Building a real-time image classification web app with Python and MLDB.ai

François Maillet
Head of AI

mldb.ai

Toronto, November 2016

PYCON
CANADA
2016

Agenda

- Intros
- High level overview of MLDB
- Building blocks
- Training an image classifier in a Notebook
- Creating an image classification web app with a plugin

About me

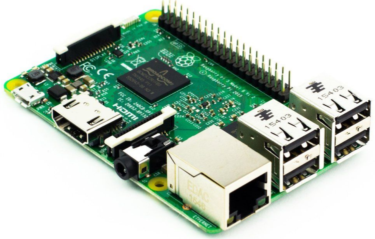
- **Head of AI at mldb.ai, previously at Datacratic**
- **Have been doing machine learning for almost 10 years**
- **Studied ML at UdeM: music recommendation, playlist generation**

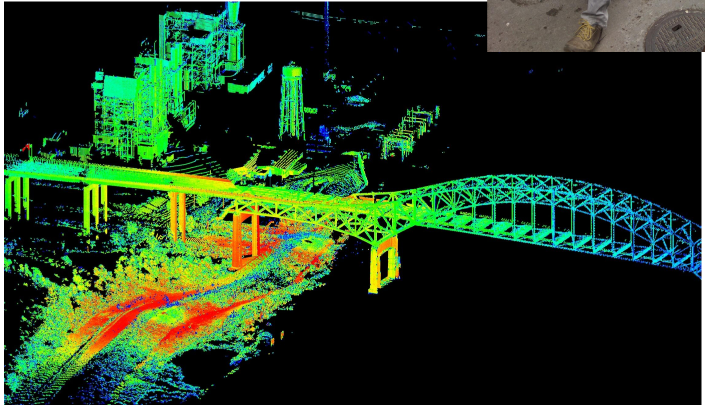
15 minutes of fame: Epic NHL goal celebration hack with a hue light show and real-time machine learning

blog.francoismaillet.com/epic-celebration

About mldb.ai

- Montréal machine learning company
- Team of 8; all ML or developers
- Building *The Machine Learning Database*
- Solved ML problems in many verticals
- In the middle of a pivot
  - ML + Lidar





What the heck is a Machine Learning Database?

```
SELECT what_happened() FROM historical_data
```

vs

```
SELECT what_will_happen() FROM new_data
```

Why did we build MLDB?

- End-to-end solution to solving production ML problems
  - Data ingestion
  - Model training and data science
  - Model deployment and real-time predictions

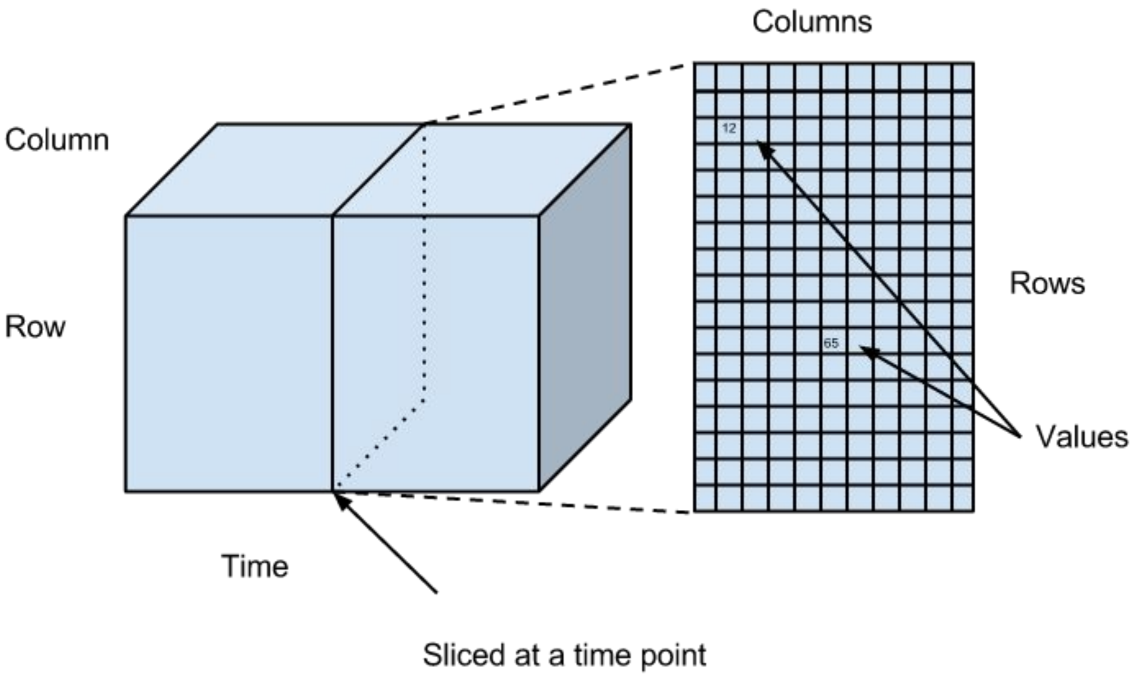
- Better workflow to do machine learning

