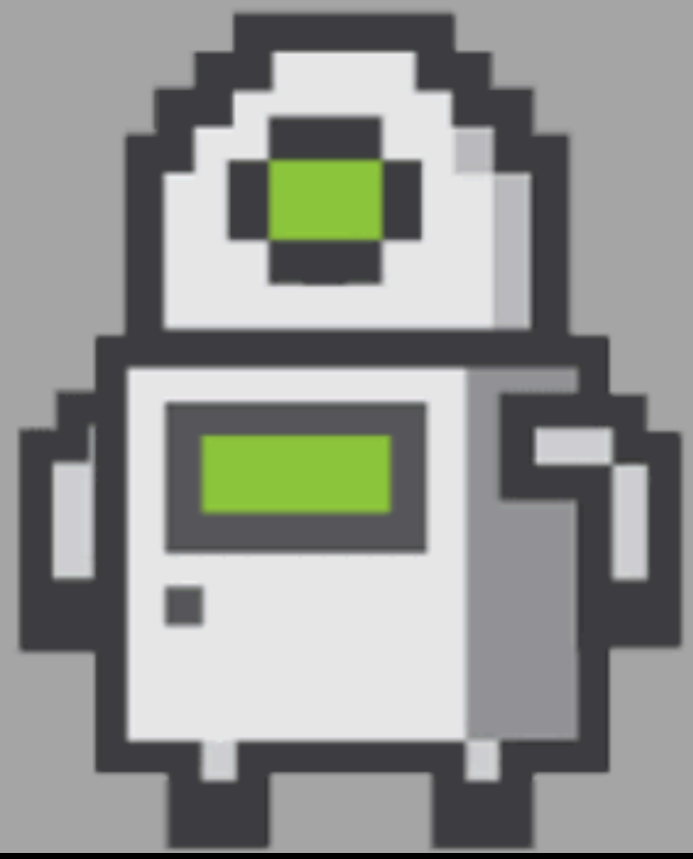
school of ai

codespace

Machine learning in iOS apps

WHAT IS POSSIBLE TODAY?

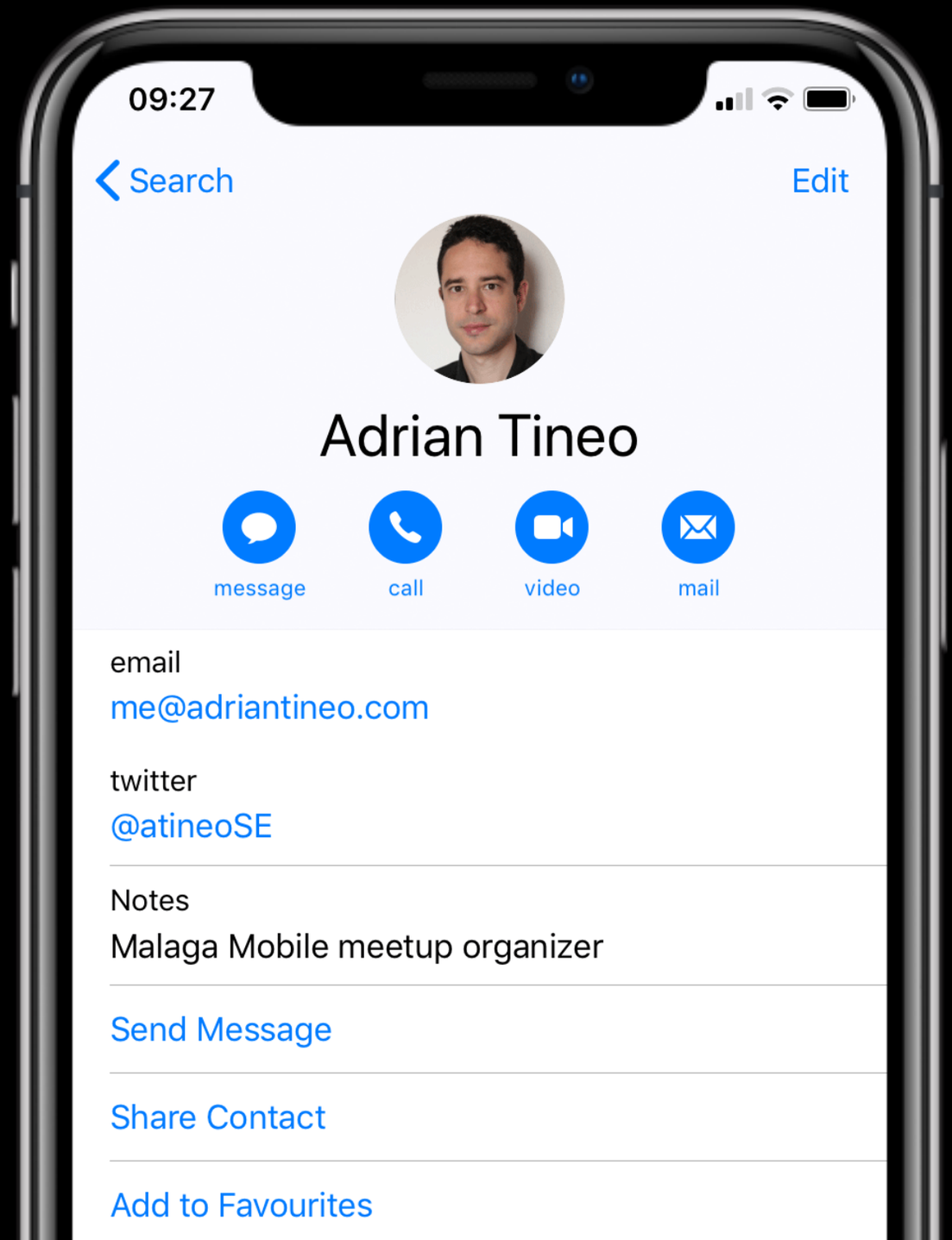


JULY 11

Codespace Academy 19:00

About me

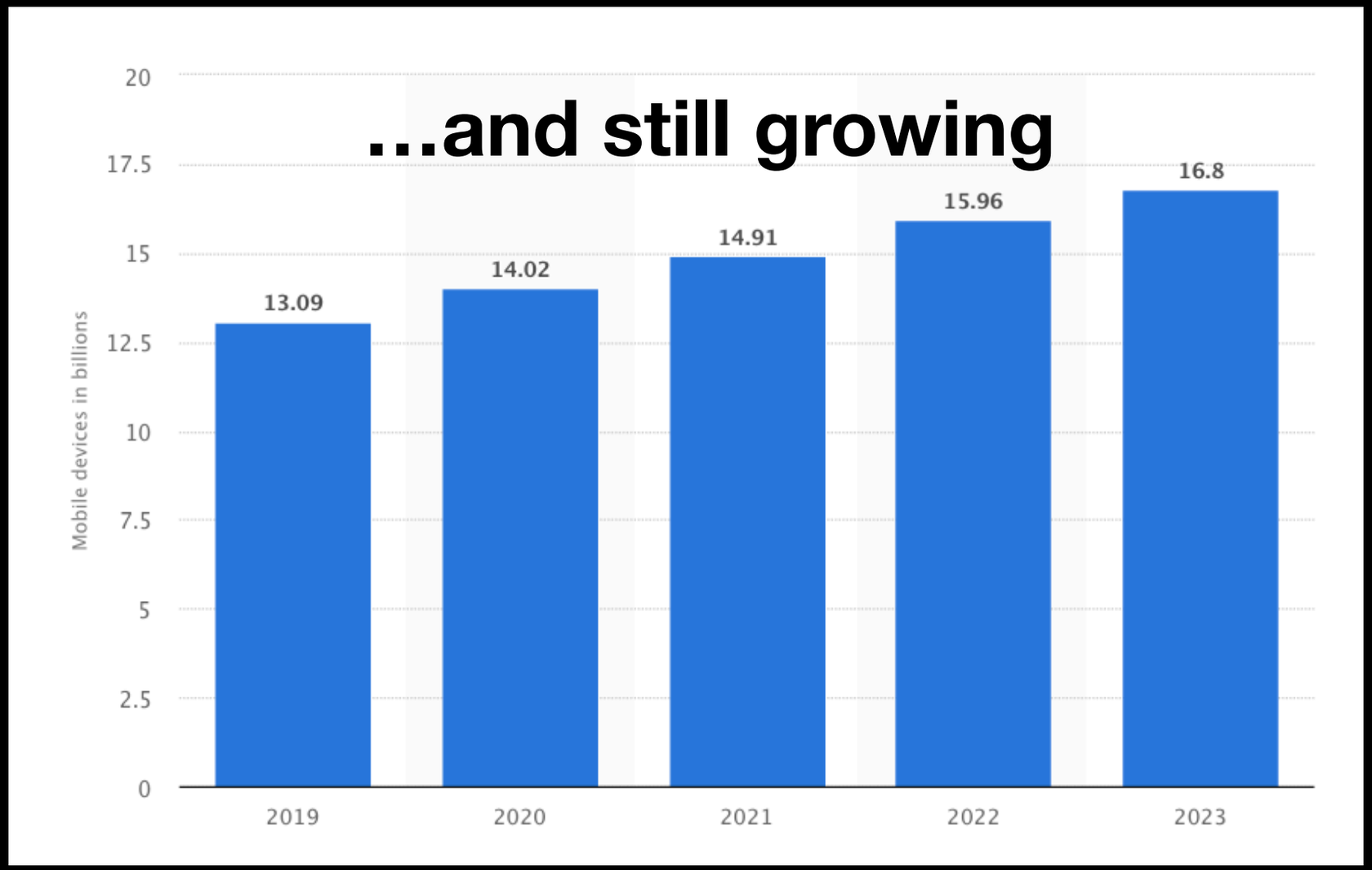
- Freelance iOS Consultant
- Ph.D. in CS
- Background:
  - research scientist
  - parallel computing
  - test automation



A mobile developer's perspective

Mobile is eating the world

2.71 billion smartphone users in the world in 2019 (1 out of 3 people owns a smartphone)



Sources:
https://techjury.net/stats-about/smartphone-usage/
https://www.statista.com/statistics/245501/multiple-mobile-device-ownership-worldwide/

