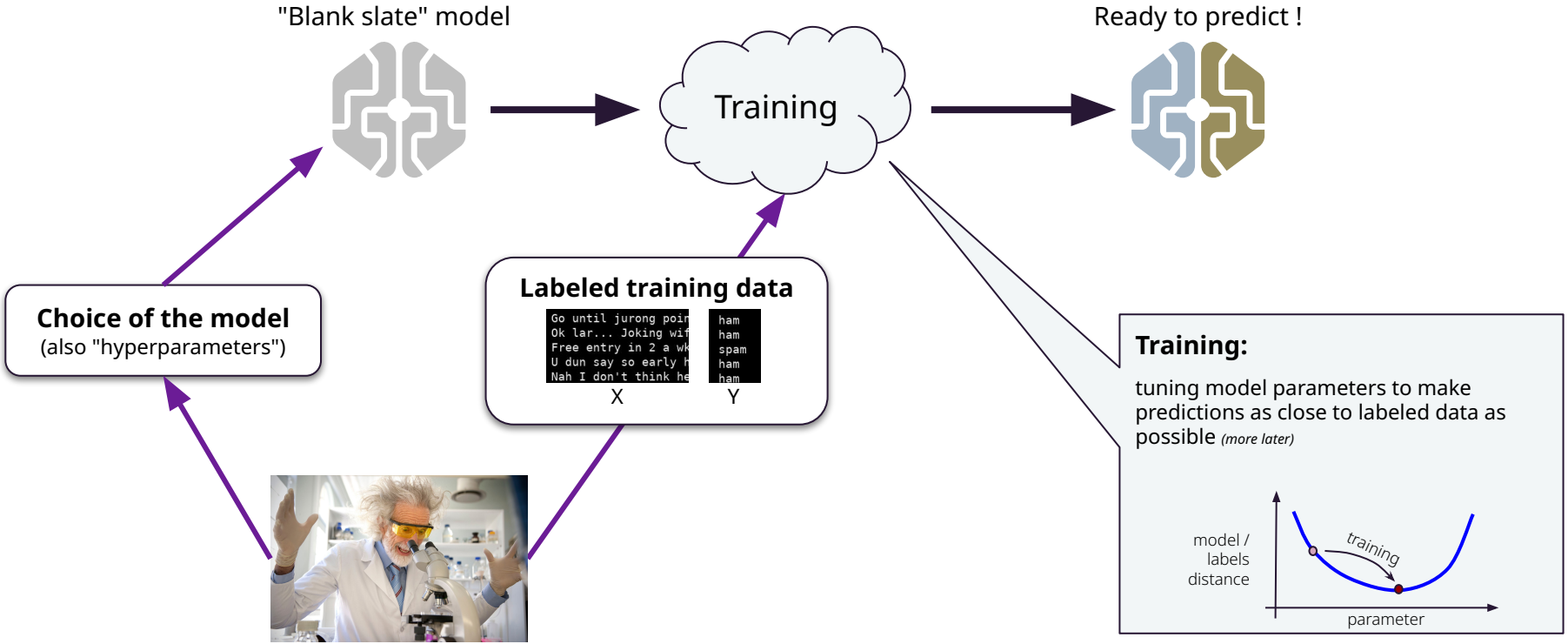
Least-Squares Fits



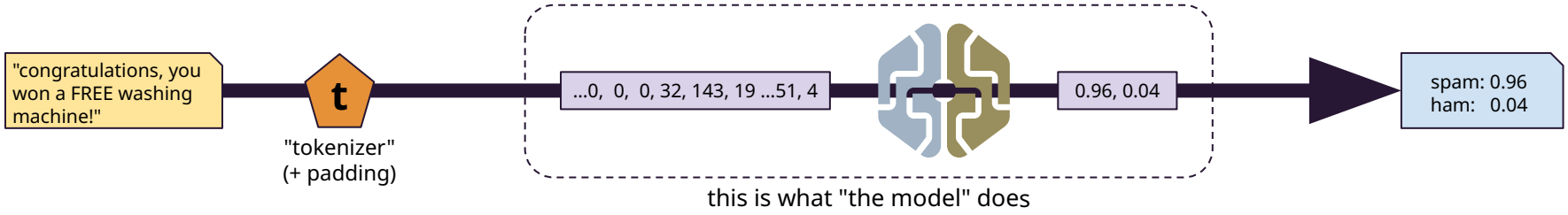
Logistic Regressions



Supervised learning



A closer look: numeric encoding



Prepare the dataset before training

Labeled training data

X	Y
Go until jurong poi	ham
Ok lar... Joking wif	ham
Free entry in 2 a wk	spam
U dun say so early h	ham
Nah I don't think he	ham
FreeMsg Hey there da	spam