Highlights

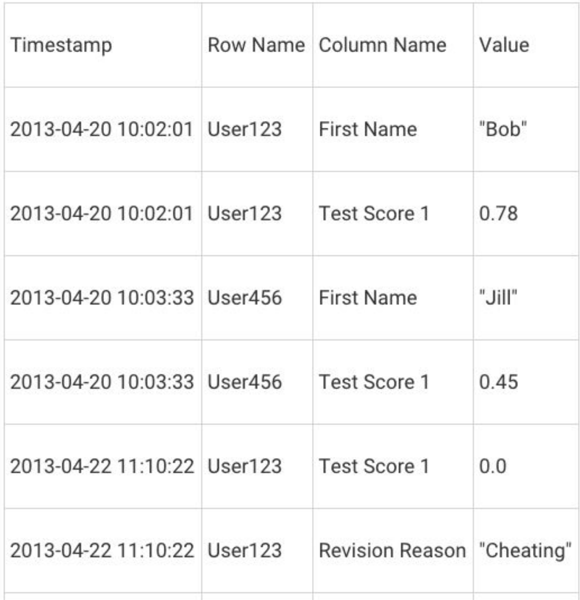
- **SQL data interface**
- **Written in c++. Extensible with plugins**
- **JSON-over-REST: talk to it from any language or platform**
- **Vertical scalability**
  - Process billions of data points on machines that cost $1/hr.
- **Train models faster than H20, Spark MLlib or scikit-learn**
- **Easy deployment**
  - As soon as a model is trained, it’s available as a REST endpoint
- **Runs on the Raspberry Pi**

Data Model

3-d sparse matrix: named rows, named columns, timestamps, values



Sliced at a time point



Timestamp	Row Name	Column Name	Value
2013-04-20 10:02:01	User123	First Name	"Bob"
2013-04-20 10:02:01	User123	Test Score 1	0.78
2013-04-20 10:03:33	User456	First Name	"Jill"
2013-04-20 10:03:33	User456	Test Score 1	0.45
2013-04-22 11:10:22	User123	Test Score 1	0.0
2013-04-22 11:10:22	User123	Revision Reason	"Cheating"

Data Model

- **Sparse and Flexible schema: can add rows/columns/timestamps**
- **Records time on every value, can slice on time**
- **Doubly-indexed: by row and column**
- **Matrix operations are easy and efficient**
    - Enables world-class SVD performance

Data Model

- **Exploits redundancy in typical dataset for extreme compression**
- **2-4 bytes per timestamped datapoint on real-world datasets**
- **1-2 billion data points can fit in 250GB of RAM:**
  - **100-million-row dataset with 150 non-null columns on average**
- **For reference, a 244GB AWS r3.8xlarge machine costs $1/hour**