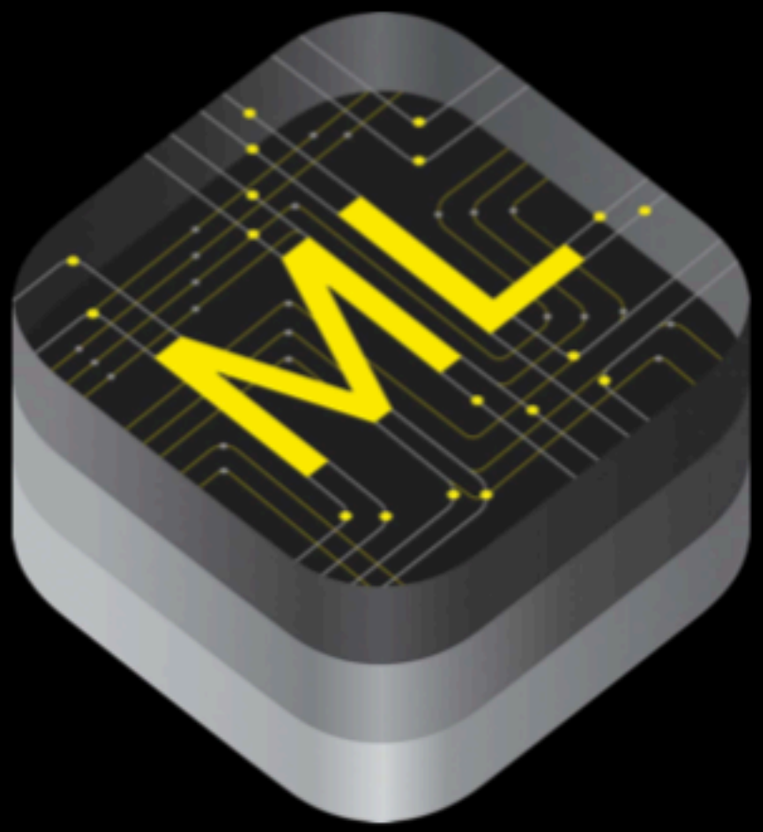
Mobile is eating the world

Photo by Yura Fresh on Unsplash

Integrative devices

Integrative devices

Core ML is announced at WWDC’17



Lots of possible applications

Sentiment Analysis

That was totally awesome Leo! → 😃

Handwriting Recognition

7 → 7

Translation

I love you mom → 사랑해 엄마

Scene Classification

 → Beach

Style Transfer

 → 

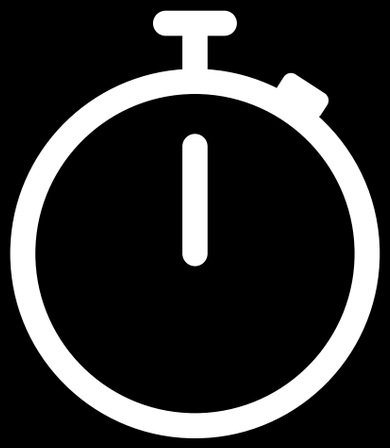
Music Tagging

 → Rock

Predicting Text

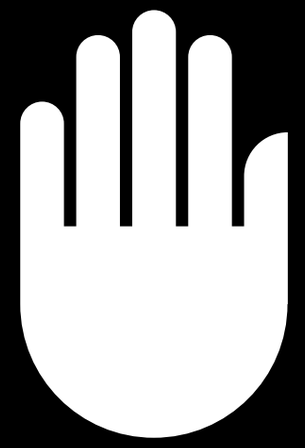
Do you know the way to → San Jose

Model runs on device



Performance

Server costs



Privacy

Initial workflow

Caffe

Keras

dmlc XGBoost

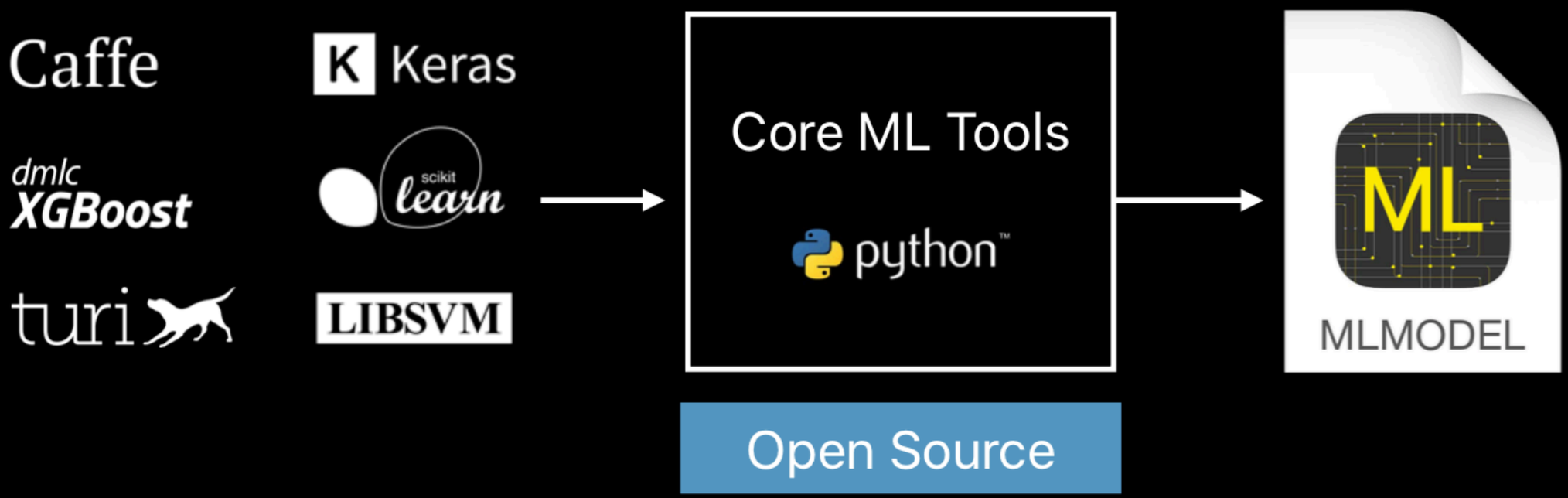
scikit learn

turi

LIBSVM

Convert

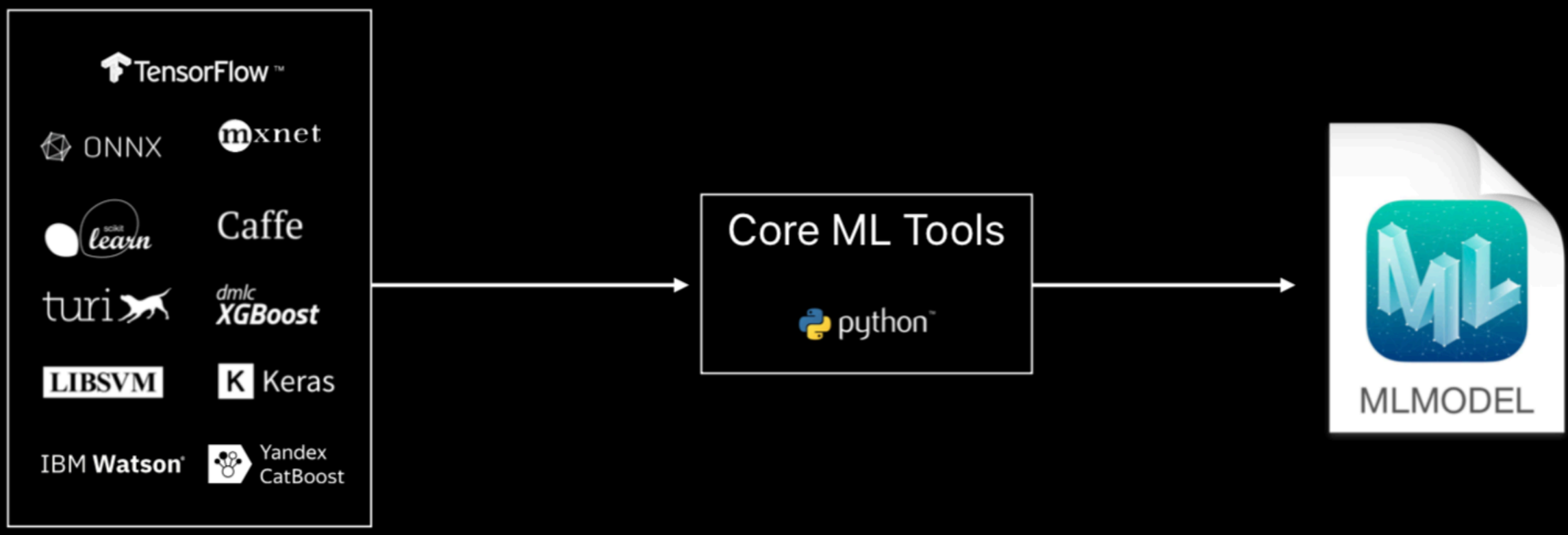
Model conversion tool



Data scientist anyone?

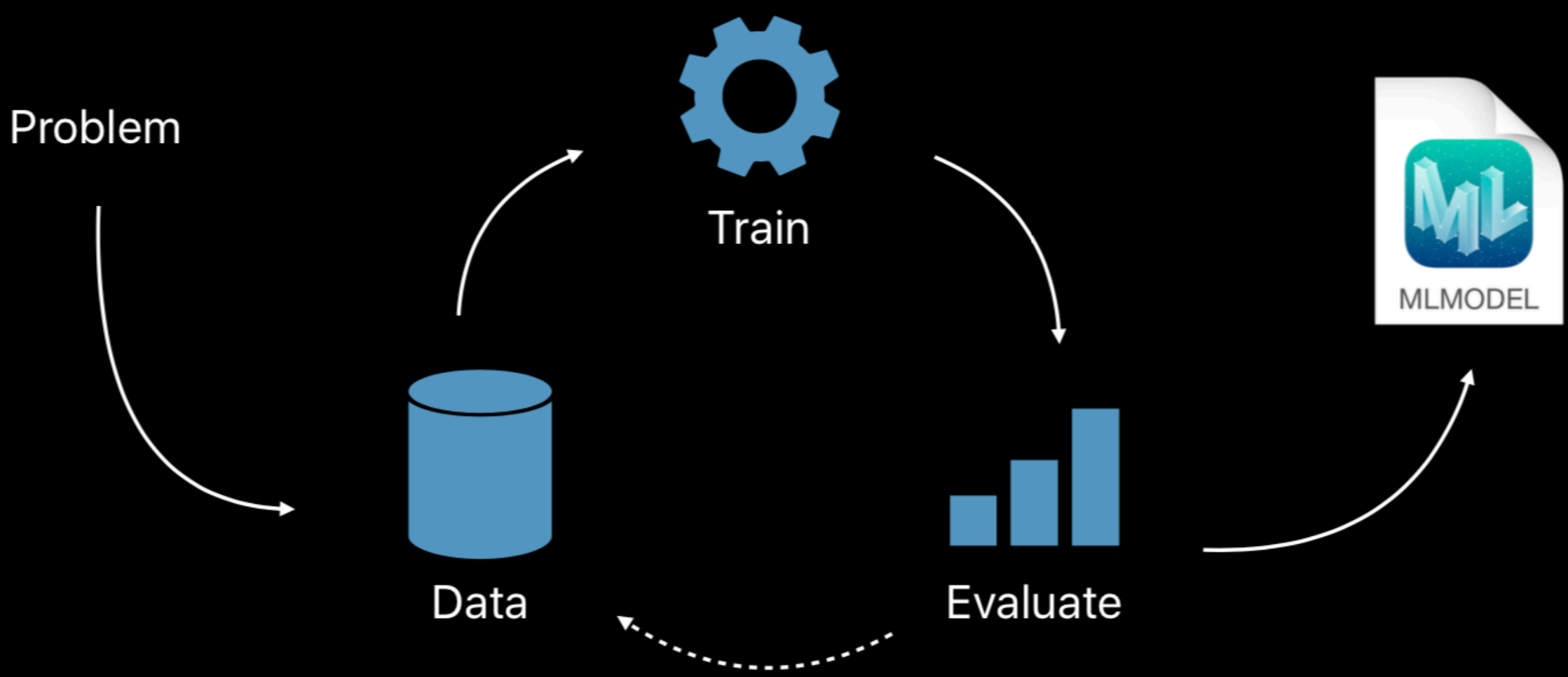
Fast-forward to WWDC’18

More formats are supported



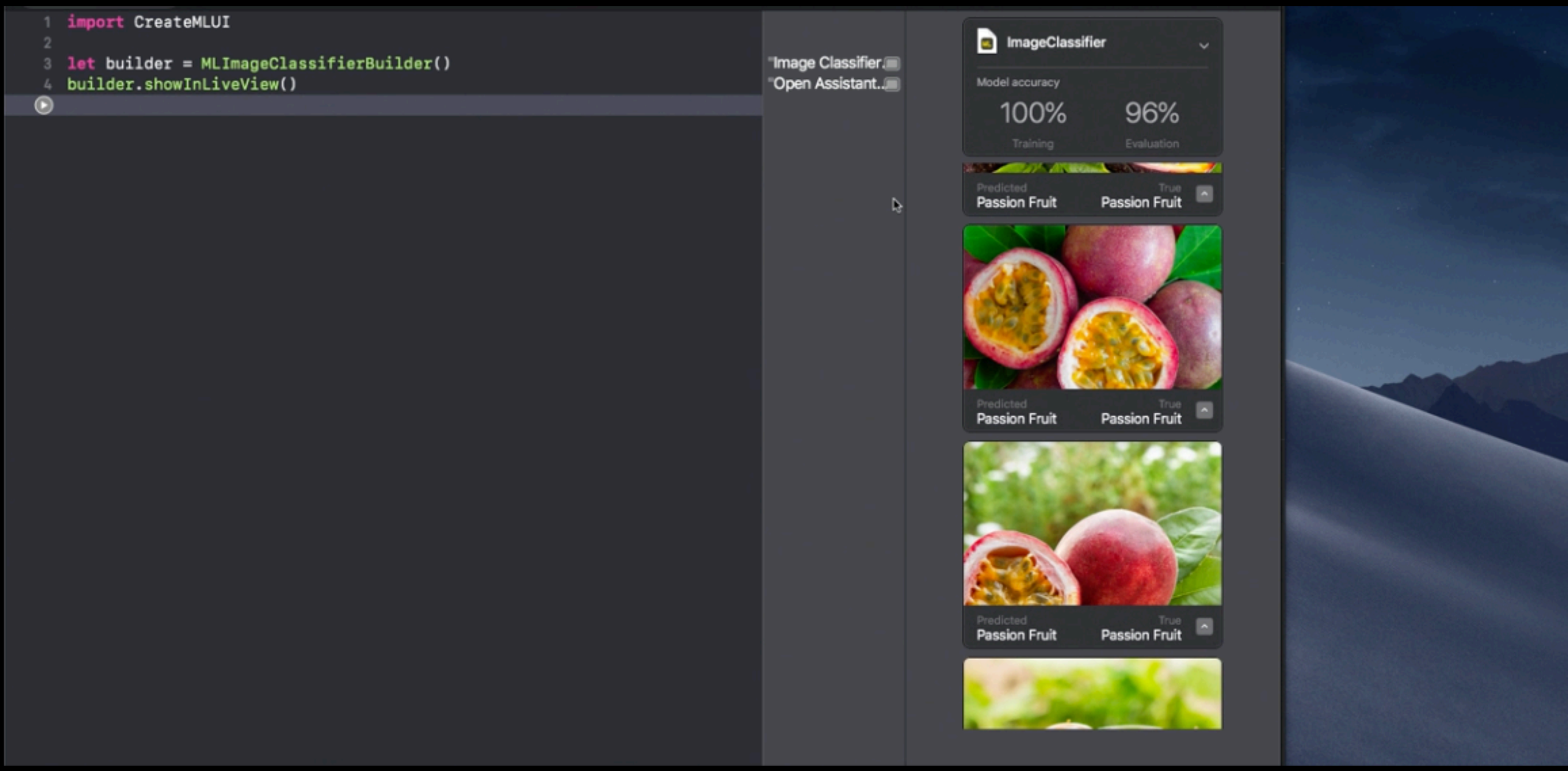
Create ML is announced

Alternative workflow



All within the Apple developer ecosystem (i.e. Xcode)

Train from a playground



Can I really play?

Three major cases supported

Images

Text

Tabular Data

Is it really so easy?

Create ML challenge

Does this really work on real problems or is it only good for toy demos?

Can we get real time performance?

For common cases

MobileNet

MobileNets are based on a streamlined architecture that have depth-wise separable convolutions to build lightweight, deep neural networks. Detects the dominant objects present in an image from a set of 1000 categories such as trees, animals, food, vehicles, people, and more.

View original model details ›

Download Core ML Model

SqueezeNet

Detects the dominant objects present in an image from a set of 1000 categories such as trees, animals, food, vehicles, people, and more. With an overall footprint of only 5 MB, SqueezeNet has a similar level of accuracy as AlexNet but with 50 times fewer parameters.

View original model details ›

Download Core ML Model

Places205-GoogLeNet

Detects the scene of an image from 205 categories such as an airport terminal, bedroom, forest, coast, and more.

View original model details ›

Download Core ML Model

ResNet50

Detects the dominant objects present in an image from a set of 1000 categories such as trees, animals, food, vehicles, people, and more.

View original model details ›

Download Core ML Model

Inception v3

Detects the dominant objects present in an image from a set of 1000 categories such as trees, animals, food, vehicles, people, and more.

View original model details ›

Download Core ML Model

VGG16

Detects the dominant objects present in an image from a set of 1000 categories such as trees, animals, food, vehicles, people, and more.

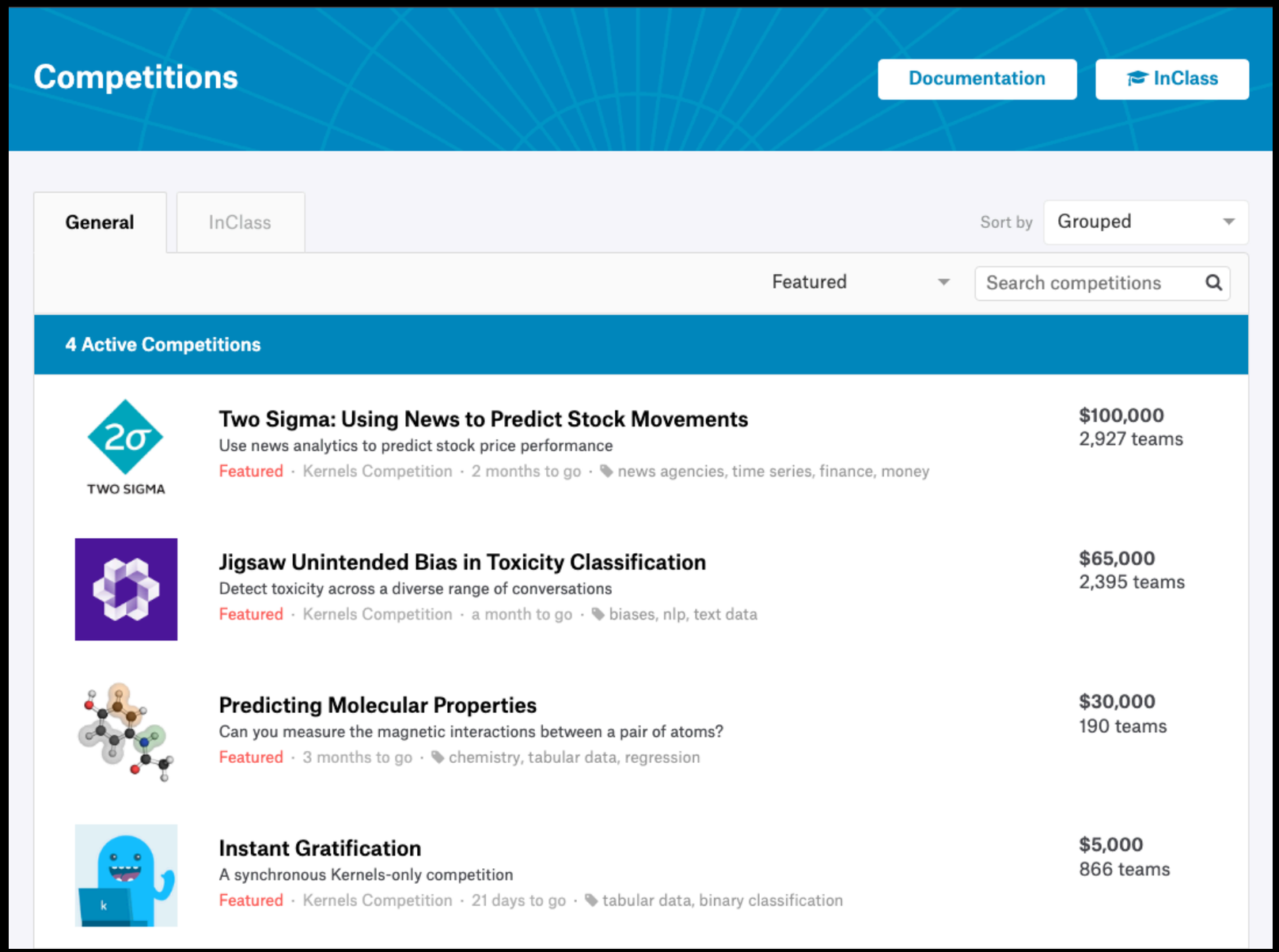
View original model details ›

Download Core ML Model

https://developer.apple.com/machine-learning/build-run-models/

How about some challenging problems?

