# BlinkML:

## Efficient Maximum Likelihood Estimation with Probabilistic Guarantees

**Yongjoo Park** Jingyi Qing Xiaoyang Shen Barzan Mozafari

University of Michigan

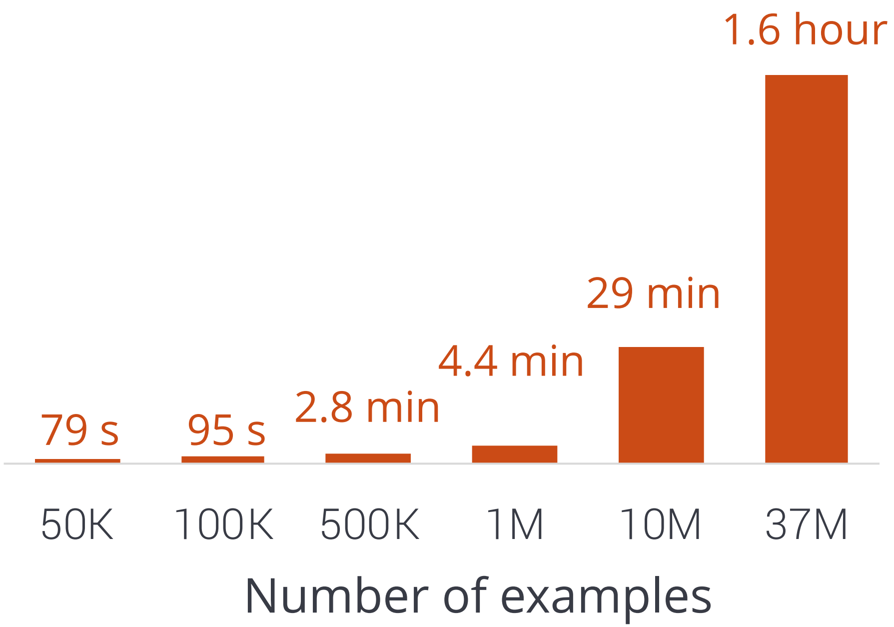
# Machine learning workloads are slow and costly

## More data ⇒ slower training

- **1 hour 35 minutes** for 37M training examples

## Often training multiple models

- New data becoming available
- Feature engineering




*Criteo dataset, Logistic Regression with L-BFGS optimization algorithm*

# Key Question:
# Can sampling accelerate ML training?

## SQL analytics

$$\text{sum}(X) = (1/N) \sum_{i=1..N} X_i$$

$$\approx (1/\mathbf{n}) \sum_{i=1..\mathbf{n}} X_i$$

## ML training

iterative gradient computation

$$\text{grad} = (1/N) \sum_{i=1..N} g(x_i \mid \theta_t)$$

$$\approx (1/\mathbf{n}) \sum_{i=1..\mathbf{n}} g(x_i \mid \theta_t)$$

[Park et al. SIGMOD'18]

A platform-independent approach

Do similar properties hold?

# Three key challenges

## Model quality guarantee

- No closed-form solution: $\mathrm{grad}(\theta_N) = (1/N) \sum_{i=1..N} g(x_i \mid \theta_N) = 0$
- CLT or Hoeffding is **NOT** directly applicable

## Generalization

- Logistic Regression ≠ Principal Component Analysis

## Efficiency

- Too many approximate models >(longer) A single full model

# Our core contribution

A system for **efficient**, **quality-guaranteed** ML training

It supports models trained via **maximum likelihood estimation**

1. Linear Regression
2. Logistic Regression (#1 classifier according to 2017 Kaggle survey)
3. Probabilistic PCA
4. Generalized Linear Models, …

[https://www.kaggle.com/surveys/2017]

We bring **Fisher**'s theory to practice, and apply it in a **novel way** for **quality-guaranteed, sampling-based ML**

# To put things into context

We: **Uniform random sampling** is **effective**!