SQL Extensions

- **Many columns:**

```
SELECT a* EXCLUDING (ab) as c* FROM x
```

- **Time dimension:**

```
SELECT * FROM x WHEN timestamp(a) < timestamp(b)
```

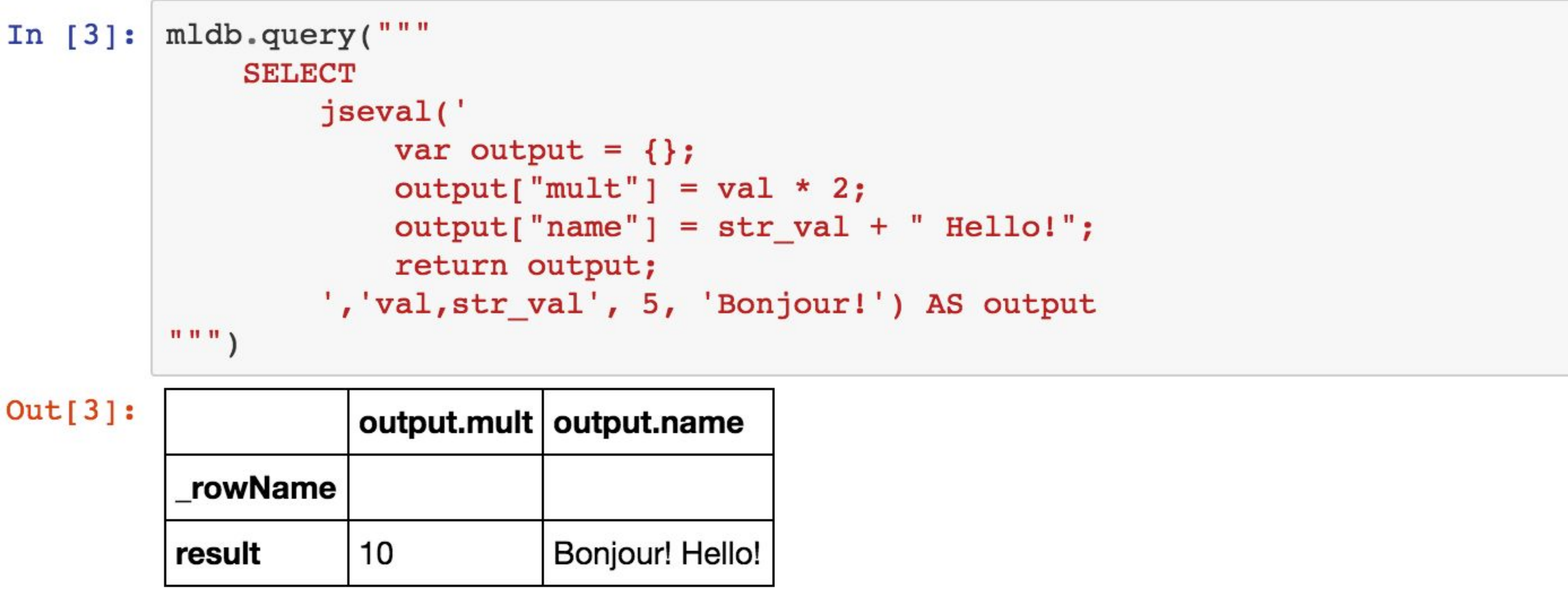
- **Row operations :**

```
SELECT horizontal_sum( {a*} ), tokenize( text ) FROM x
```

- **Fancy aggregates :**

```
SELECT pivot(movie_id, rating) FROM x GROUP BY user_id
```

SQL Extensions: jseval



```
In [3]: mldb.query("""
            SELECT
                jseval('
                    var output = {};
                    output["mult"] = val * 2;
                    output["name"] = str_val + " Hello!";
                    return output;
                ','val,str_val', 5, 'Bonjour!') AS output
        """)
```