kaggle



Three problems

kaggle

Classify distracted drivers

Images

Detect toxic comments

Text

Predict customer actions

Tabular Data

Direct approach

Multidisciplinary approach

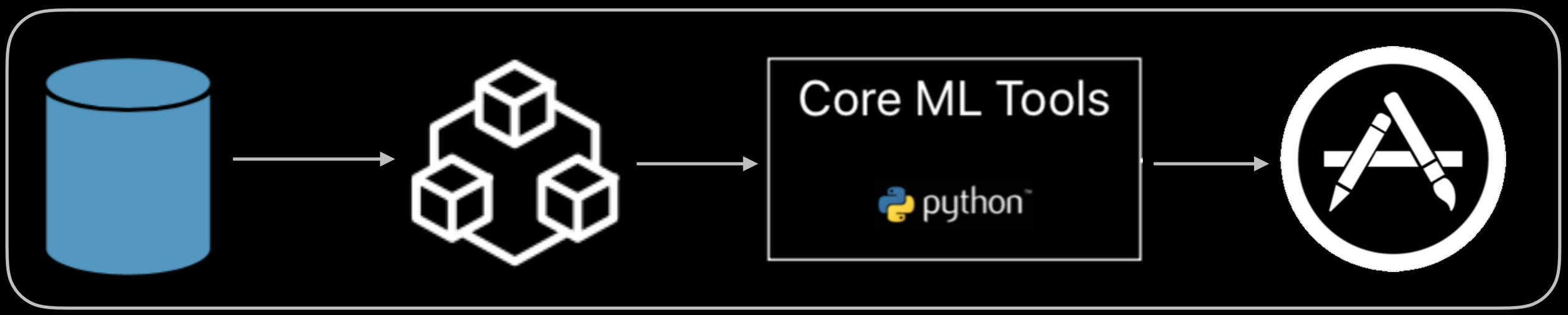


Image Classification

State Farm Distracted Driver Detection

Can computer vision spot distracted drivers?

$65,000 · 1,440 teams · 3 years ago

Overview | Data | Kernels | Discussion | Leaderboard | Rules | Team | My Submissions | Late Submission

Overview

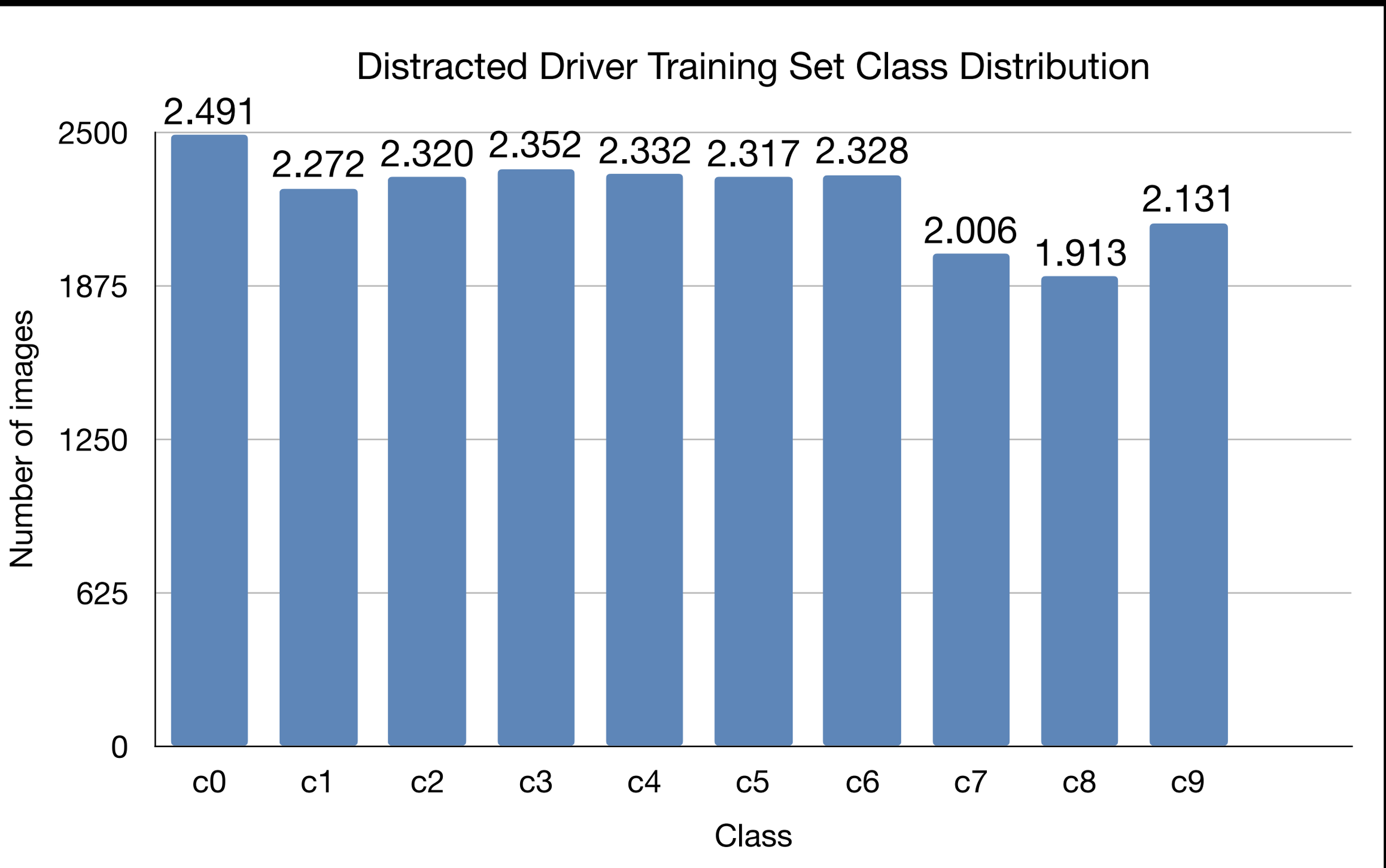
- Description
- Evaluation
- Prizes
- Timeline

We've all been there: a light turns green and the car in front of you doesn't budge. Or, a previously unremarkable vehicle suddenly slows and starts swerving from side-to-side.

When you pass the offending driver, what do you expect to see? You certainly aren't surprised when you spot a driver who is texting, seemingly enraptured by social media, or in a lively hand-held conversation on their phone.

Training data distribution

22,462 images



Distracted Driver Dataset

c0 - safe driving

c1 - texting right

c2 - talking right

c3 - texting left

c4 - talking left

c5 - radio

c6 - drinking

c7 - reach behind

c8 - grooming

c9 - talking

Create image classifier

```
// Define parameters for training
let parameters =
MLImageClassifier.ModelParameters(featureExtractor: .scenePrint(revision: 1),
                                  validationData: nil,
                                  maxIterations: 150,
                                  augmentationOptions: [])

// Train classifier
let classifier =  try? MLImageClassifier(trainingData: trainingData,
                                         parameters: parameters)
```

Optimal solution

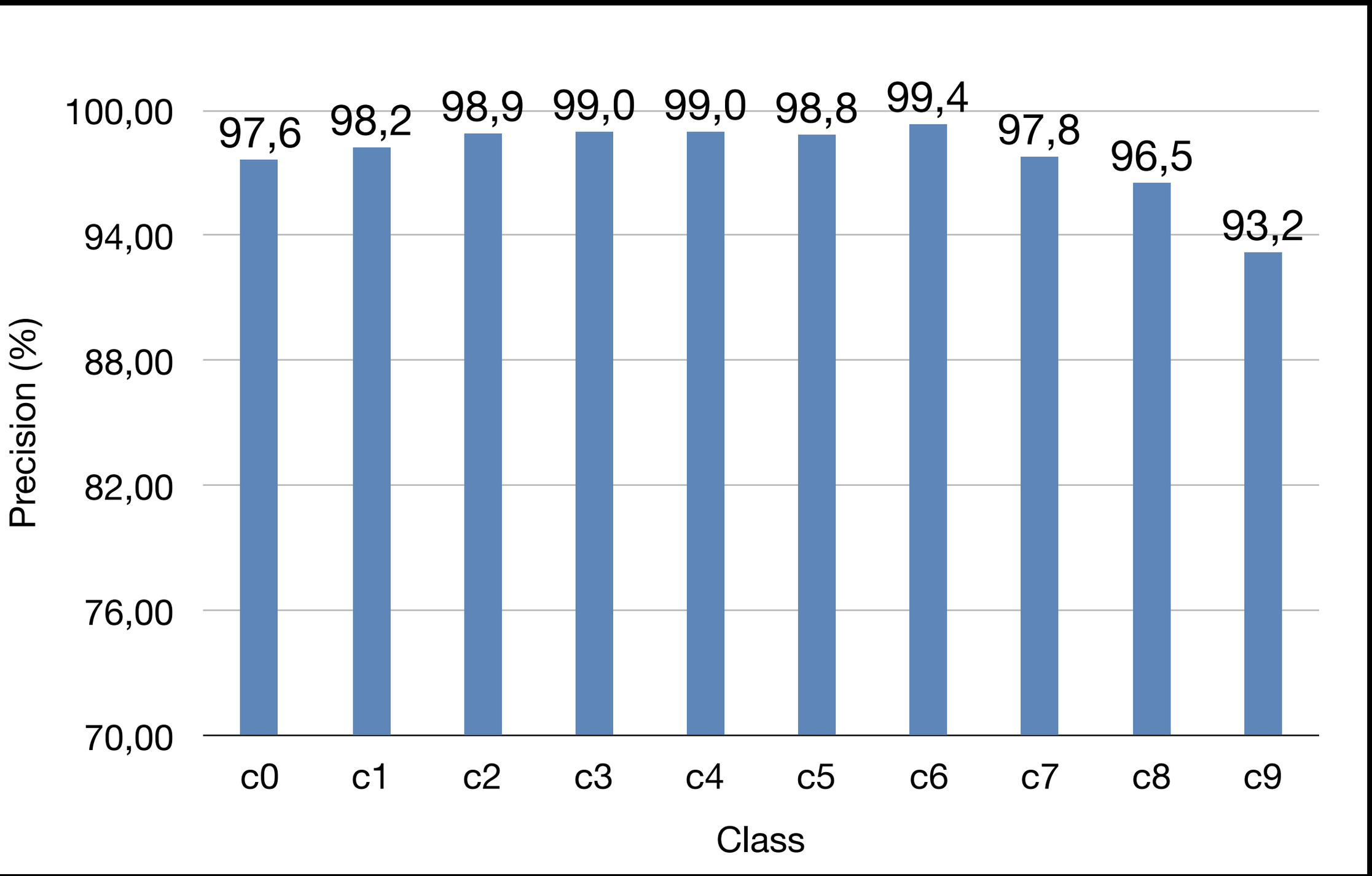
1000 it **85 min**

```
+-----------+--------------+--------------------+----------------------+
| Iteration | Elapsed Time | Training Accuracy | Validation Accuracy |
+-----------+--------------+--------------------+----------------------+
```

...

```
| 710       | 911.891504   | 1.000000           | 0.976035             |
| 715       | 917.239272   | 1.000000           | 0.976035             |
| 720       | 923.040021   | 1.000000           | 0.977124             |
| 725       | 928.810305   | 1.000000           | 0.977124             |
| 730       | 934.036429   | 1.000000           | 0.977124             |
| 735       | 941.578091   | 1.000000           | 0.977124             |
+-----------+--------------+--------------------+----------------------+
SUCCESS: Optimal solution found.
```

Classifier precision



Model size

178 Kb

Performance on device

A12

Neural engine
8 cores

Avg 33 images/second

A10

No dedicated HW
4 cores

Avg 8 images/second

Measured over ~19,000 images

Accuracy on device

Fails 909 out of 4’467 sample images (20%)

Accuracy on device

Model Evaluation Parameters

Name	Type	Flexibility	Description
▼ Inputs			
image	Image (Color 299 x 299)	299... x 299...	Input image to be classified
▼ Outputs			
classLabelProbs	Dictionary (String → Double)		Probability of each category
classLabel	String		Most likely image category

Accuracy on device

`request.imageCropAndScaleOption`

Dataset image (c7)
640x480

.centerCrop
299x299

.scaleFill
299x299

.scaleFit
299x299

Accuracy on device

- Several failed strategies
  - Change crop and scale option -> same result
  - Add flexible sizes to the model -> same result
  - Try to manipulate images manually -> crashes
  - Train the model with augmentation (cropped images) -> worse precision

Model size

Keras + TensorFlow™

```
def build_model():
    inputs = Input(shape=(config.img_width_adjust, config.img_height_adjust, 3), name="input")

    # Convolution 1
    conv1 = Conv2D(128, kernel_size=(3, 3), activation="relu", name="conv_1")(inputs)
    pool1 = MaxPooling2D(pool_size=(2, 2), name="pool_1")(conv1)

    # Convolution 2
    conv2 = Conv2D(64, kernel_size=(3, 3), activation="relu", name="conv_2")(pool1)
    pool2 = MaxPooling2D(pool_size=(2, 2), name="pool_2")(conv2)

    # Convolution 3
    conv3 = Conv2D(32, kernel_size=(3, 3), activation="relu", name="conv_3")(pool2)
    pool3 = MaxPooling2D(pool_size=(2, 2), name="pool_3")(conv3)

    # Convolution 4
    conv4 = Conv2D(16, kernel_size=(3, 3), activation="relu", name="conv_4")(pool3)
    pool4 = MaxPooling2D(pool_size=(2, 2), name="pool_4")(conv4)

    # Fully Connected Layer
    flatten = Flatten()(pool4)
    fc1 = Dense(1024, activation="relu", name="fc_1")(flatten)

    # output
    output = Dense(10, activation="softmax", name="softmax")(fc1)

    # finalize and compile
    model = Model(inputs=inputs, outputs=output)
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=["accuracy"])

    return model
```