Much different from the work on biased/importance sampling

1. simple
2. no *a priori* knowledge required
3. still significant speedups

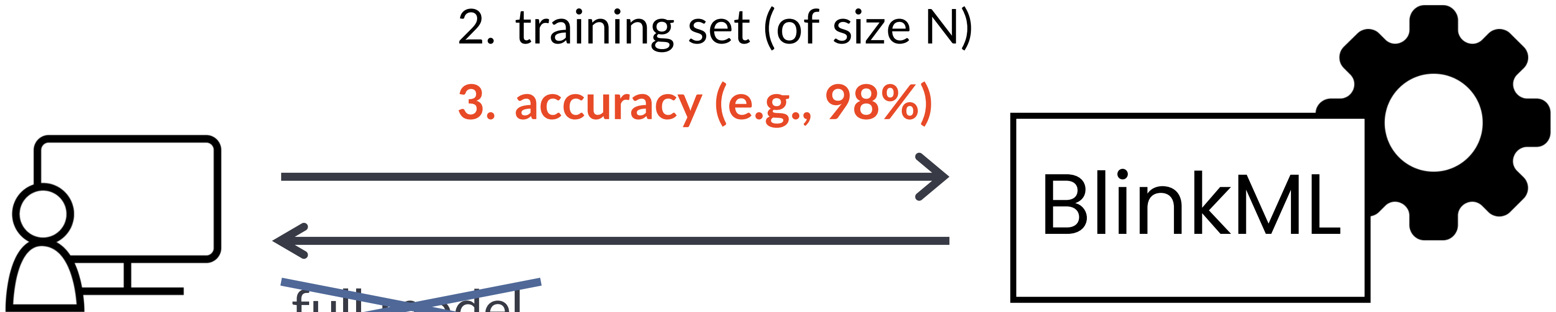
Generalize AQP to **multivariate** models

**Orthogonal** to AutoML

<SystemOverview>...

# BlinkML: interface

1. model type (e.g., Logistic Regression)
2. training set (of size N)
3. accuracy (e.g., 98%)




what's this process?

~~full model~~

98% accurate approx model

what does accuracy mean?

**Accuracy 1- ε** means

$E_x[\mathbf{1}(\text{full}(x) \neq \text{approx}(x))] \leq \varepsilon$ with high probability (e.g., 95%)

# BlinkML: internal workflow

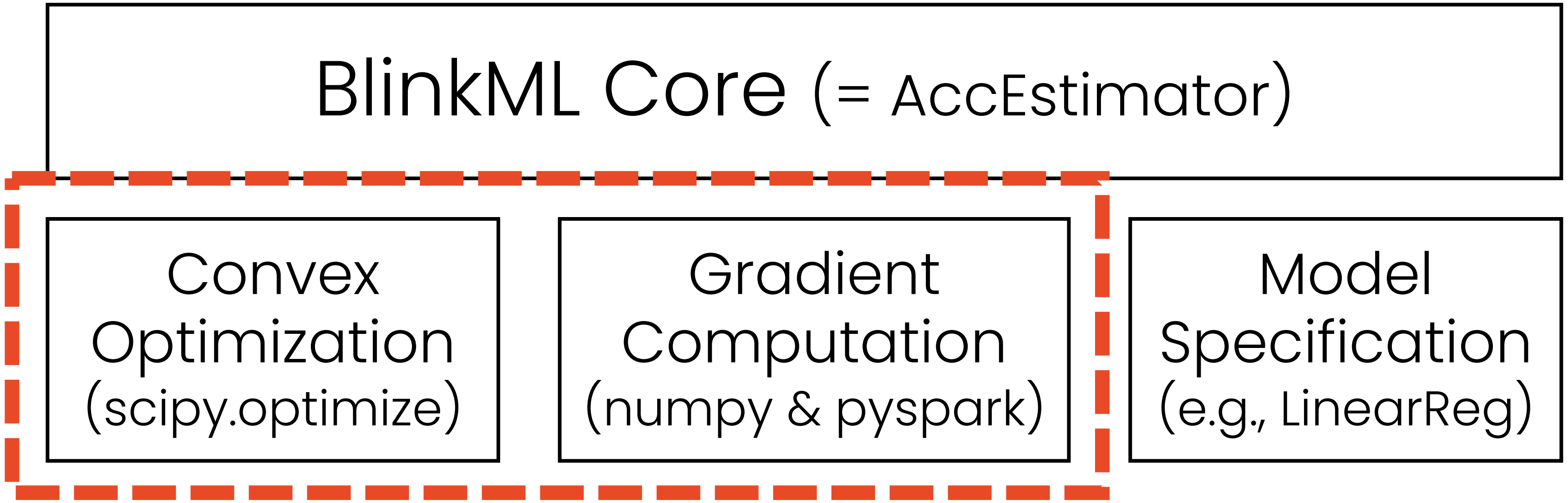
Step 1: **profile** model/data complexity