...0, 0, 0, 32, 143, 19 ... 51, 4
...0, 0, 0, 0, 41, 4 ... 65, 10
...0, 0, 0, 0, 0, 21 ... 84, 5
...0, 0, 0, 0, 81, 205 ... 3, 51
...0, 0, 0, 7, 56, 4 ... 15, 75
...0, 0, 0, 0, 0, 194 ... 2, 65

t

1, 0
1, 0
0, 1
1, 0
1, 0
0, 1

Split "train" / "test"

train

...0, 0, 0, 32, 143, 19 ... 51, 4
...0, 0, 0, 0, 41, 4 ... 65, 10
...0, 0, 0, 0, 0, 21 ... 84, 5
...0, 0, 0, 0, 81, 205 ... 3, 51
...0, 0, 0, 7, 56, 4 ... 15, 75

test

...0, 0, 0, 0, 0, 194 ... 2, 65
... ...

The antispam model architecture

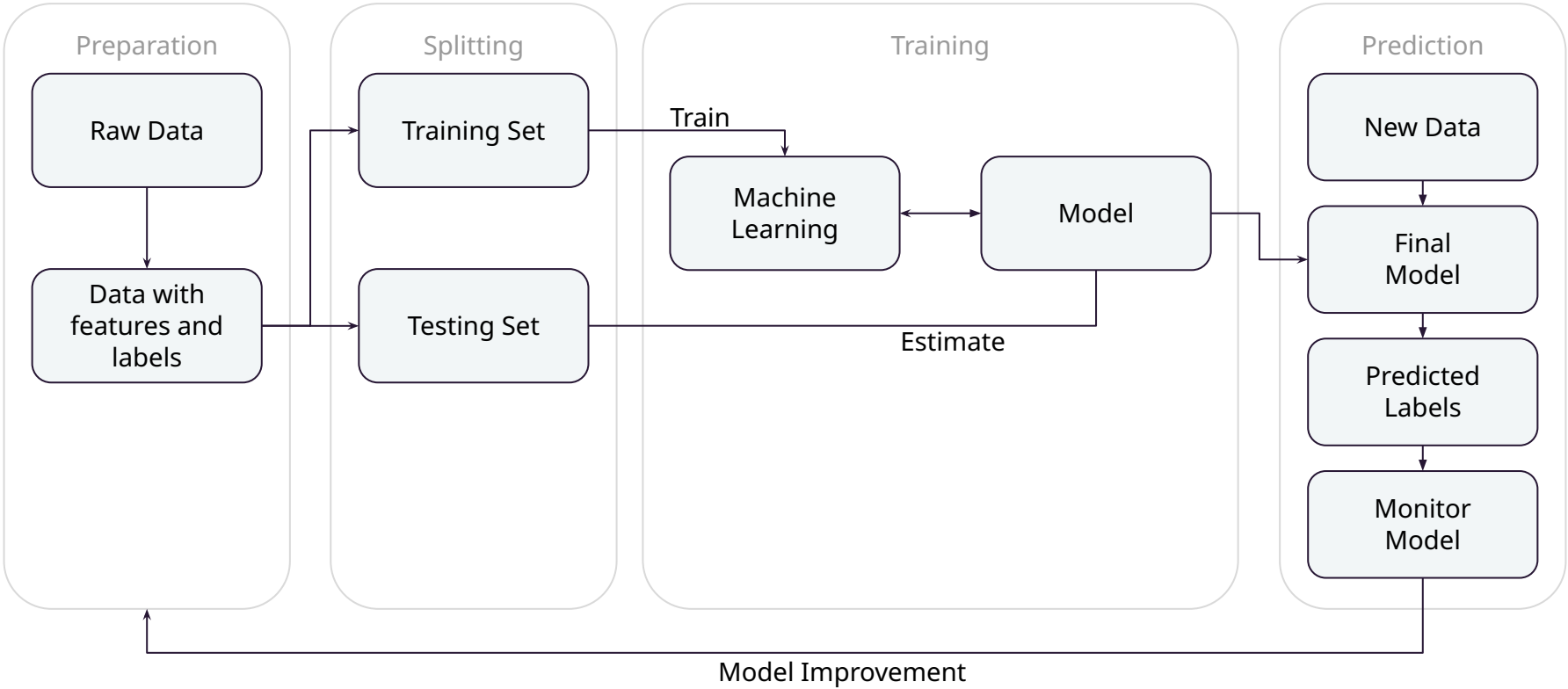
```
model = Sequential()
model.add(Embedding(MaxNumWords, embedDim, i
model.add(SpatialDropout1D(0.4))
model.add(LSTM(LstmOut, dropout=0.3, recurre
model.add(Dense(2, activation='softmax'))
model.compile(loss='categorical_crossentropy
```