Model size

Keras + TensorFlow™

8 hours training time

4 epochs

Accuracy: 95%

111.5 MB

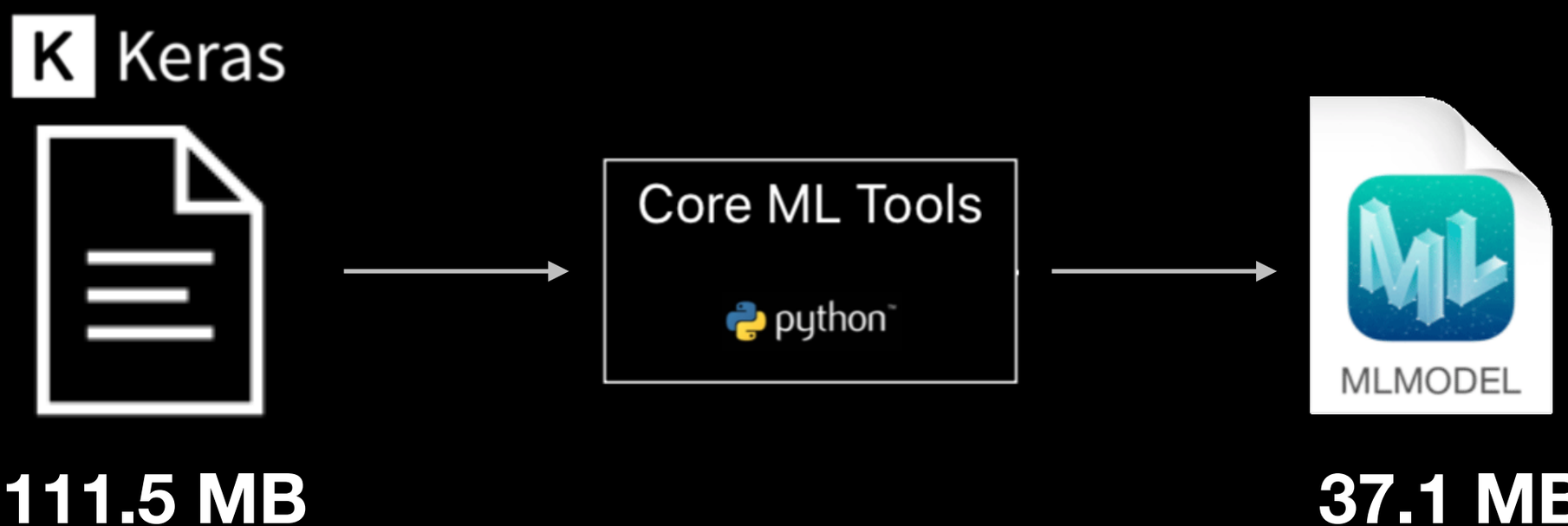
iMac Late 2013, Intel Core i7, 4 cores, 16 GB

TensorFlow built with CPU vectorized operations (AVX2)

Convert model

```
from coremltools.converters.keras import convert
from keras.models import load_model

keras_model = load_model("./Model/distracted_driver.h5")
coreml_model = convert(keras_model, image_input_names = "input1")
coreml_model.save("./Model/DistractedDriverKeras.mlmodel")
```



Accuracy on device

Fails 2’352 out of 4’467 sample images (+50%)

Image resizing

Dataset image (c7)
640x480

Default

.centerCrop
360x480

.scaleFill/.scaleFit
360x480

Accuracy on device

```
kerasClassificationRequest.imageCropAndScaleOption = .scaleFill
```

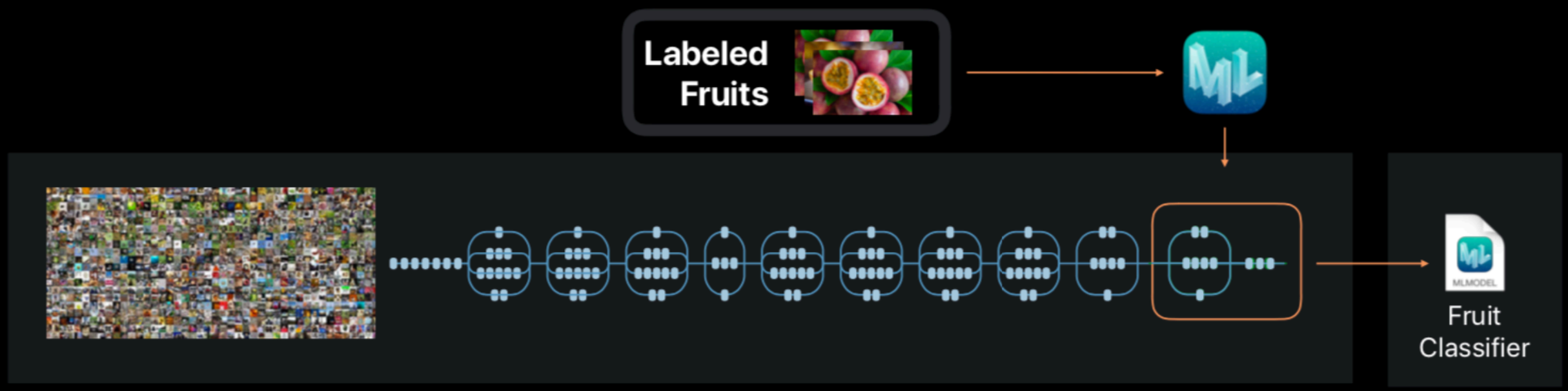
Fails 404 out of 4’467 sample images (~9%)

VS.

Precision: 98% in script 80% on device	**Precision:** 95% in script 91% on device
Training time: 1.4 hours	**Training time:** 8 hours
Performance: iPhone 7: 8 sec/img iPhone XR: 33 sec/img	**Performance:** iPhone 7: 9 img/sec iPhone XR: 45 img/sec
 178 KB	 37.1 MB

Not to

With Create ML, the model is really just the last layer over a pre-trained deep neural network



In a nutshell

- Image pre-processing can cause significant precision lost
- Leverages already excellent iOS support
- Real-time performance

- Complex architecture, long training times and heavy models
- When matched with the input resolution, the solution can render very good precision
- Can perform faster, due to highly-tuned architecture

Text Classification

Featured Prediction Competition

Toxic Comment Classification Challenge

Identify and classify toxic online comments

$35,000
Prize Money

Jigsaw/Conversation AI · 4,551 teams · a year ago

Overview | Data | Kernels | Discussion | Leaderboard | Rules | Team | My Submissions | Late Submission

Data Description

You are provided with a large number of Wikipedia comments which have been labeled by human raters for toxic behavior. The types of toxicity are:

- `toxic`
- `severe_toxic`
- `obscene`
- `threat`
- `insult`
- `identity_hate`

You must create a model which predicts a probability of each type of toxicity for each comment.

Offensive language ahead

Toxic

> **You are a know it all.....Jimbo! Why don't you get a life instead of sitting in front of your computer all day. The high priestess of knowledge, indeed!**

> **I don't mean to startle anyone but God is dead. We should not worry about him anymore. Just thought I would let everyone know. Well, goodbye and good luck with your newfound crisis of faith!**

Severe toxic

> **I'd rather deal with an honest asshole (which I would really like to see around here) than a bunch of ass-kissing, arrogant snakes (which I see in spades). This project just goes to hell that little bit further every day; I wonder how long until the articles I've written are bastardised in the way all of these are?**

> **I just learned how to suck up to people. You're very good at it, FisherQueen. As for the grammer, it should be fairly obvious that they're typos, now pick out a mistake here, bitch!**

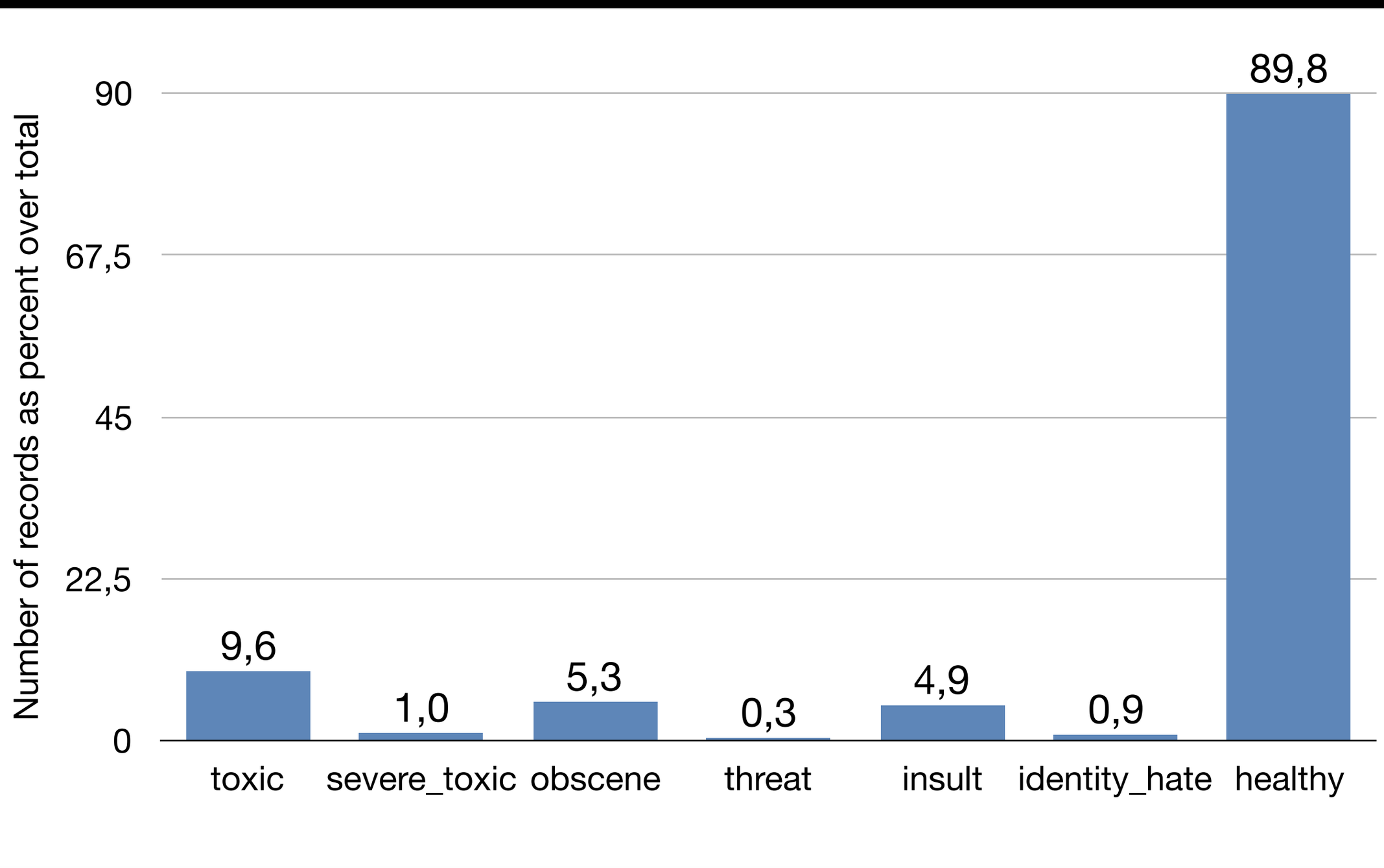
Identity hate

> **Black americans have a hive mind mentality and automatically switch political party preferences just like that.**

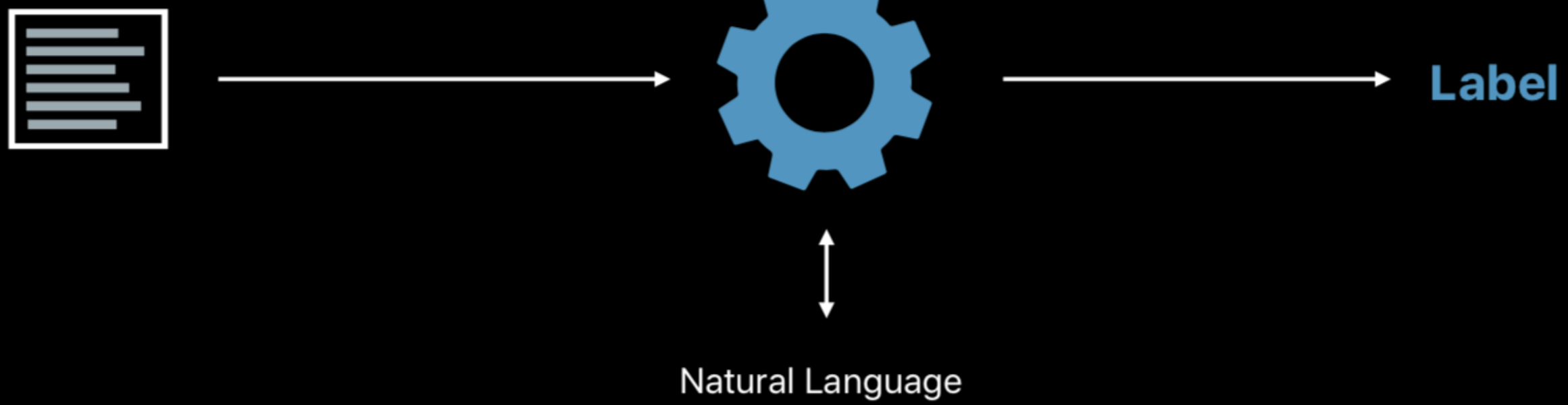
> **Ossmann, are you jew or your father was an SS officer?**

Data distribution

~160,000 comments



Stages for text classification



Training model

```
let parameters =
MLTextClassifier.ModelParameters(validationData: nil,
                                 algorithm: .maxEnt(revision: 1),
                                 language: .english)

let classifier = try? MLTextClassifier(trainingData: trainingData,
                                       textColumn: "comment_text",
                                       labelColumn: targetColumn,
                                       parameters: parameters)
```

Training model

```
Automatically generating validation set from 5% of the data.
Tokenizing data and extracting features
10% complete
20% complete
30% complete
40% complete
50% complete
60% complete
70% complete
80% complete
90% complete
100% complete
Starting MaxEnt training with 121170 samples
Iteration 1 training accuracy 0.904143
Iteration 2 training accuracy 0.909672
Iteration 3 training accuracy 0.941735
Iteration 4 training accuracy 0.952719
Iteration 5 training accuracy 0.960535
Iteration 6 training accuracy 0.967665
Iteration 7 training accuracy 0.974730
```