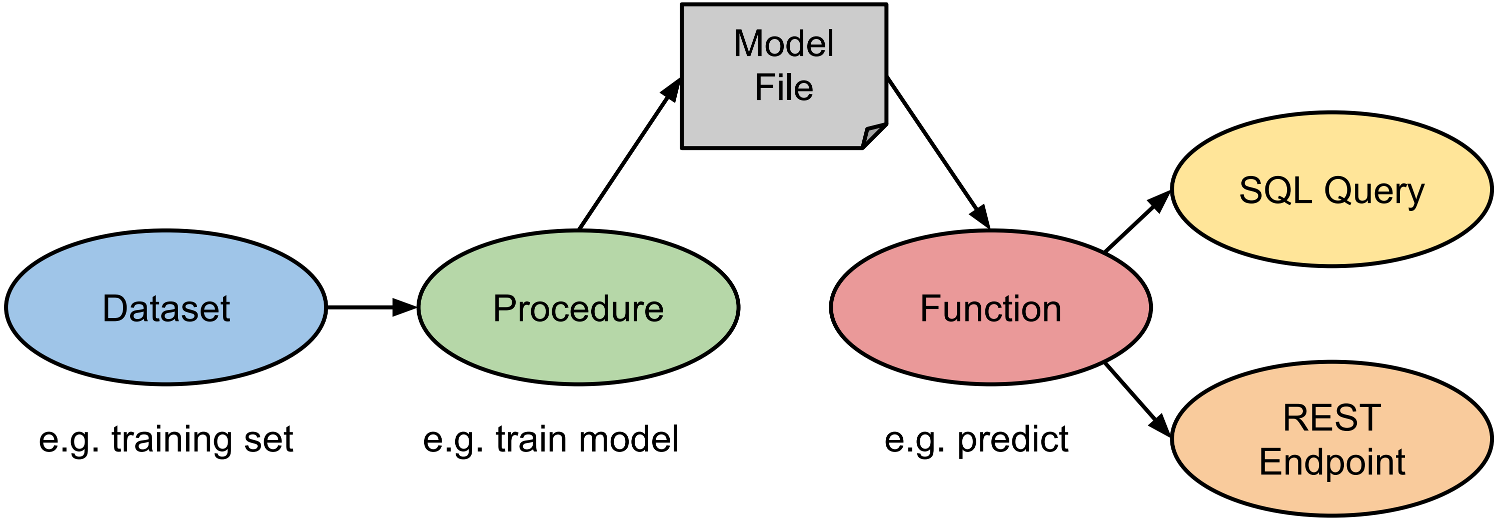
Out[3]:

	output.mult	output.name
_rowName		
result	10	Bonjour! Hello!

More in: Executing JavaScript Code Directly in SQL Queries Using the jseval Function Tutorial

Machine Learning with MLDB

mldb
.ai



Model File

Dataset

Procedure

Function

SQL Query

REST Endpoint

e.g. training set

e.g. train model

e.g. predict

Predictions via REST :

```
GET /v1/functions/my_predictor/application?input={features:<new_data>}
```

or via SQL :

```
SELECT my_predictor( {features: {*} ) FROM new_data
```

Built-in Algorithms

- **Efficient multithreaded implementations:**
  - ***Classifiers*: GLM, Decision Trees, SVM, Neural Nets, k-NN, Naive Bayes**
  - ***Deep learning:* TensorFlow integration**
  - ***Ensembles*: Bagging, Boosting**
  - ***Calibration*: Probabilizer**
  - ***Insight*: feature importance for all classifiers including ensembles**
  - ***Clustering*: k-Means**
  - ***Dimensionality reduction*: Singular Value Decomposition (SVD)**
  - ***Visualization*: t-SNE**
  - ***Feature extraction*: bag-of-words, feature hashing, word2vec, stats tables**

Getting Started with MLDB

- **Open-source!**
- **Free trial of all features at http://mldb.ai/ with cool demos**
- **Runs on Linux with Docker or AWS with an AMI**
- **Mac/Windows support via VirtualBox OVA**

Let’s start exploring!