A stack of layers:

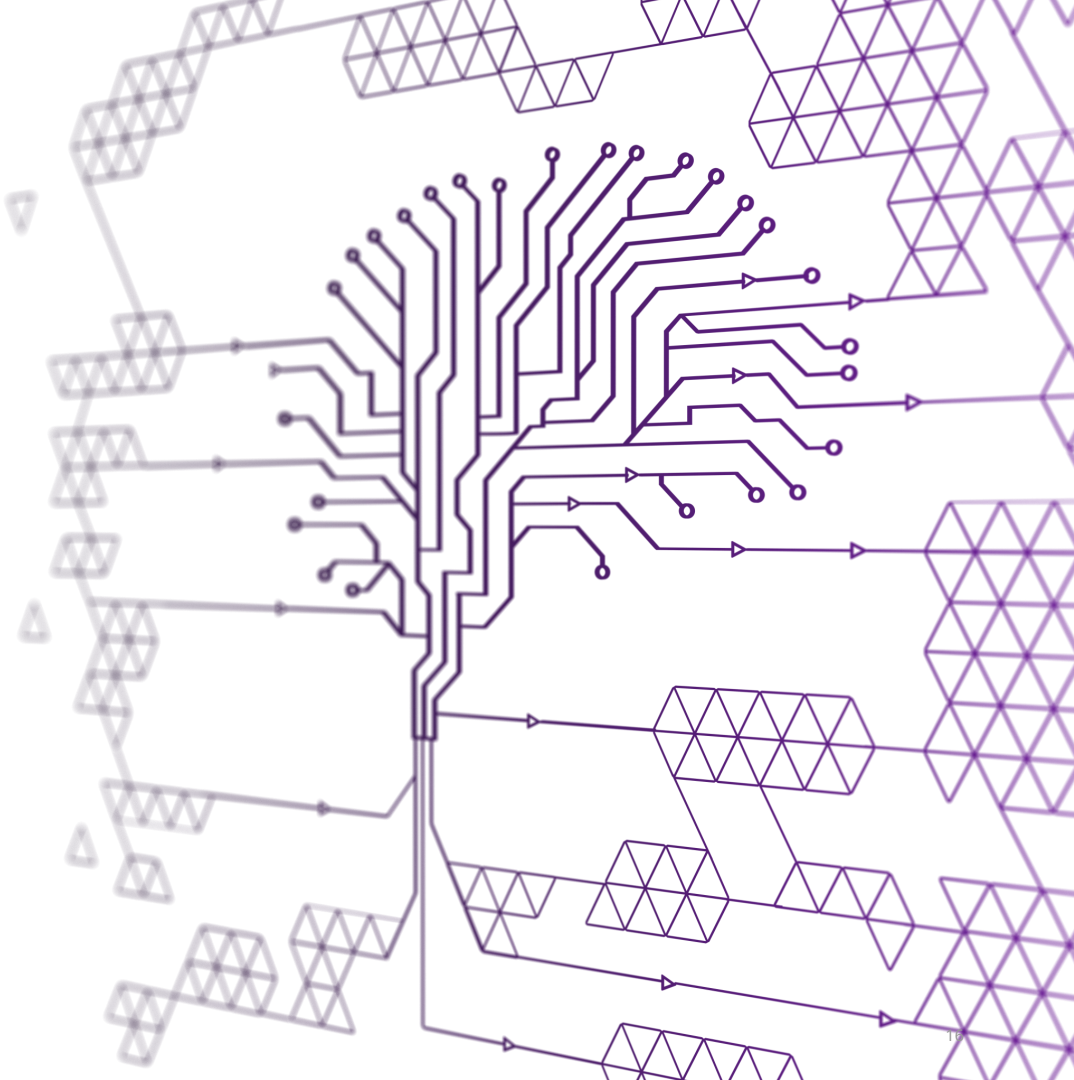
- translate an input number into a vector
- randomly disables pieces of network to enhance training ("dropout")
- the LSTM (recurrent within the layer)
- reducing to a 2-option output (spam/ham)

"finalize" the model: ready to train!