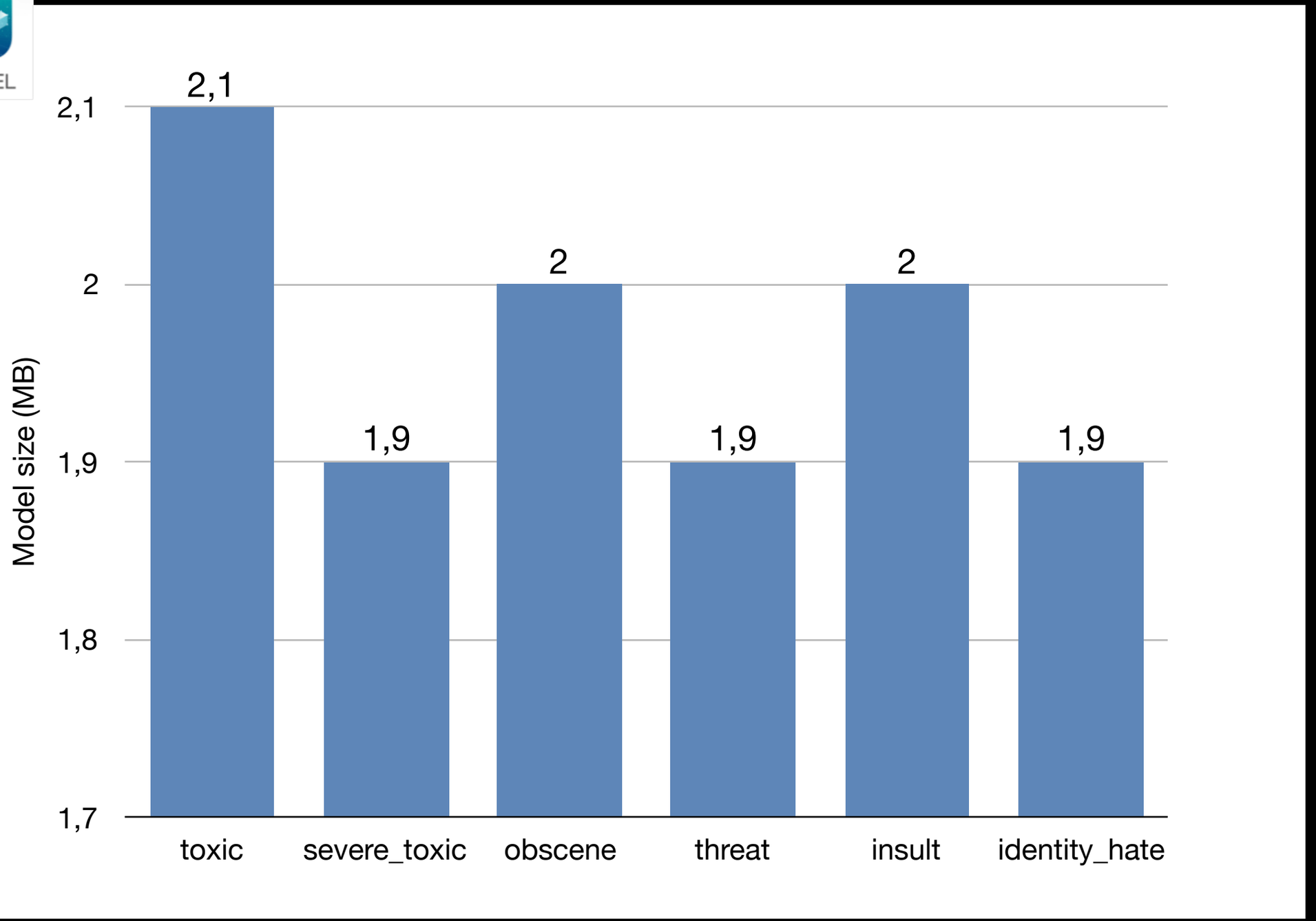
...

```
Iteration 42 training accuracy 0.997549
Iteration 43 training accuracy 0.997565
Iteration 44 training accuracy 0.997598
Iteration 45 training accuracy 0.997615
Finished MaxEnt training in 48.01 seconds
Classifier for "toxic":
TextClassifier
```

Accuracy ~ 95 %

Model size


MLMODEL


Model size (MB)
2,1
2
1,9
1,8
1,7
2,1
2
1,9
1,9
2
1,9
toxic
severe_toxic
obscene
threat
insult
identity_hate

Performance on device

A12

Neural engine
8 cores

>4,500 comments/second

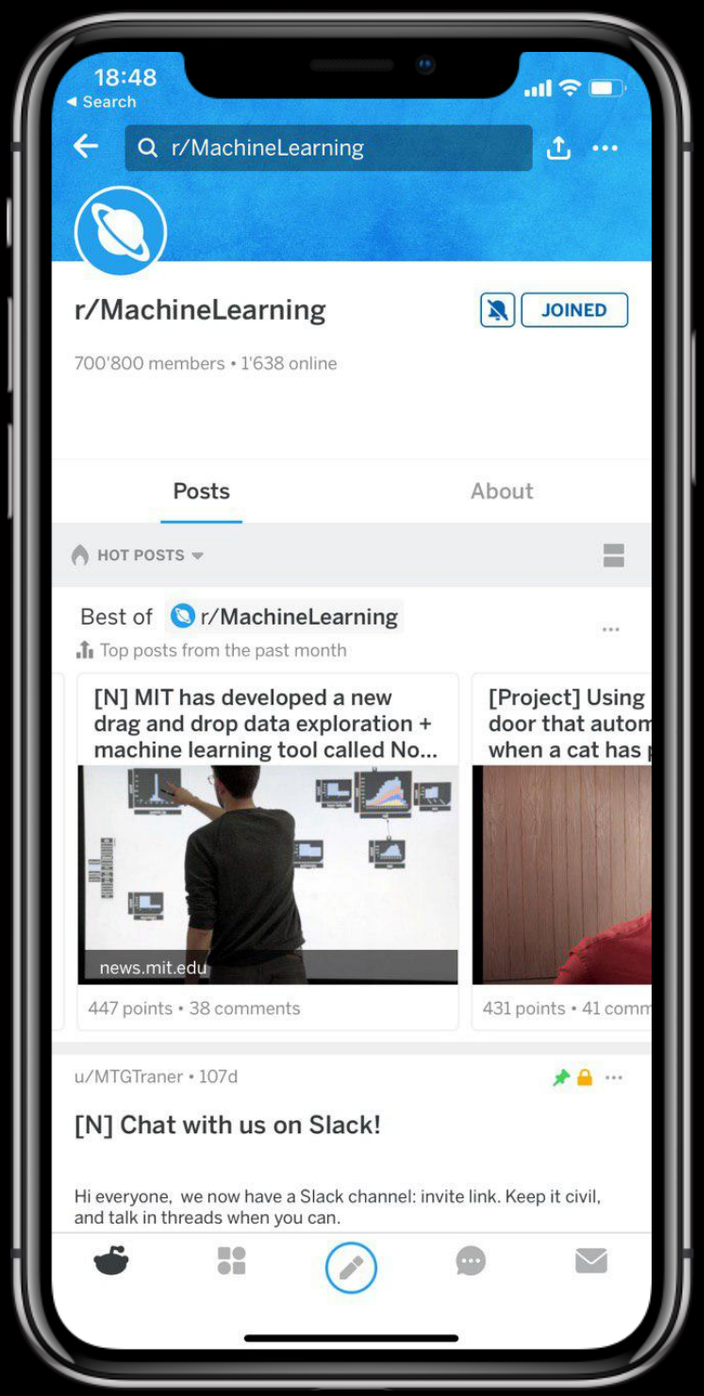
A10

No dedicated HW
4 cores

~2,300 comments/second

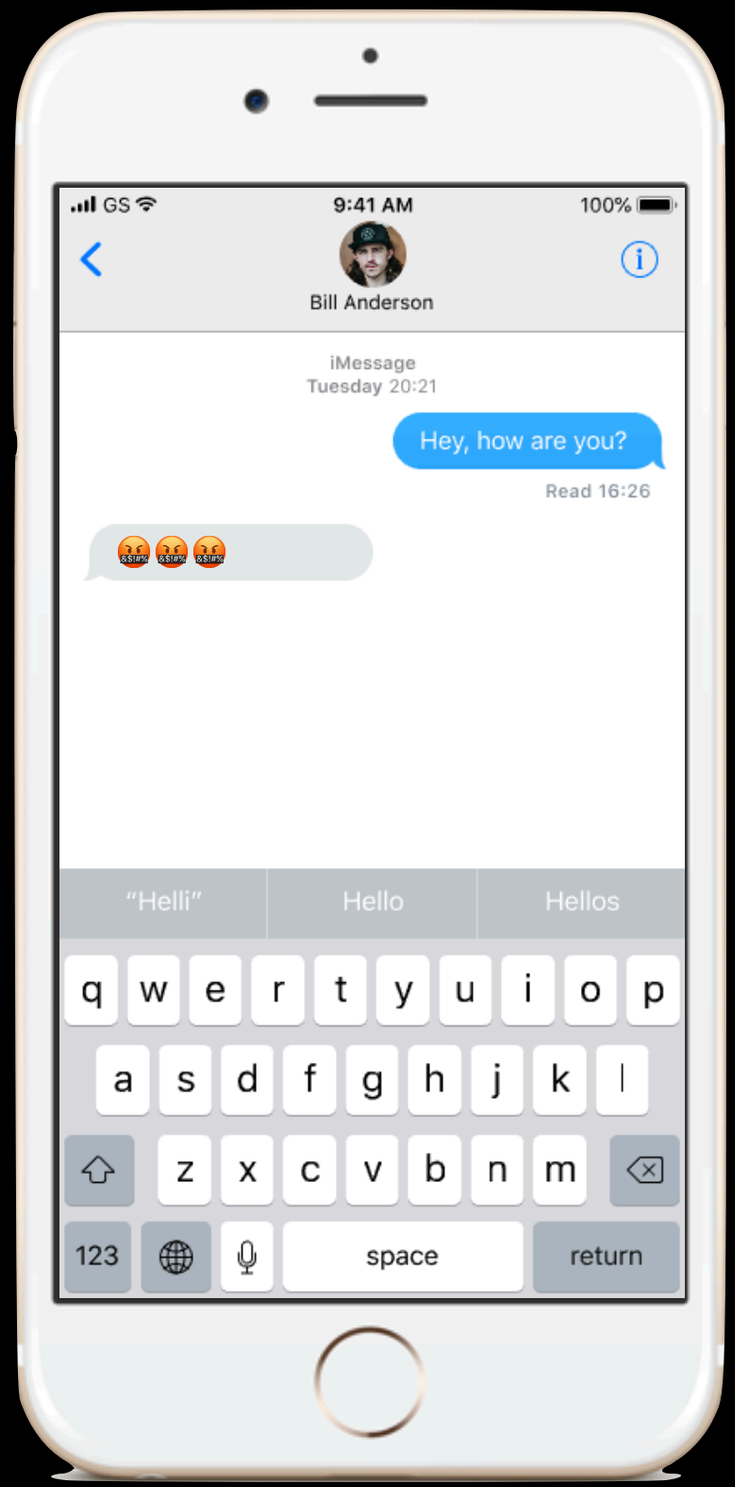
Measured over ~80,000 comments

Device vs cloud: forum app



Aggregated processing

Device vs cloud: end-to-end encrypted chat



On-demand processing

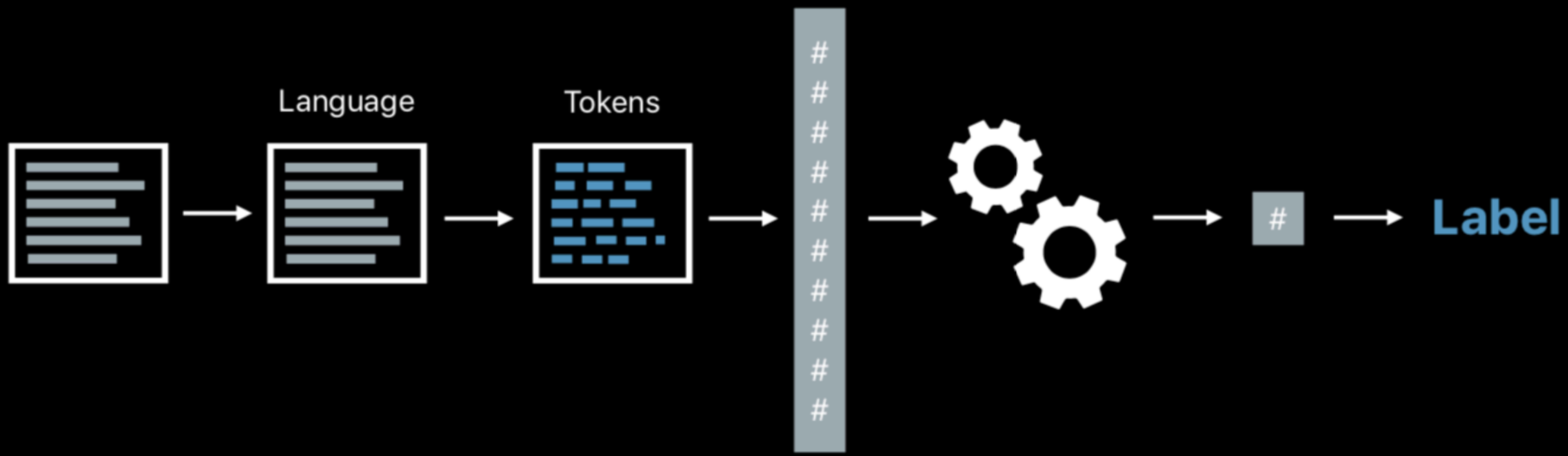
Bag of words

			features (f)								
Text	**Label**		*this*	*movie*	*is*	*good*	*the*	*bad*	*p(d|1)*	*p(d|0)*	*ratio*
This movie is good	1		1	1	1	1	0	0	0.6667	0.2222	3
The movie is good	1		0	1	1	1	1	0			
This movie is bad	0		1	1	1	0	0	1			
The movie is bad	0		0	1	1	0	1	1			
		ones	1	1	1	1	1	1			
p(c=1)	0.5	*p(f|1)*	0.667	1	1	1	0.667	0.333			
p(c=0)	0.5	*p(f|0)*	0.667	1	1	0.333	0.667	1			

> **… situation where a linear model is pretty close to the state of the art**
>
> **— Jeremy Howard, from fast.ai**

Source: https://youtu.be/37sFIak42Sc?t=3745

Stages for text classification



Clean up texts ➡ Tokenize ➡ Vectorizer ➡ Logistic regression with Naive-Bayes ➡ Probabilities ➡ Classification