Step 2: **estimate** *min sample size*

Crucial component (= AccEstimator):
estimate accuracy of approx model w/o full model

Step 3: **train** an approximate model

# BlinkML: architecture




1. ease of implementation
2. compatibility with existing ecosystems
3. distributed computation

...</SystemOverview>

<QualityGuarantee>...

# Goal: bounding the prediction difference

The expected prediction difference:

$$\texttt{diff}(\text{full, approx}) = E_x\left[\mathbf{1}(\text{approx}(x) \neq \text{full}(x))\right]$$ (for classification tasks)

**Our goal:**

$\texttt{diff}(\text{full, approx}) \leq \varepsilon$ with high probability

e.g., $\varepsilon = 0.01 \rightarrow$ 99% same predictions

How can we estimate $\texttt{diff}(\text{full, approx})$?

# Difference in params → `diff`(full, approx)

A model is a function of parameters

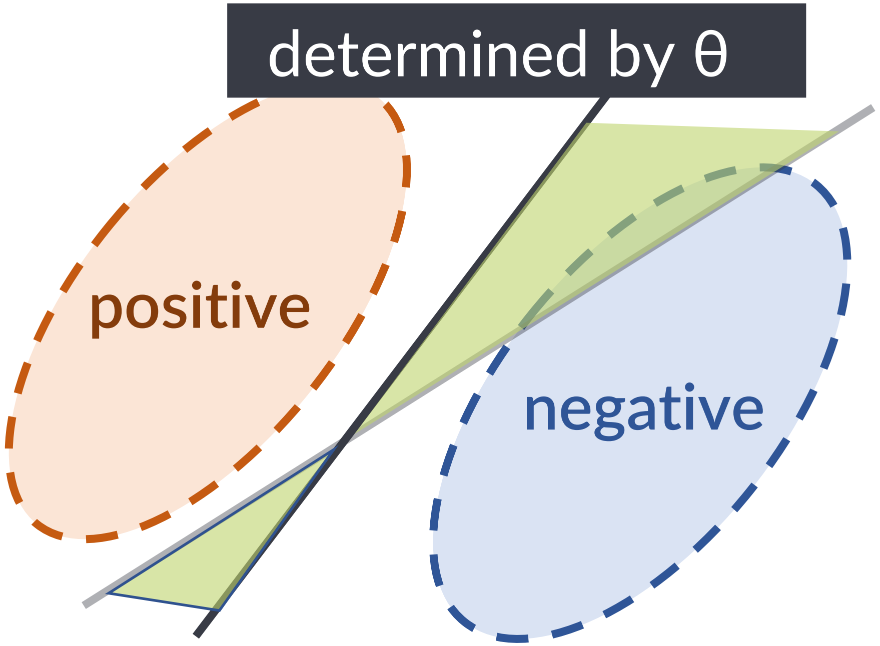
A logistic regression model predicts:

**1 (pos)** if $1/(1+\exp(-\theta^T x) > 0.5$

**0 (neg)** otherwise $f(x; \theta)$

If we know $\theta_N$ and $\theta_n$

we can infer **`diff(`**$f(x; \theta_N), f(x; \theta_n)$**`)`**




BUT, we don't know $\theta_N$. How to infer the difference?