On to the hub: mldb.ai

Training an image classifier in a Notebook

Transfer learning with Inception

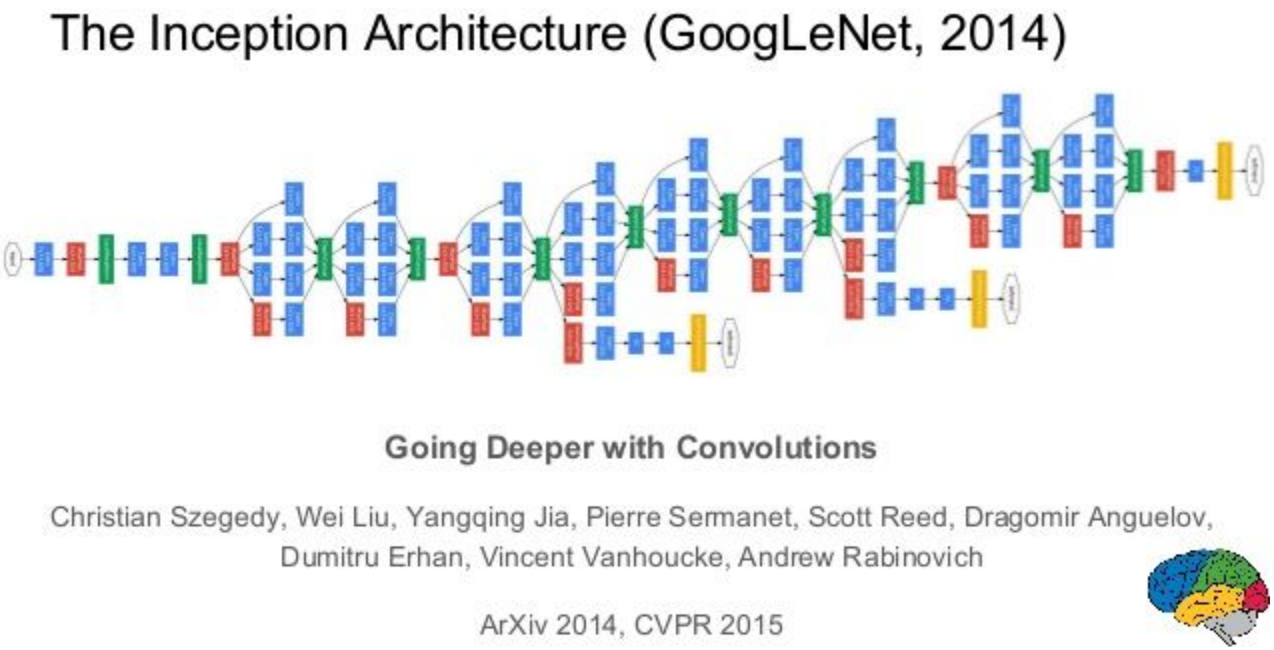
- Deep convolutional neural networks are really powerful
  - Require lots of training data, lots of GPUs

- Transfer learning: take a model that was trained on one task and use it on another task

- We use the Inception-v3 model that was trained on the ImageNet task
  - 1000 labels (ex: badger, freight car, cheeseburger)
  - Released as a trained TensorFlow graph

- More info: *Tensorflow Image Recognition Tutorial*

Transfer learning with Inception



The Inception Architecture (GoogLeNet, 2014)