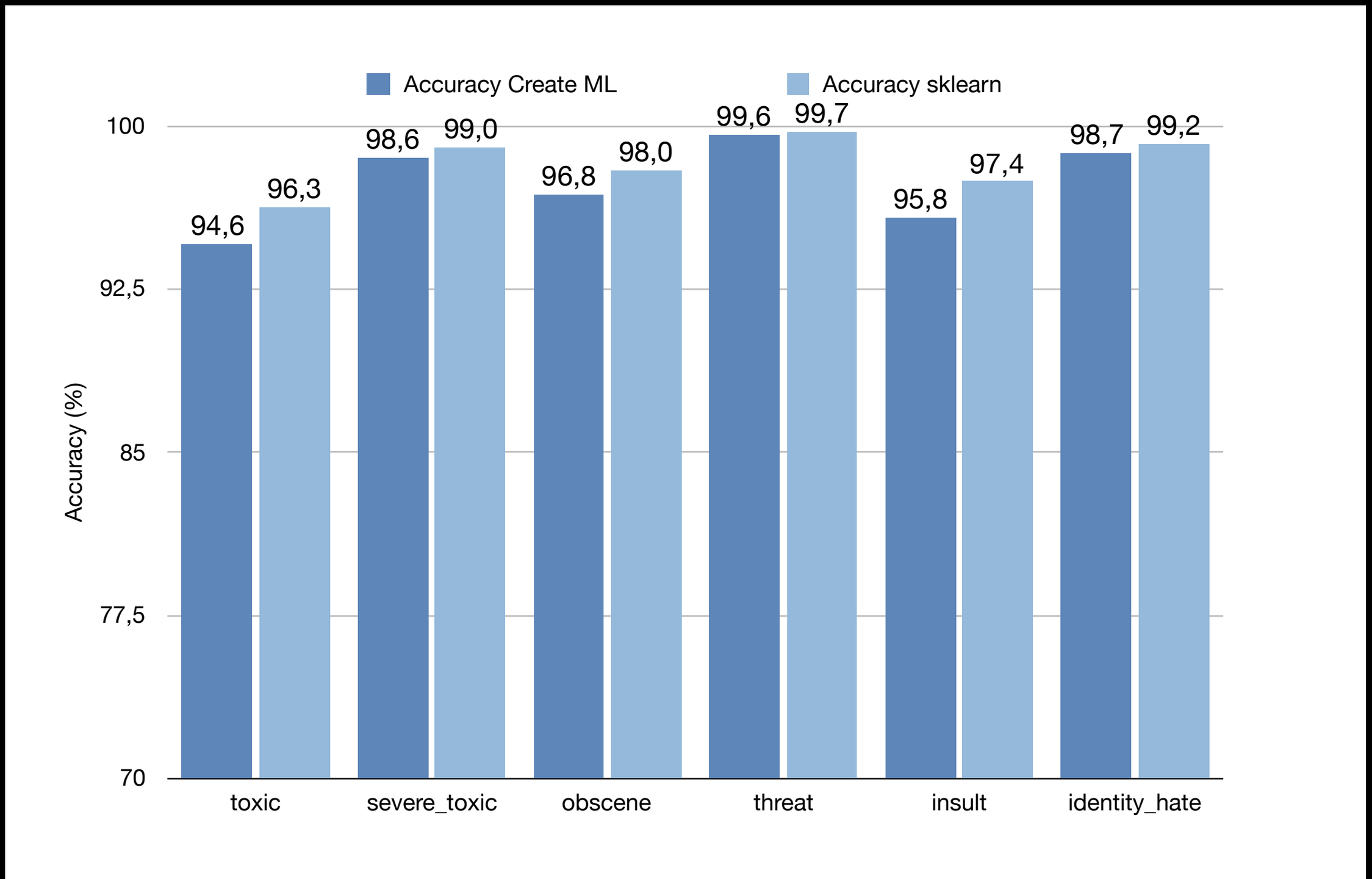
Stages for text classification

166k terms (features)

```
159479,wikimeetup
159480,wikimegamaster
159481,wikimembers10
159482,wikimemories
159483,wikimilitary
159484,wikimitzvah
159485,wikimod
159486,wikimods
159487,wikimoral
159488,wikimossad
159489,wikimurder
159490,wikination
159491,wikinazi
159492,wikinazis
159493,wikinews
159494,wikinews_interviews_scott_lucas
159495,wikinfo
159496,wiking
159497,wikinger
159498,wikinic
159499,wikininja
159500,wikinoid
159501,wikinonsense
159502,wikinorthernireland
159503,wikinumbers
159504,wikinut
159505,wikiocaasiyahoo
159506,wikioedia
159507,wikioffice
159508,wikioh
```

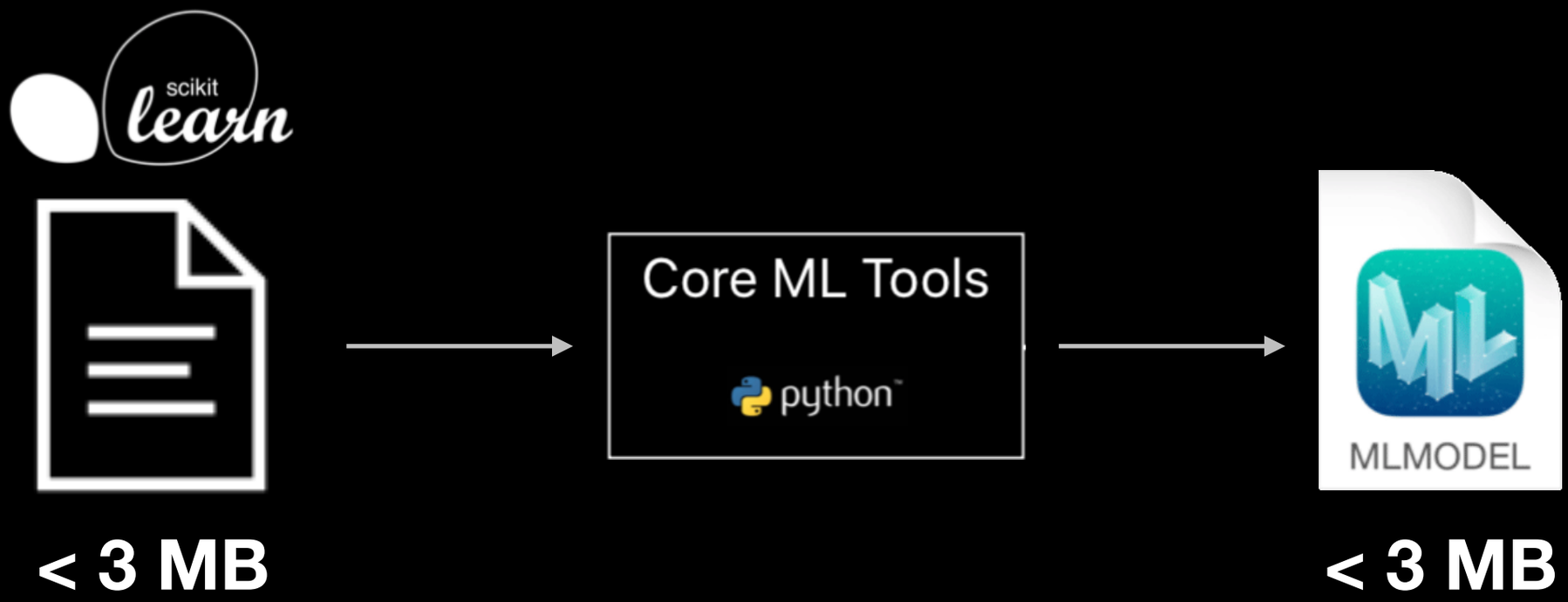
vs.

Accuracy


Accuracy Create ML
Accuracy sklearn
100
92,5
85
77,5
70
Accuracy (%)
94,6
96,3
98,6
99,0
96,8
98,0
99,6
99,7
95,8
97,4
98,7
99,2
toxic
severe_toxic
obscene
threat
insult
identity_hate

Convert models

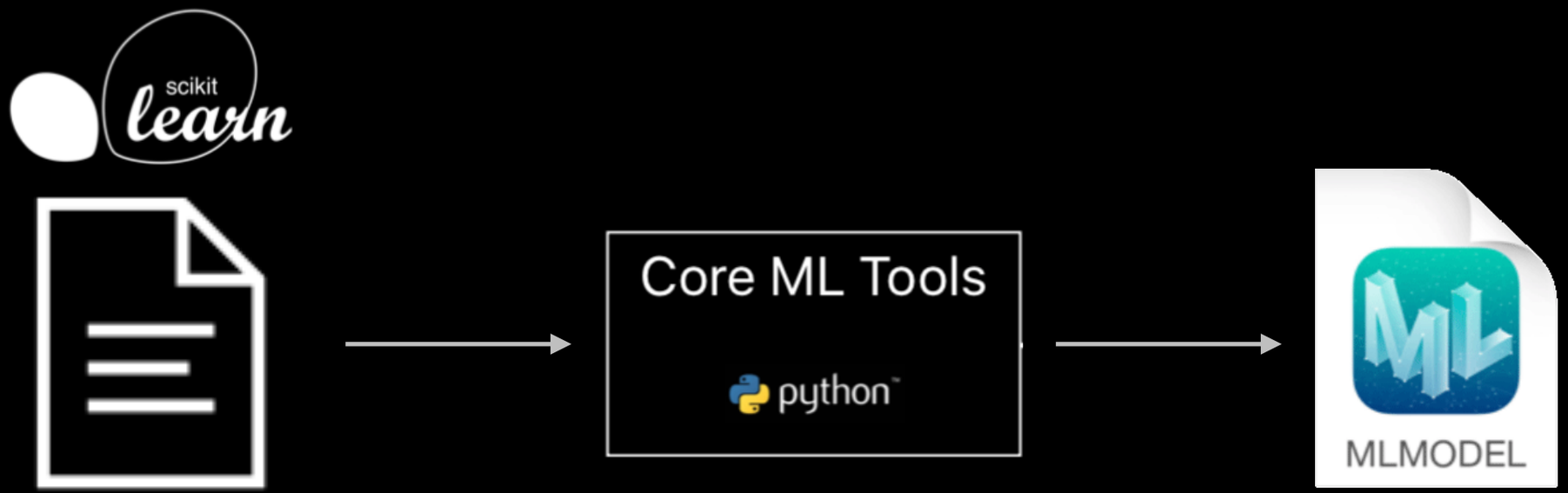
Logistic Regressor (sklearn)



< 3 MB **< 3 MB**

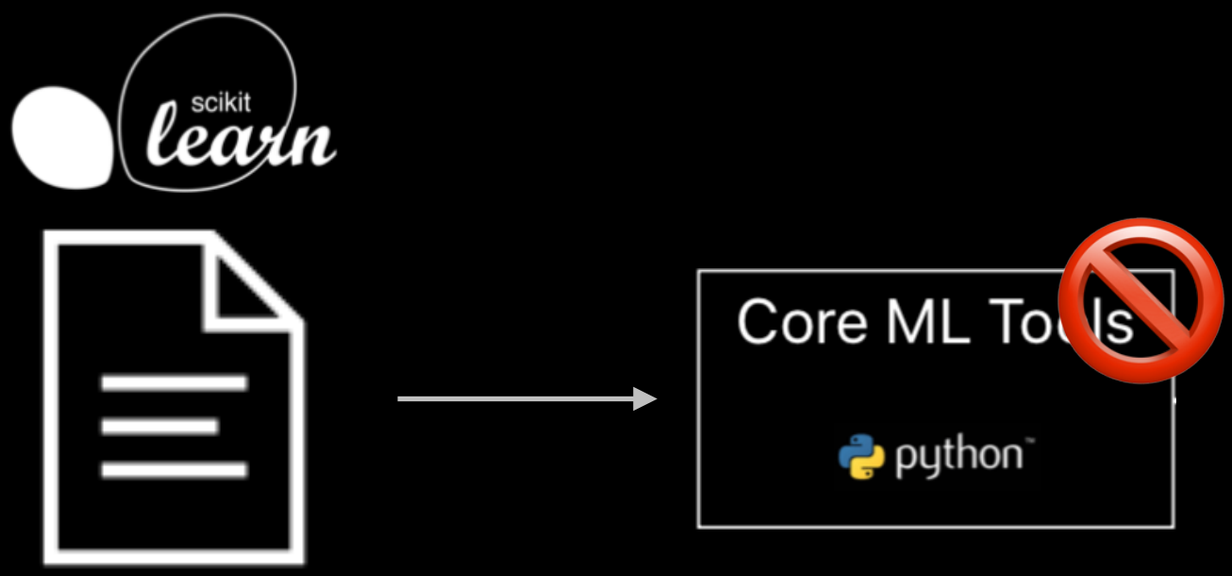
Convert models

Logistic Regressor (sklearn)



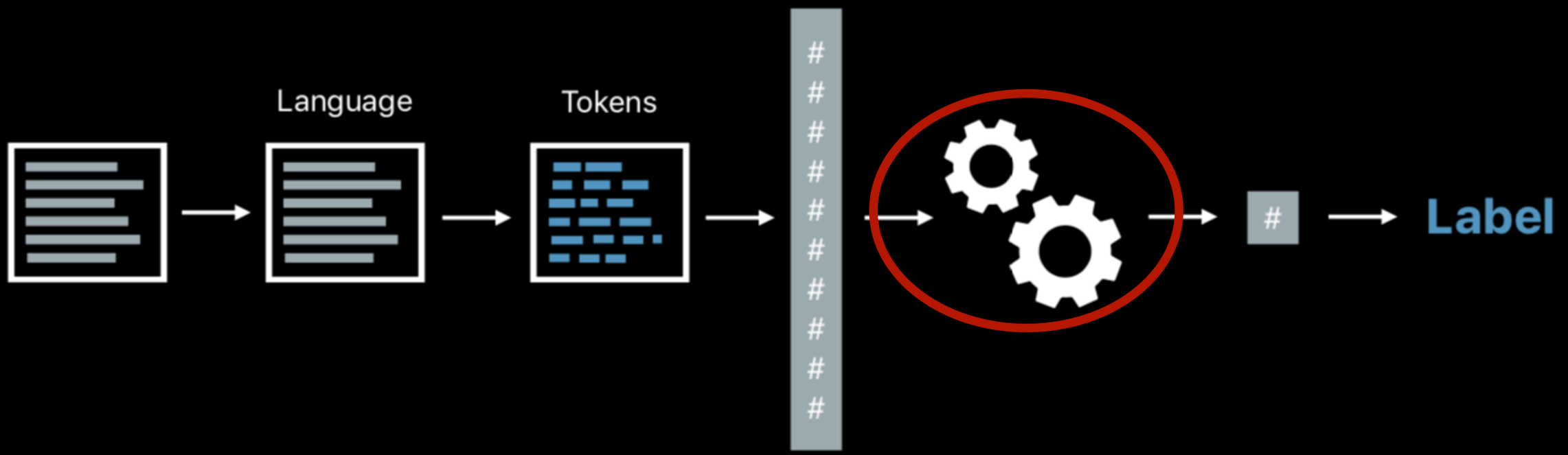
< 3 MB **< 3 MB**

TF-IDF Vectorizer (sklearn)



Not supported

NLP pipeline



Model is just one step in the pipeline

Towards the device

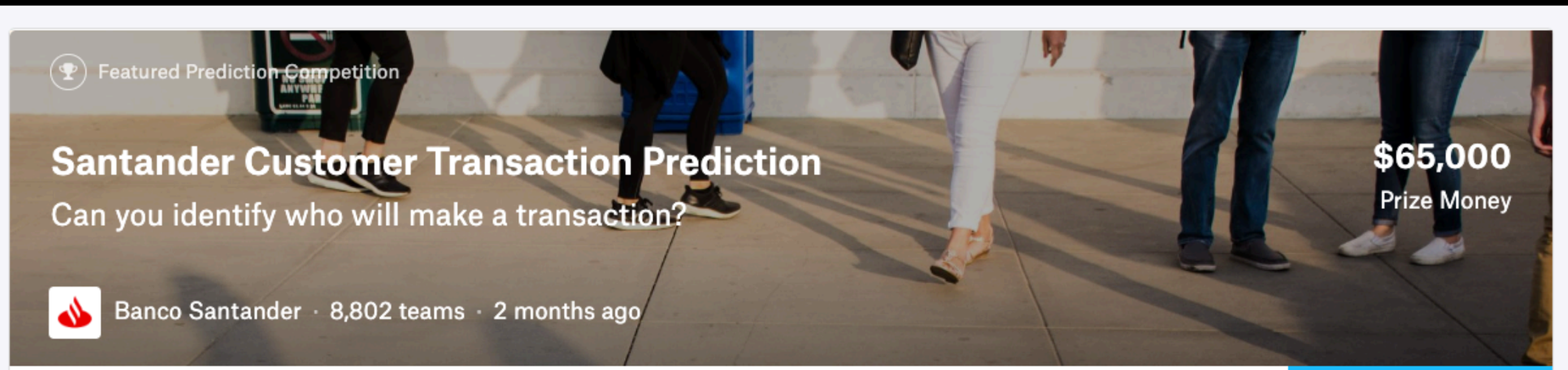
- Several obstacles
  - Though tokenizing is available via NL framework, there is no built-in support for creating the bag of words
  - No built-in support for sparse matrices (needed to store the bag of words)
  - No built-in support for dot product (needed to multiply for corrective ratio)

In a nutshell

- Streamlined solution
- High accuracy, low model size, fast performance, fast training

- Need to code feature extraction on device
- Lacking support for typical algorithms

Tabular data



Data Description

You are provided with an anonymized dataset containing numeric feature variables, the binary `target` column, and a string `ID_code` column.