# Infer probabilistically w/ Monte Carlo simulation

We estimate `diff`(full, approx) using **samples from Pr($\theta_N$)**

$= E_x\left[\mathbf{1}(f(x;\theta_N) \neq f(x;\theta_n))\right]$

| | $\theta_{N,1}$ | $\theta_{N,2}$ | $\theta_{N,3}$ | $\theta_{N,4}$ | $\theta_{N,5}$ |
|---|---|---|---|---|---|
| `diff`(full, approx) | 0.01 | 0.005 | 0.015 | 0.01 | 0.008 |

We say `diff`(full, approx) $\leq$ 0.01 with **80% probability (4/5)**

BlinkML uses **thousands of** samples for accurate estimation

How do we obtain **samples** from Pr($\theta_N$)?

# Obtain samples from Fisher + optimization

Based on Fisher's theory, we get:

$\theta_N \; - \; \theta_n \sim \mathit{Normal}(0, \alpha_n H^{-1} J H^{-1})$

param of full model — param of approx model — multivariate normal distribution

$H^{-1}$: model complexity

J: data variance

The size of covariance matrix, O(#features^2), makes sampling **slow**

**We employ mathematical tricks**

$z \sim N(0, I) \;\rightarrow\; L z \sim N(0, LL^T)$ **sampling = matrix multiplication**

We obtain L such that $LL^T = H^{-1}JH^{-1}$ directly from gradients using the information matrix equality

# Recap of our quality guarantee mechanism

For a certain sample size n:

1. Obtain a parameter $\theta_n$ and a factor L
2. Obtain samples of full model parameters $\theta_N$
3. Compute many **`diff(`**full, approx**`)`** using samples
4. Ensure **`diff(`**full, approx**`)`** $\leq \varepsilon$ with high probability

We must train an approximate model to obtain $\theta_n$

**In our paper:** performs this by training **at most TWO approx models**

...</QualityGuarantee>

<Experiments>...