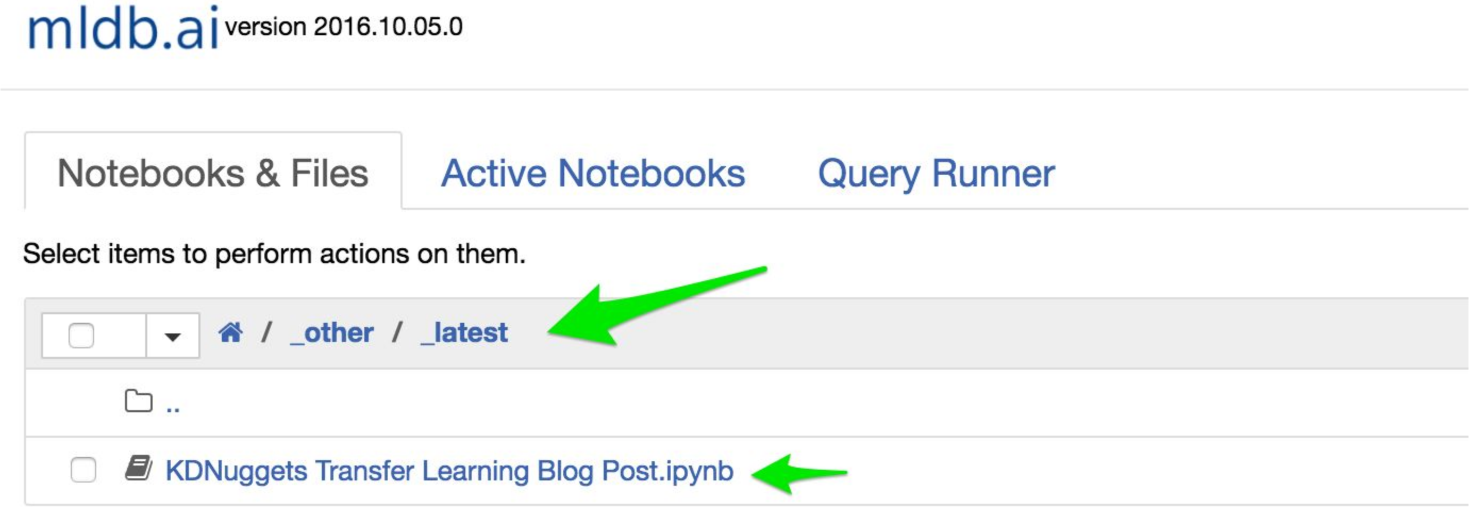
Going Deeper with Convolutions

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich

ArXiv 2014, CVPR 2015

Let’s start exploring!

Let’s train a model!



mldb.ai version 2016.10.05.0

Select items to perform actions on them.

/ _other / _latest

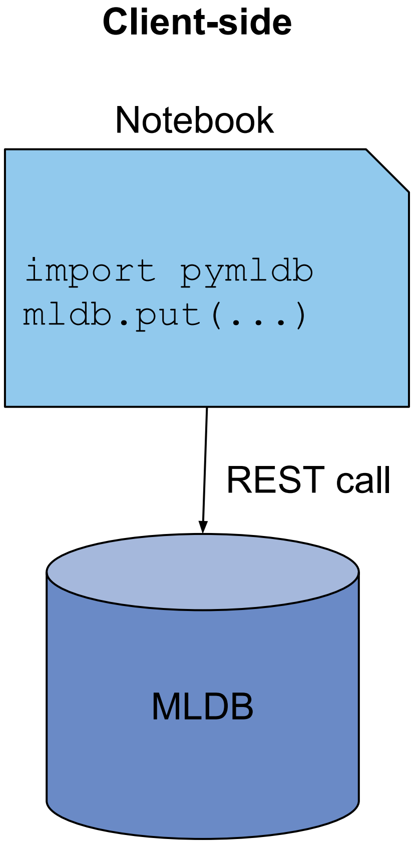
..

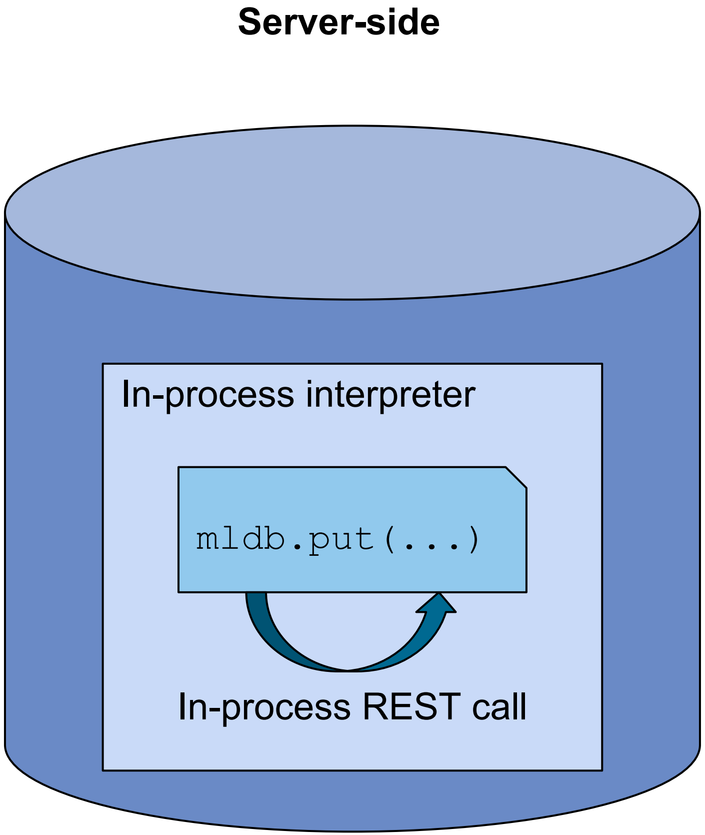
□ KDNuggets Transfer Learning Blog Post.ipynb