The AI



Database



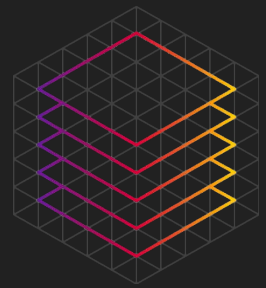
Database-as-a-Service powered by Apache Cassandra

DATASTAX

ASTRA DB

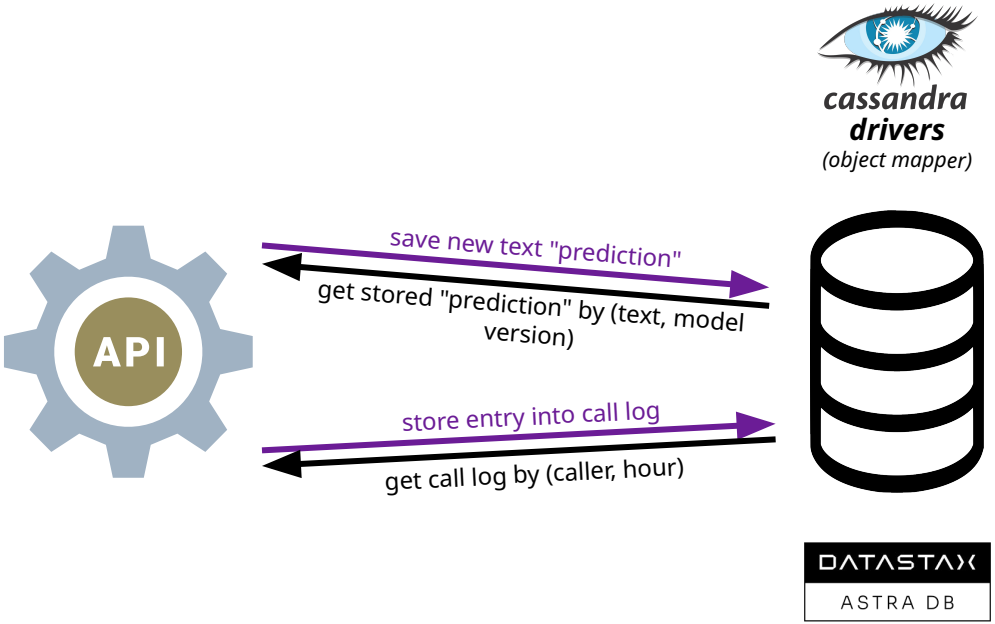
Sign up for free

www.datastax.com/products/datastax-astra



- Build in any language
- Global Scale
- Zero Operations
- Zero lock-in
- Enterprise security
- Extensibility

DB Query patterns

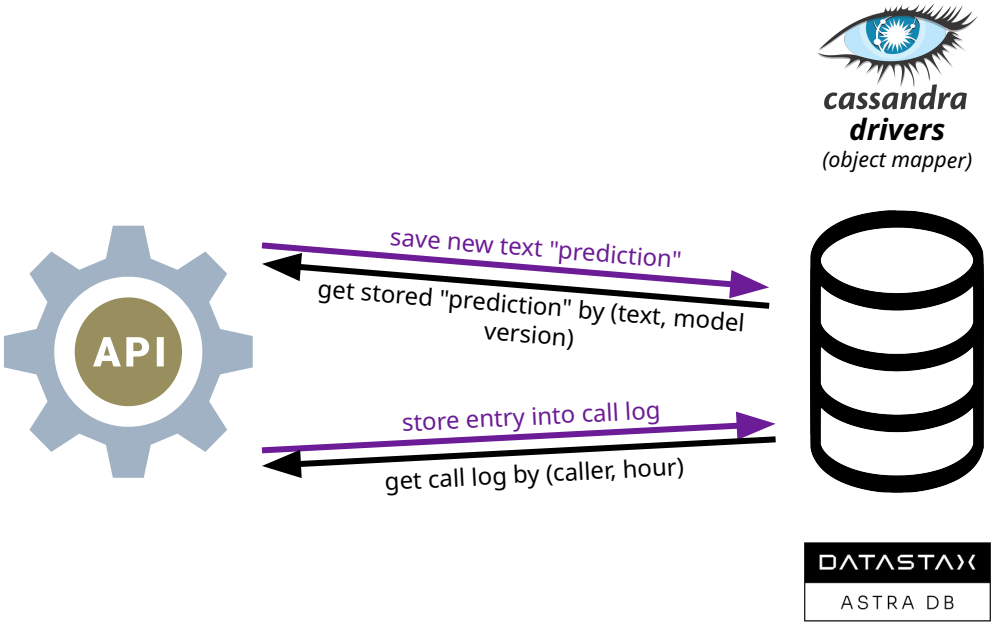


Data modeling, the Astra way:

"design the table after the query"

(also: tables are partitioned!)