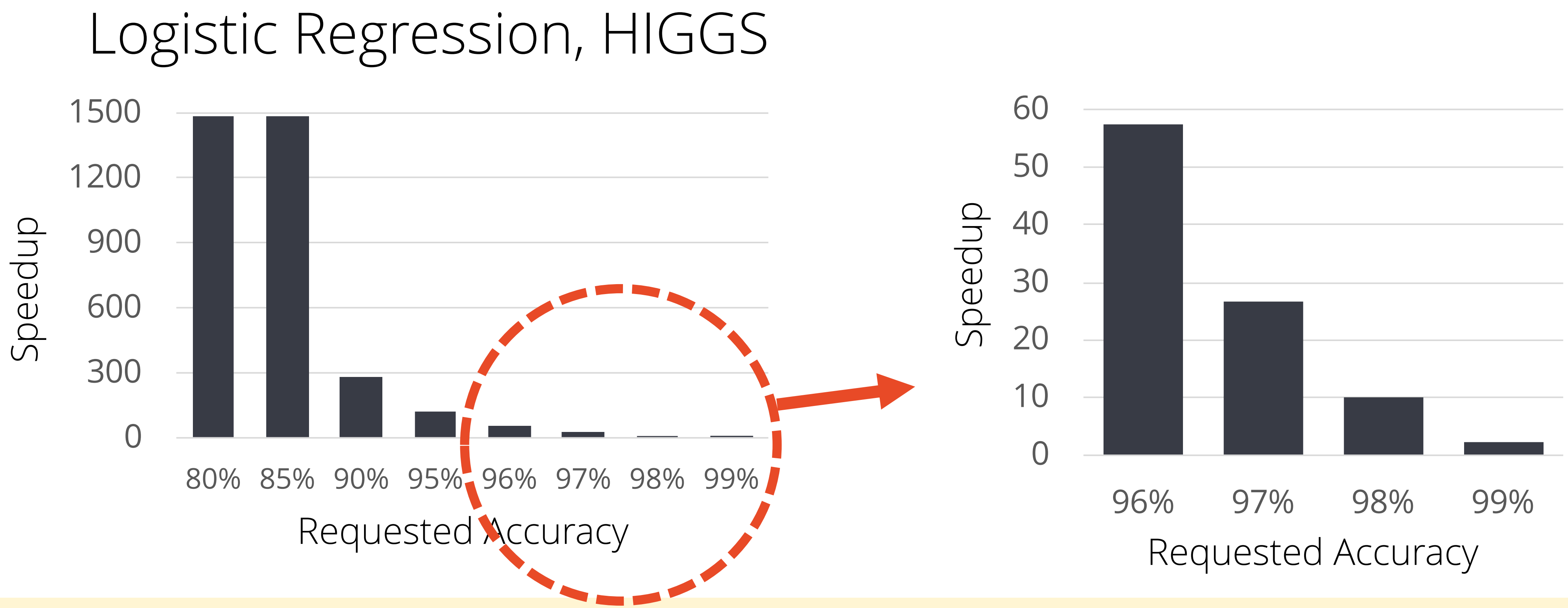
# Models and datasets

| Model | Dataset | # of examples | # of features |
|---|---|---|---|
| Linear Regression | Gas | 4M | 57 |
| | Power | 2M | 114 |
| Logistic Regression | Criteo | 46M | 998,922 |
| | HIGGS | 11M | 28 |
| Max Entropy Classifier | MNIST | 8M | 784 |
| | Yelp | 5M | 100,000 |
| Probabilistic PCA | MNIST | 8M | 784 |
| | HIGGS | 11M | 28 |

Publicly available **GB-scale** machine learning datasets

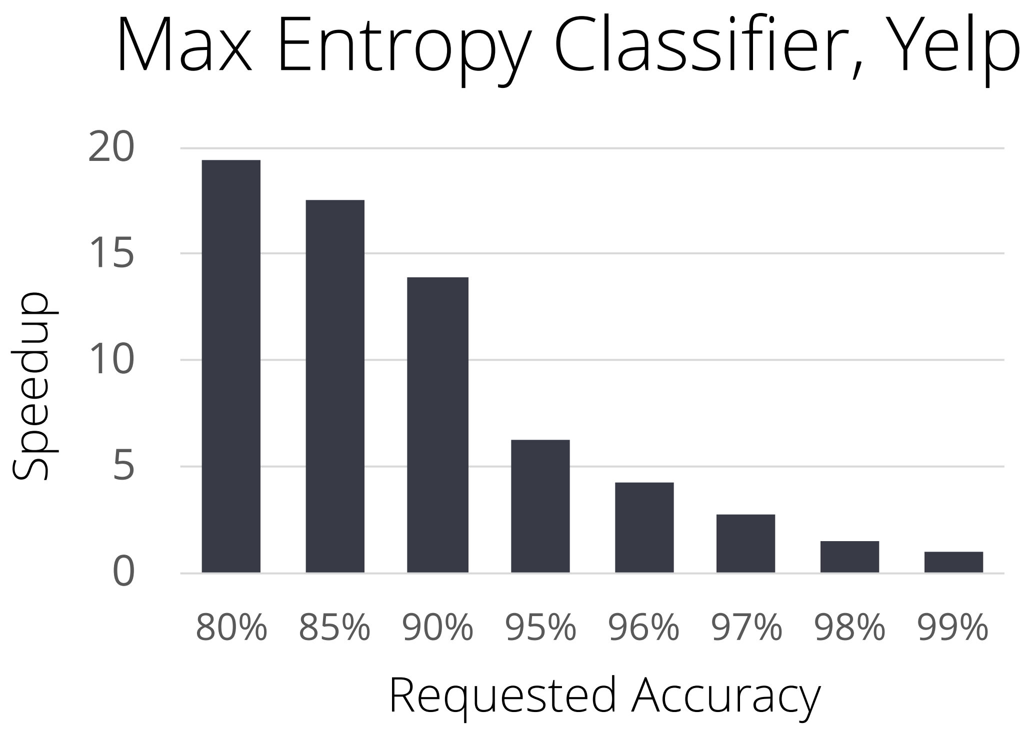
The number of features range from 28 **to 1 million**