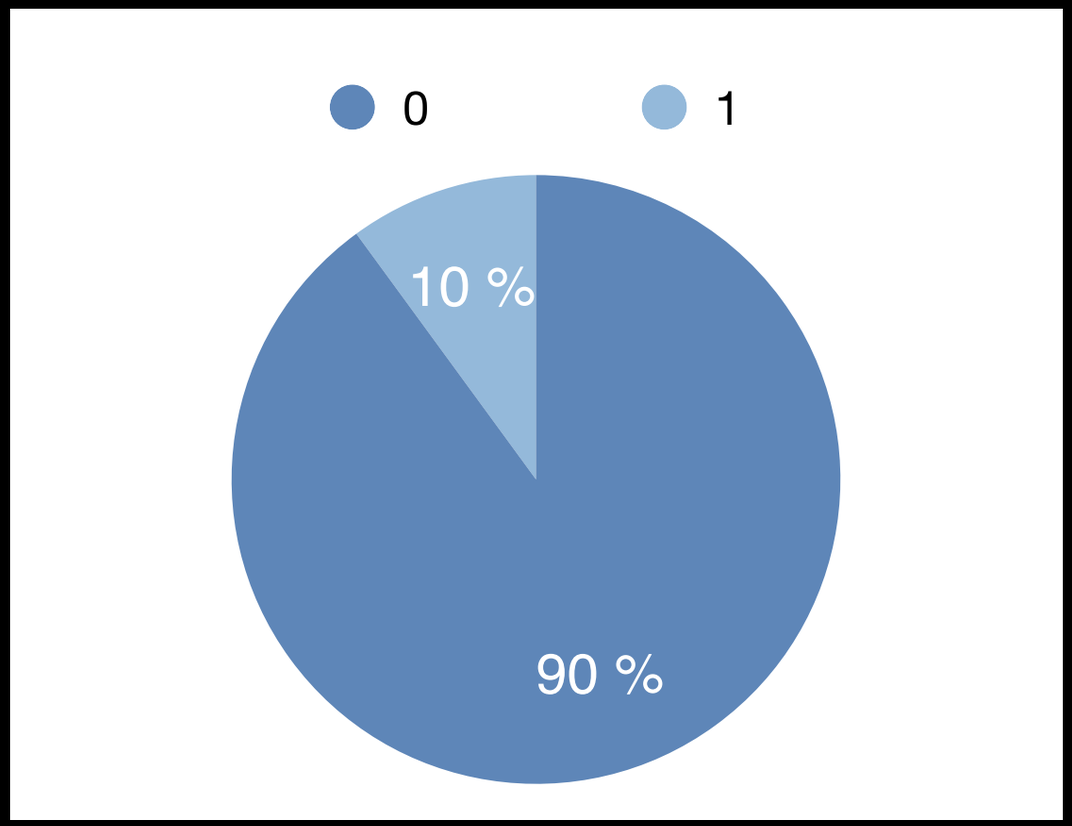
The task is to predict the value of `target` column in the test set.

File descriptions

- **train.csv** - the training set.
- **test.csv** - the test set. The test set contains some rows which are not included in scoring.
- **sample_submission.csv** - a sample submission file in the correct format.

Tabular data

- 200,000 records
- 200 numerical features
- 1 binary target: customer will make a transaction or not



Model training

Regressor Cases

The regressor case picked automatically when initializing an `MLRegressor`.

```
case linear(MLLinearRegressor)
```

A regressor that estimates the target as a linear function of the features.

```
case decisionTree(MLDecisionTreeRegressor)
```

A regressor that estimates the target by learning rules to split the data.

```
case boostedTree(MLBoostedTreeRegressor)
```

A regressor based on a collection of decision trees combined with gradient boosting.

```
case randomForest(MLRandomForestRegressor)
```

A regressor based on a collection of decision trees trained on subsets of the data.

Model training

```
let parameters =
MLBoostedTreeRegressor.ModelParameters(validationData: validationData,
                                       maxDepth: 6,
                                       maxIterations: 2500,
                                       minLossReduction: 0.0,
                                       minChildWeight: 0.1,
                                       randomSeed: 42,
                                       stepSize: 0.02,
                                       earlyStoppingRounds: nil,
                                       rowSubsample: 0.3,
                                       columnSubsample: 0.3)

let regressor = try? MLBoostedTreeRegressor(trainingData: trainingData,
                                            targetColumn: "target",
                                            featureColumns: featureColumns,
                                            parameters: parameters)
```

Model training

`maxDepth`
The maximum depth of the tree. Must be a value of at least 1.

The default value is 6.

`maxIterations`
The maximum number of passes through the data. Each iteration creates an extra tree.

The default value is 10.

`minLossReduction`
The minimum amount of reduction in the loss function that is required to make another split to the data. Larger values help prevent overfitting.

The default value is 0.

`minChildWeight`
Determines the minimum weight of each leaf node of the tree. Larger values help prevent overfitting.

The default value is 0.1.

`randomSeed`
A seed for internal random operations. Set this value to ensure reproducible results.

The default value is 42.

Background reading

When used to minimize the above function, a standard (or "batch") gradient descent method would perform the following iterations :

$$w := w - \eta \nabla Q(w) = w - \eta \sum_{i=1}^{n} \nabla Q_i(w)/n,$$

where η is a step size (sometimes called the *learning rate* in machine learning).

In classical statistics, sum-minimization problems arise in least squares and in maximum-likelihood estimation (for independent observations). The general class of estimators that arise as minimizers of sums are called M-estimators. However, in statistics, it has been long recognized that requiring even local minimization is too restrictive for some problems of maximum-likelihood estimation.[2] Therefore, contemporary statistical theorists often consider stationary points of the likelihood function (or zeros of its derivative, the score function, and other estimating equations).

The convergence of stochastic gradient descent has been analyzed using the theories of convex minimization and of stochastic approximation. Briefly, when the learning rates η decrease with an appropriate rate, and subject to relatively mild assumptions, stochastic gradient descent converges almost surely to a global minimum when the objective function is convex or pseudoconvex, and otherwise converges almost surely to a local minimum.[4][5] This is in fact a consequence of the Robbins-Siegmund theorem.[6]

Background reading

Choosing solution

Stratified k-folds to prevent overfitting

5 models trained with partitioned training set

Output averaged over 5 models

Solutions available for LightGBM, XGBoost and CatBoost

Choosing solution

Best solution with LightGBM 🚫

`Converters`

Automatically convert models from popular machine learning libraries such as Keras, Caffe, scikit-learn, LibSVM, and XGboost to the Core ML format.

`caffe.convert`	Convert a Caffe model to Core ML format.
`keras.convert`	Convert a Keras model to Core ML protobuf specification (.mlmodel).
`libsvm.convert`	Convert a LIBSVM model to Core ML format.
`sklearn.convert`	Convert scikit-learn pipeline, classifier, or regressor to Core ML format.
`xgboost.convert`	Convert a trained XGBoost model to Core ML format.

Converting model

Try with XGBoostClassifier

```
    raise TypeError("Unexpected type. Expecting XGBoost model.")
TypeError: Unexpected type. Expecting XGBoost model.
```

Choosing solution

apple / coremltools

Used by 169 | Watch 80 | Star 1,368 | Fork 169

Code | Issues 125 | Pull requests 14 | Projects 0 | Wiki | Security | Insights

Tree: cf0dcfc8d1 coremltools / coremltools / converters / xgboost / _tree_ensemble.py

Find file | Copy path

gustavla XGBoost 0.7 name change fix d9e1bbc on 19 Jan 2018

2 contributors

157 lines (121 sloc) | 5.56 KB

Raw | Blame | History

```
105     import json
106     import os
107     feature_map = None
108     if isinstance(model,  (_xgboost.core.Booster, _xgboost.XGBRegressor)):
109
```

Converting model

Try with XGBoostRegressor

RMSE = 0.082

RMSE = 3.17

Converting model

apple / coremltools

Used by 170 | Watch 80 | Star 1,370 | Fork 169

Code | Issues 126 | Pull requests 14 | Projects 0 | Wiki | Security | Insights

Filters | is:issue is:open | Labels 11 | Milestones 0 | New issue

126 Open | 78 Closed | Author | Labels | Projects | Milestones | Assignee | Sort

apple / coremltools

Used by 170 | Watch 80 | Star 1,370 | Fork 169

Code | Issues 126 | Pull requests 14 | Projects 0 | Wiki | Security | Insights

XGBoost model in CoreMLTools produces completely different results #174

New issue

Open stylianos-kampakis opened this issue on 16 Apr 2018 · 8 comments

Converting model

Xgboost classifier conversion #293

Open **zaccharieramzi** wants to merge 23 commits into `apple:master` from `zaccharieramzi:xgboost-classifier-conversion`

Conversation 2 | Commits 23 | Checks 0 | Files changed 4 | +390 −27

zaccharieramzi commented on 20 Nov 2018 • edited

This solves #174

It will allow users to convert an xgboost classifier model to a coreml one.