DB Query patterns



spam_cache_items		
model_version	**text**	**K**
input	**text**	**K**
confidence	float	
prediction_map	map<text,float>	
result	text	
stored_at	timeuuid	

spam_calls_per_caller		
caller_id	**text**	**K**
called_hour	**timestamp**	**K**
called_at	**timeuuid**	**C↑**
input_text	text	

"partition key" (rows stored together)

"primary key" (row uniqueness)

"Chebotko diagrams"

Real-Time Data Processing (Feature Engineering)

Table Row Examples

version	input	confidence	prediction_map	result	stored_at
v1	Combing the white hair of the waves blown back	0.955171	{'ham': 0.955171, 'spam': 0.044829}	ham	08f849b6-f92a-
v1	We have lingered in the chambers of the sea	0.992575	{'ham': 0.992575, 'spam': 0.007425}	ham	090de000-f92a-
v1	Till human voices wake us, and we drown.	0.988767	{'ham': 0.988767, 'spam': 0.011233}	ham	092371b8-f92a-
v1	Click TO WIN a FREE CAR	0.739895	{'ham': 0.260105, 'spam': 0.739895}	spam	5d957634-f929-
v1	By sea-girls wreathed with seaweed red and brown	0.912248	{'ham': 0.912248, 'spam': 0.087752}	ham	e9424acc-f929-
v1	When the wind blows the water white and black.	0.979708	{'ham': 0.979708, 'spam': 0.020292}	ham	d4647bca-f929-
v1	I have seen them riding seaward on the waves	0.977917	{'ham': 0.977917, 'spam': 0.022083}	ham	d44dafda-f929-

spam_cache_items

model_version	**text**	**K**
input	**text**	**K**
confidence	float	
prediction_map	map<text,float>	
result	text	
stored_at	timeuuid	

caller_id	called_hour	called_at	input
127.0.0.1	2023-05-25 12:00:00.000000+0000	77049492-fafa-11ed-8a10-a2ab11a8d5af	Click TO WIN
192.168.150.76	2023-05-23 05:00:00.000000+0000	d42292e6-f929-11ed-a192-1a3838142467	I have seen them riding seaward o
192.168.150.76	2023-05-23 05:00:00.000000+0000	d438241c-f929-11ed-a192-1a3838142467	When the wind blows the water white
192.168.150.76	2023-05-23 05:00:00.000000+0000	e901c97a-f929-11ed-a192-1a3838142467	I have seen them riding seaward o
192.168.150.76	2023-05-23 05:00:00.000000+0000	e9174da4-f929-11ed-a192-1a3838142467	When the wind blows the water white
192.168.150.76	2023-05-23 05:00:00.000000+0000	e92cc8dc-f929-11ed-a192-1a3838142467	By sea-girls wreathed with seaweed re
192.168.150.76	2023-05-23 05:00:00.000000+0000	08771a08-f92a-11ed-a192-1a3838142467	I have seen them riding seaward o

spam_calls_per_caller

caller_id	**text**	**K**
called_hour	**timestamp**	**K**
called_at	**timeuuid**	**C↑**
input_text	text	

"partition key" (rows stored together)

"primary key" (row uniqueness)

Hands On Time

Tools

- Nothing to Install!
- GitHub repository: [bit.ly/irt-ai-as-an-api](bit.ly/irt-ai-as-an-api)

GitHub — Source code + Exercises + Slides

Gitpod — Cloud Development Environment

DATASTAX ASTRA DB — Database

FastAPI — API

Know your tools

- Python
- Jupyter
- TensorFlow
- Keras
- FastAPI
- Astra DB

Lab Steps

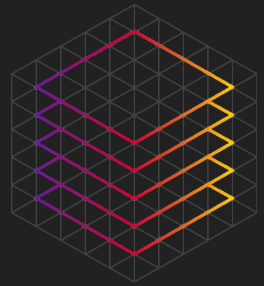
1. Initialise GitPod
2. Create a database in Astra DB
3. Inspect the Dataset
4. Train the Model in Jupyter
5. Expose the Model as an API
6. Use the API
7. Inspect the database

DATASTAX

ASTRA DB

Sign up for free

www.datastax.com/products/datastax-astra



- Build in any language
- Global Scale
- Zero Operations
- Zero lock-in
- Enterprise security
- Extensibility

Available from aws marketplace

DATASTAX

HANDS-ON WORKSHOP | MELBOURNE, AU

IRT AI/ML – Out of the Lab, Into Production

20 July, Thursday | 5pm - 7pm | WeWork Office, Level 22, 120 Spencer St

Hilton Rosenfeld
Data Architect, DataStax

REGISTER NOW

DATASTAX

Thank You


DATASTAX