# BlinkML offers large speedups




Speedups adjust based on requested accuracy

# BlinkML offers large speedups

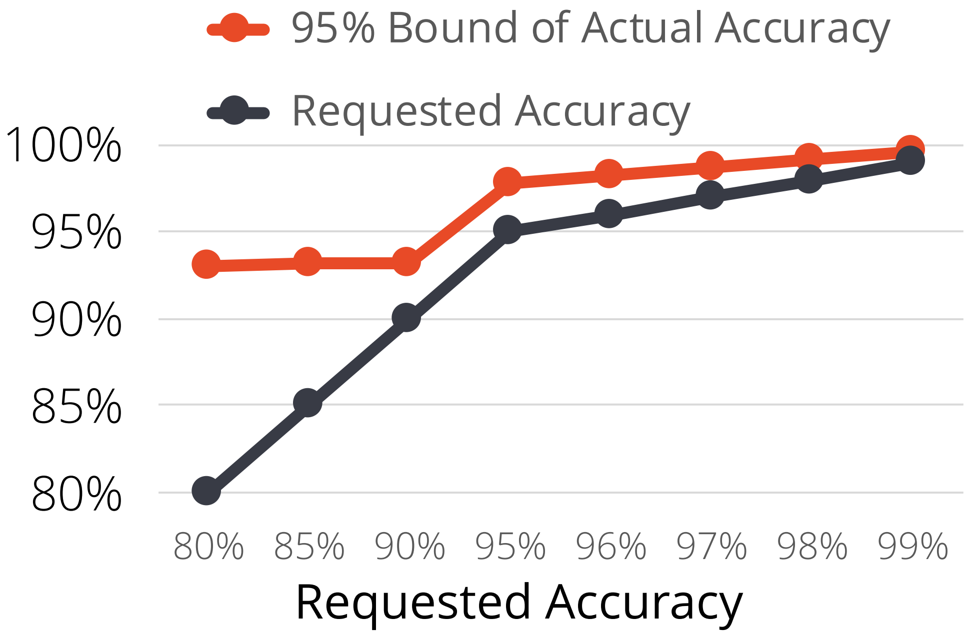




**Speedups adjust based on model/data complexity**

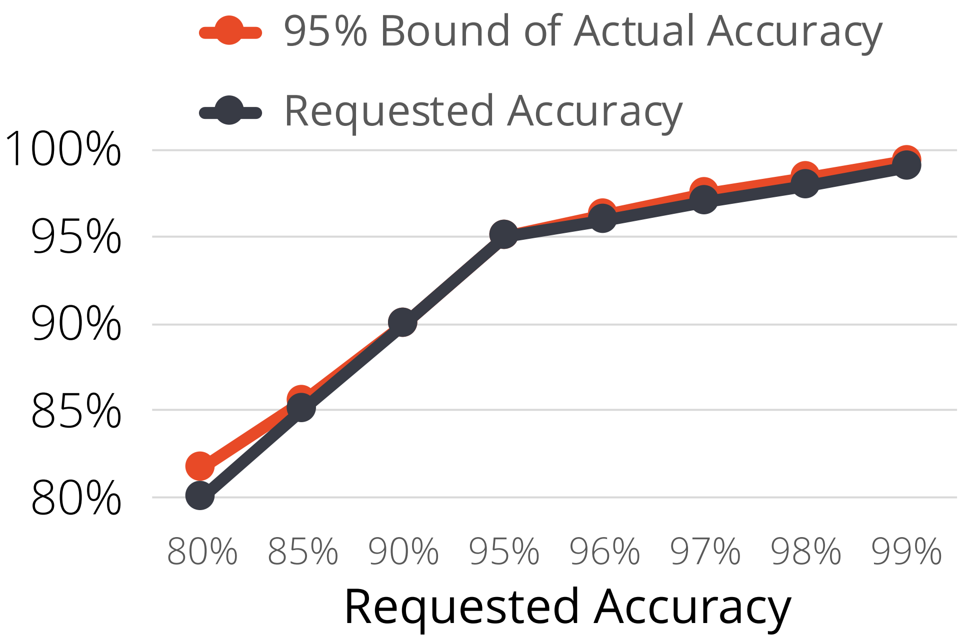
(see more systematic study in our paper)