Reviewers

No reviews

Assignees

No one assigned

Approved and merged on 22 Jun 2019

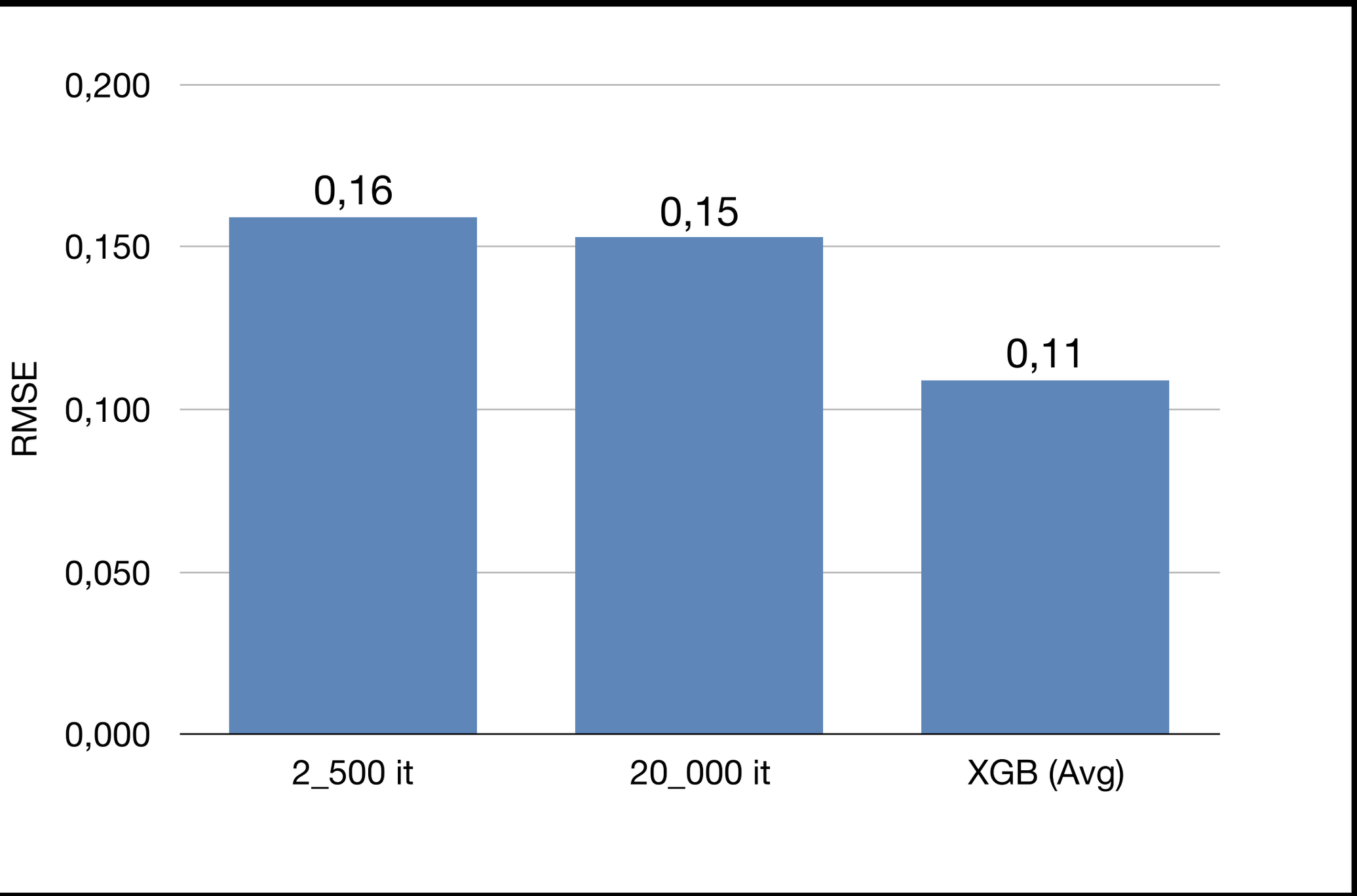
Converting model

Back to XGBoostClassifier

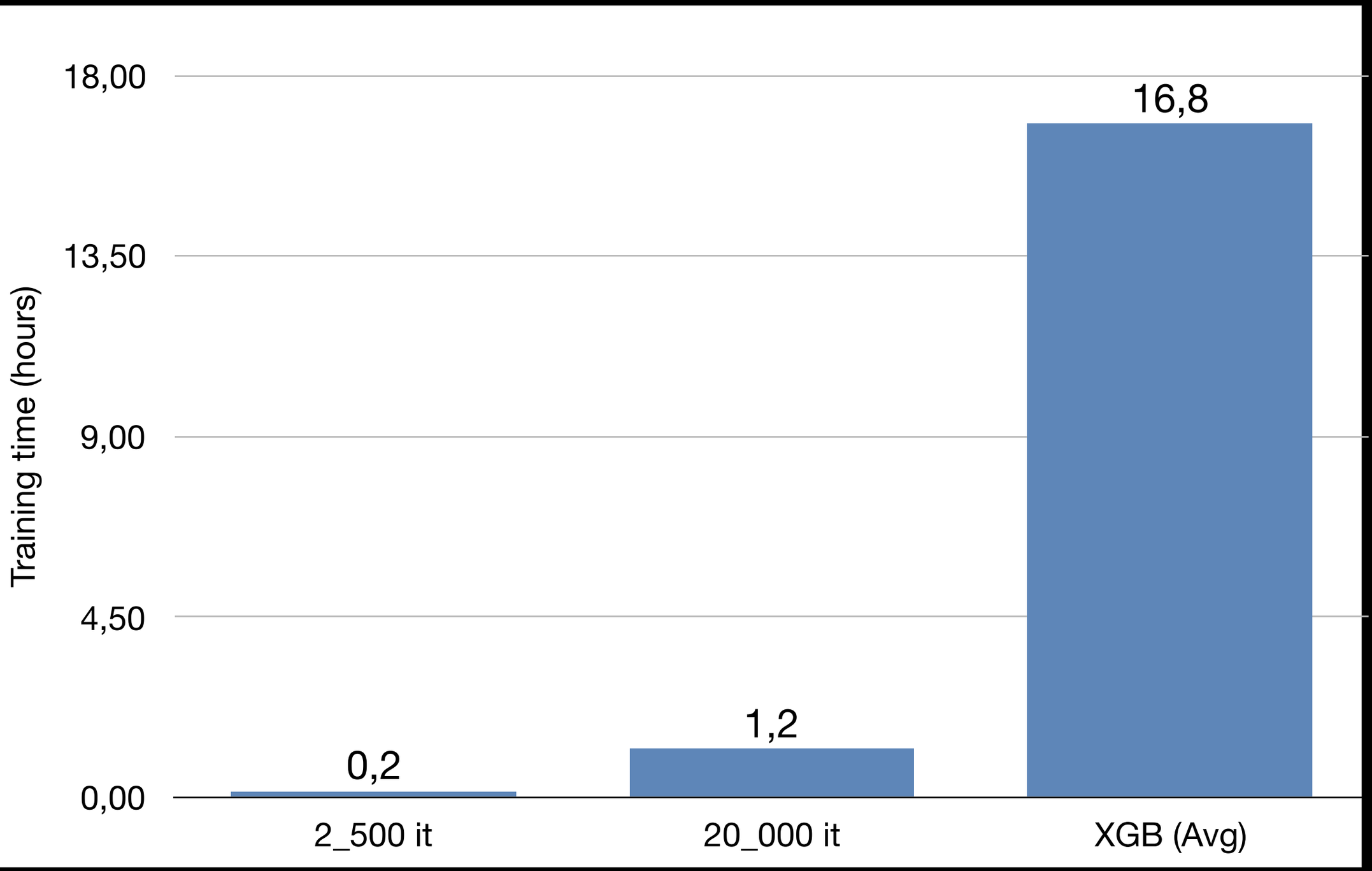
RMSE = 0.082

RMSE = 0.11

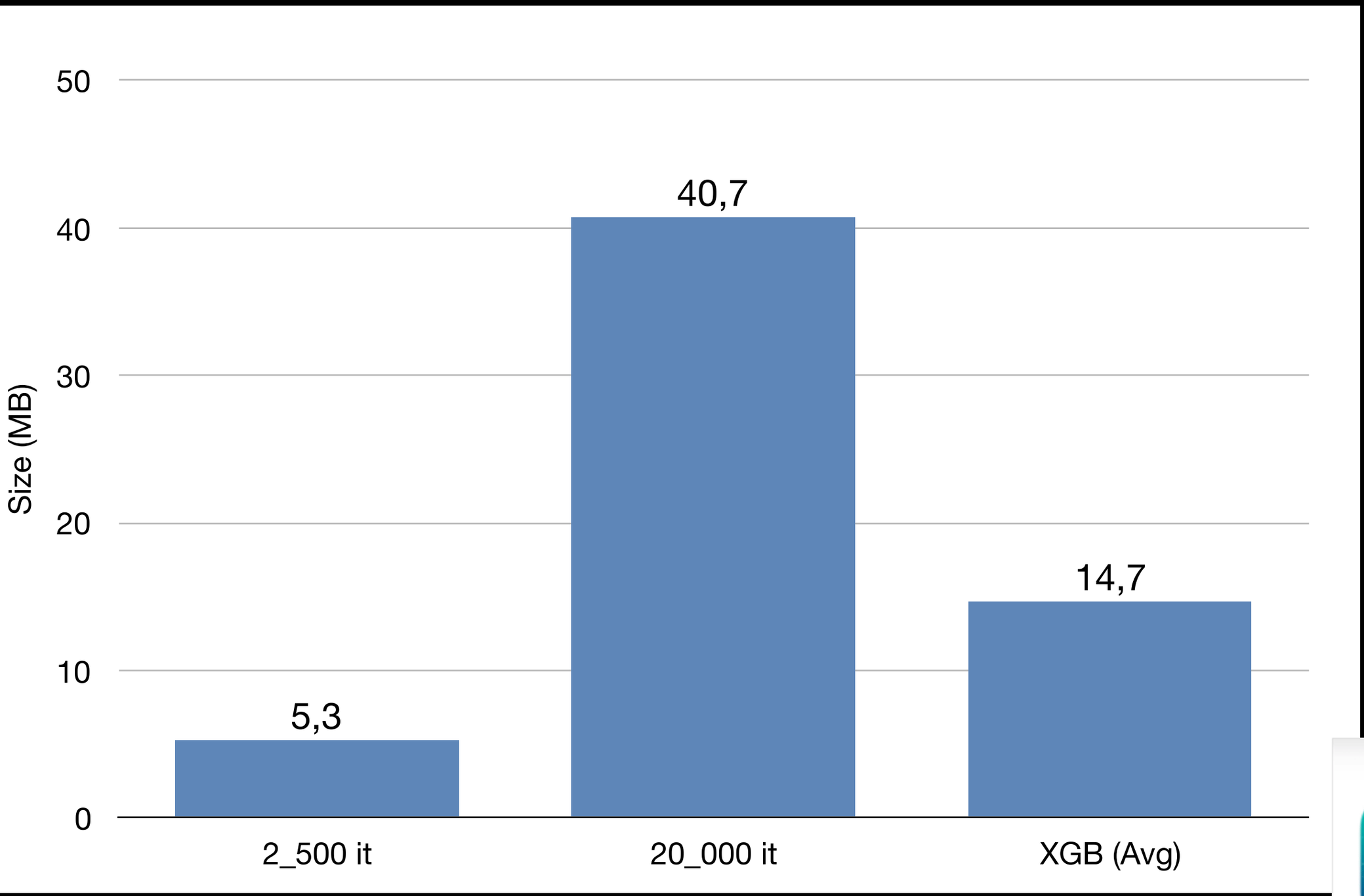
Model accuracy



Training time

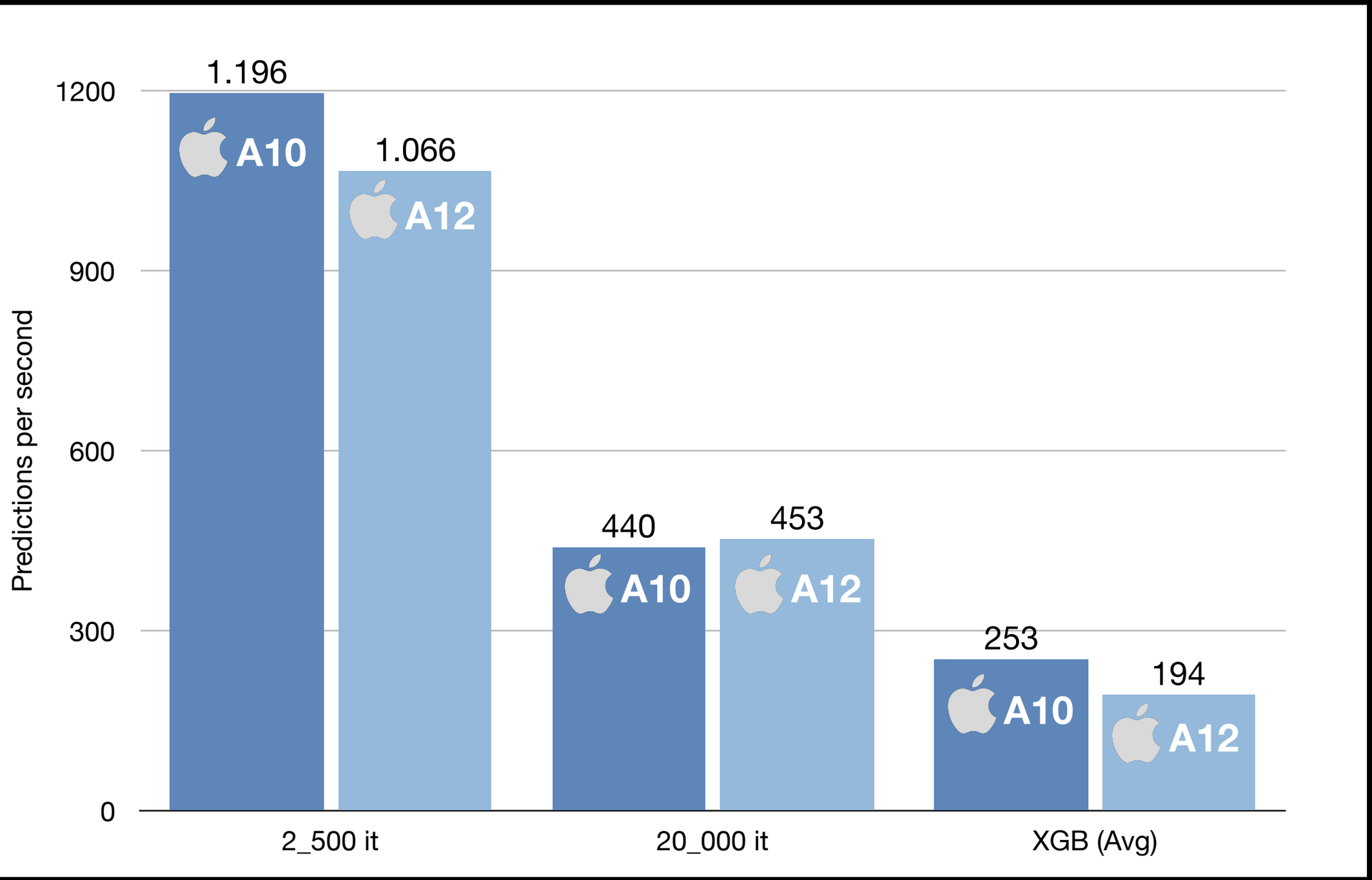


Model size



MLMODEL

Performance on device



In a nutshell

- Params are hard to choose
- With a few training iterations we get a competitive model, small in size and blazing fast

- Limited (and buggy) support of model formats
- Longer training time
- Fine-tuned trade-offs between accuracy and size

Take-away

- Create ML is a seamless entry into ML for iOS developers
- Good results out of the box
- Create ML may lack flexibility
- iOS lacks support for scientific algorithms
- Model conversion is still very much a WIP
- Performance is real-time in every case

Create ML challenge

Does this really work on real problems or is it only good for toy demos?

✅

Can we get real time performance?

✅

Bridging the gap

Turi Create

MLMODEL

I want to...	Machine Learning Task
Label Images	Image Classification
Recognize objects within images	Object Detection
Find similar images	Image Similarity
Create stylized avatars / profile images	Style Transfer
Personalize choices for users	Recommender
Detect an activity using sensors	Activity Classification
Analyze sentiment of messages	Text Classifier
Predict a label	Classifiers
Predict numeric values	Regression
Group similar datapoints together	Clustering

Coming this fall

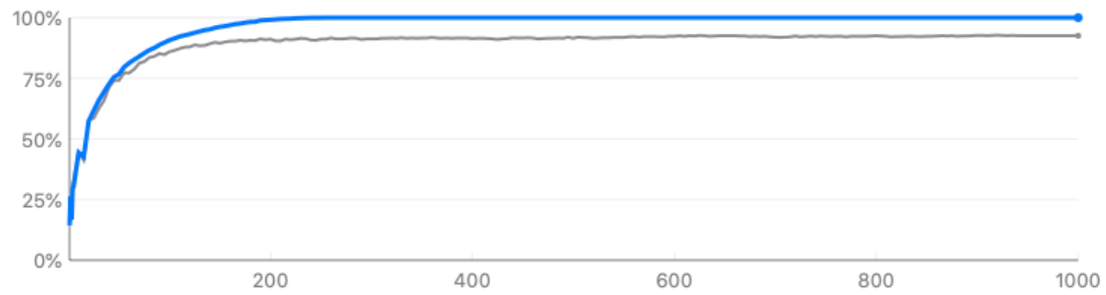
Create ML macOS app
(bundled with Xcode)

Train models from a GUI

Input	Accuracy		Output
10 Classes	100% Training	93% Validation	11% Testing	148 KB

▼ Activity

Completed training



Completed after iteration 1000 of maximum 1000

— Training accuracy 100%

— Validation accuracy 93%

▼ Accuracy

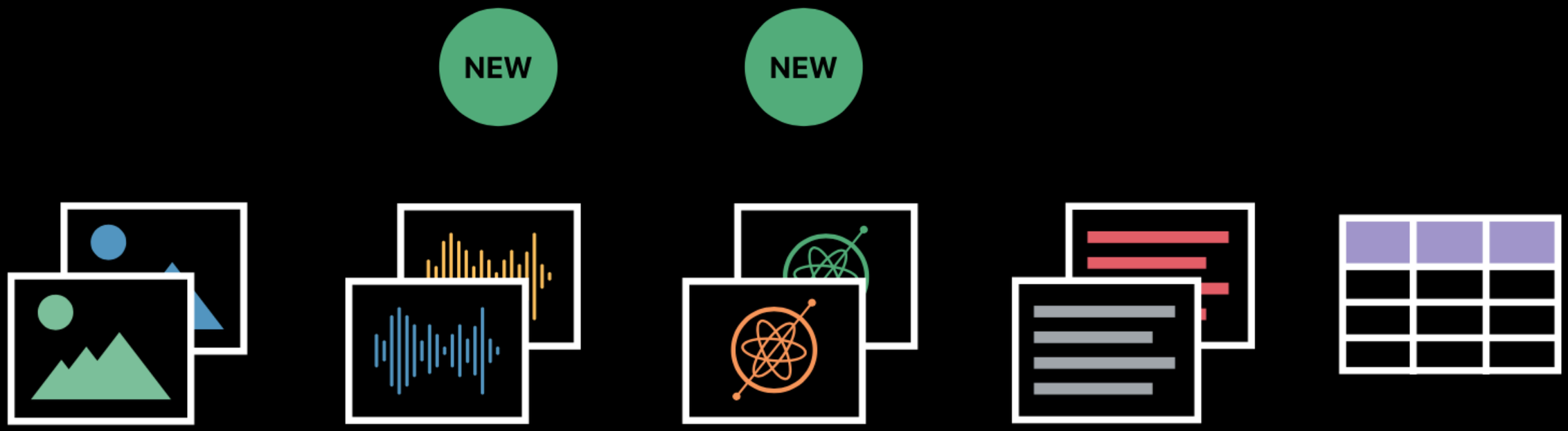
Class	Item Count	Precision	Recall
c8	1375	100%	100%
c4	1656	100%	100%
c0	1754	100%	100%
c1	1627	100%	100%
c7	1376	100%	100%
c5	1655	100%	100%
c6	1625	100%	100%
c3	1661	100%	100%
c9	1523	100%	100%

Training completed after 1 hour, 17 minutes — today at 10:45

Make a Copy

Currently in beta 3

New models supported



Further advances

- Transfer learning for NLP (static or dynamic embeddings)
- On-device model personalisation

Sources

- WWDC images property of Apple (Fair Use)
- Image classification kernel by Robert Ruzzo
- Text classification kernel by Jeremy Howard
- Tabular data kernel by Vladislav Bogorod

Experiments

- Distracted Driver repo
- Jigsaw Toxicity repo
- Santander Customer Transaction Prediction repo

Thank you

@atineoSE
me@adriantineo.com