Creating a web app

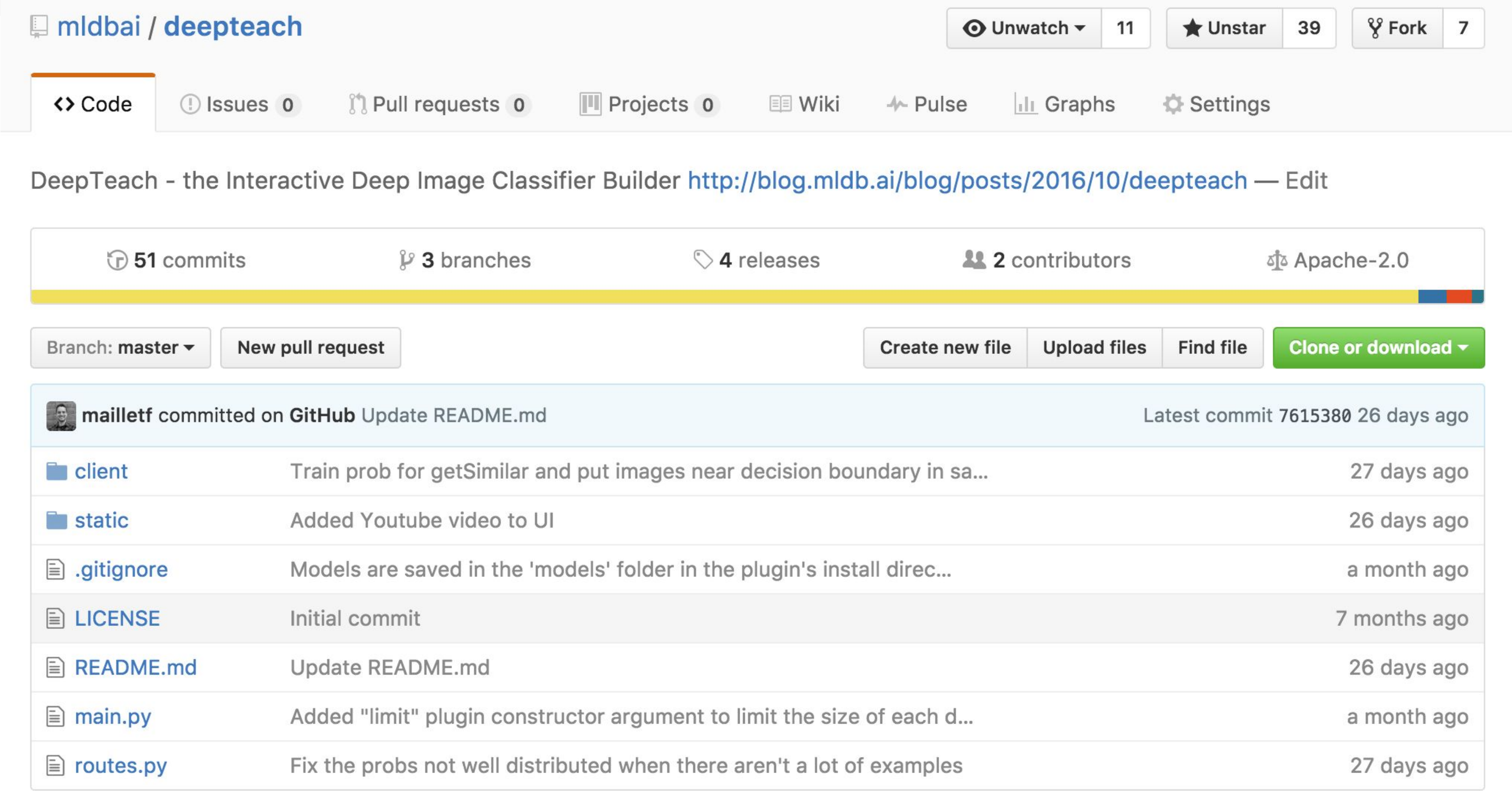
with a plugin

Client vs Server-side Python API





Anatomy of a Python Plugin



DeepTeach - the Interactive Deep Image Classifier Builder http://blog.mldb.ai/blog/posts/2016/10/deepteach — Edit