# Approx. models satisfy requested accuracy

## Logistic Regression, HIGGS

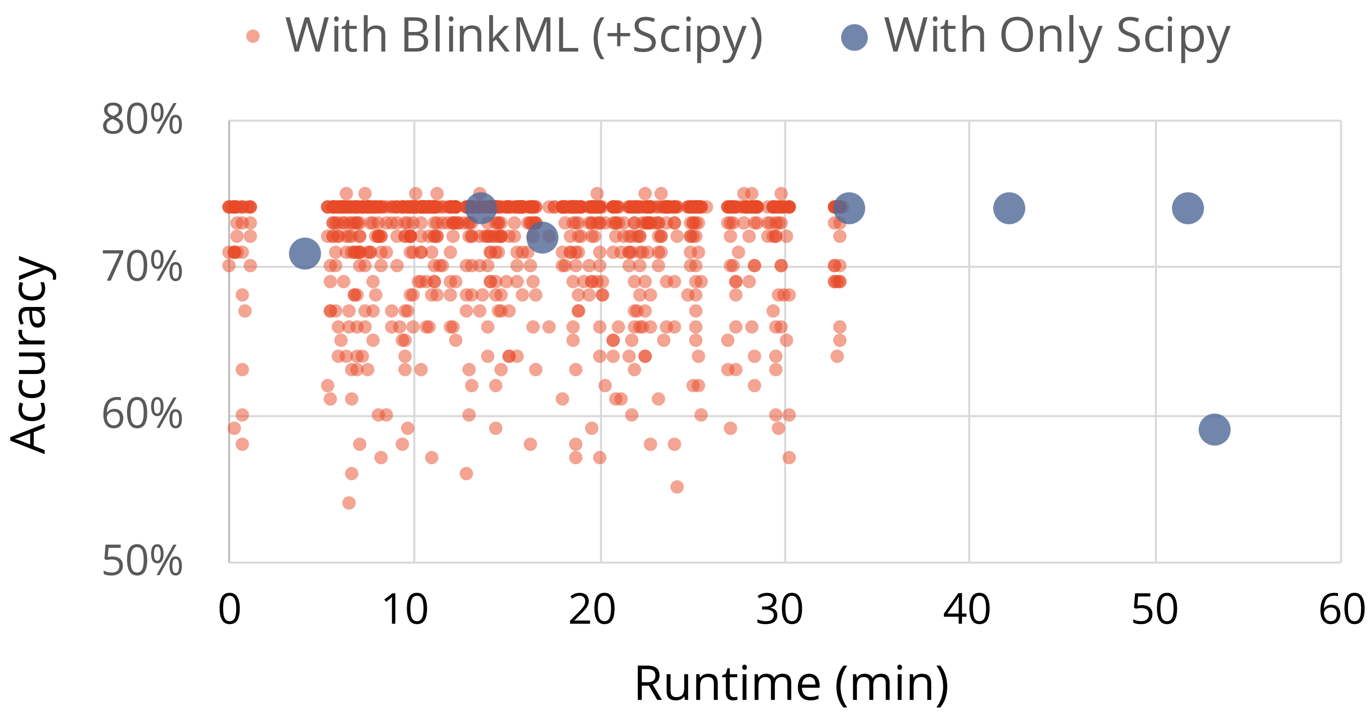



## Max Entropy Classifier, Yelp




Accuracy guarantees were **conservative** (which is not bad)

# Faster hyperparameter searching with BlinkML




Logistic Regression, Criteo

**Regular**
3 models in 30 mins

**BlinkML**
961 models in 30 mins
(sample size: 10K–9M)

**BlinkML** found the best model at itr #91 (in 6 mins, test acc 75%)

**Regular** could not find it in 1 hour

...</Experiments>

# Summary

1. Extended sampling-based analytics to commonly used ML

2. Our approach offers probabilistic quality-guarantees

3. **Core:** uncertainty on params $\rightarrow$ uncertainty on predictions

4. Empirical studies show that *min sample size* automatically adjusts

# What's next?

Can we extend this approach to other models?

- SVM, ensemble models, deep neural nets, …

Run BlinkML directly on SQL engines?

- Relational DBs are well optimized for structured data
- No need to move/migrate data

Propagating errors to downstream applications

- Formal semantics required

Thank you!