mailletf committed on GitHub Update README.md — Latest commit 7615380 26 days ago

client	Train prob for getSimilar and put images near decision boundary in sa...	27 days ago
static	Added Youtube video to UI	26 days ago
.gitignore	Models are saved in the 'models' folder in the plugin's install direc...	a month ago
LICENSE	Initial commit	7 months ago
README.md	Update README.md	26 days ago
main.py	Added "limit" plugin constructor argument to limit the size of each d...	a month ago
routes.py	Fix the probs not well distributed when there aren't a lot of examples	27 days ago

Server-side Python API

- `mldb.log(message)` basic logging facility
  - View output at the `http://host:port/logs/mldb` route

- `mldb.plugin.serve_static_folder(route, dir)` serve up static content under `dir` on the given plugin route `GET /v1/plugins/<id>/routes/<route>`

- HTTP calls, when using wrapper: `mldb = mldb_wrapper.wrap(mldb)`
  - `mldb.get(url)`
  - `mldb.put(url, payload)`
  - `mldb.post(url, payload)`
  - `mldb.delete(url)`

Server-side Python API: Handling a custom route

- Calling `/v1/plugins/<id>/routes/<route>` triggers the execution of `routes.py`

- Must handle the `(verb, remaining)` tuple, available in the `mldb.plugin.rest_params` object.

- If it represents a valid route, the `mldb.plugin.set_return` function must be called with a non-null body, which will be returned in the response.

- If the function is not called or called with a null body, the HTTP response code will be 404.

Server-side Python API: Handling a custom route

`mldb.plugin.rest_params`: object available within `routes.py` which represents an HTTP REST call. It has the following fields and methods:

- `verb`, `remaining`, `rest_params`: route and query-string details such that for

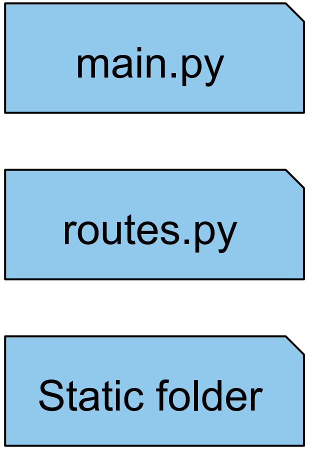
  `GET /v1/plugins/X/routes/hello?who=you`

  - `verb` = `GET`
  - `remaining` = `hello`
  - `rest_params` = `[['who', 'you'], ['yes','you']`

`mldb.plugin.set_return(body)`: available within `routes.py`, function called to write to HTTP response body and HTTP return code

Plugin development workflow

- Work off a plugin folder placed in `mldb_data`
- Plugin creation loads **main.py**
  - Reload if main.py is modified (ie new static route needs to be created)
- Each custom route call loads **routes.py**, taking into account any modification that was made
- Static contents is served as is on disk



Layout for our plugin

main.py

- Load trained model & inception function
- Create pipeline function
- Serve static folder containing UI

routes.py

- Handle custom routes call to run an SQL query and return the top label

Static folder

- Create index.html page allowing the user to run a prediction by calling one of 2 MLDB routes:
  - sql.expression function returning us the raw predictions
  - Custom python route that will run an SQL query and return the top label

Working version here:
https://github.com/mldbai/pyconca-2016

Let’s start building!

Let’s build the plugin!

Thanks!!

- Check out a more complex plugin using the same concepts:
  - DeepTeach: http://blog.mldb.ai/blog/posts/2016/10/deepteach
- Come talk to us on Gitter: https://gitter.im/mldbai/mldb
- Get started with MLDB: http://mldb.ai

François Maillet
@mailletf - francois@mldb.ai
blog.francoismaillet.com

mldb.ai