Solving Regression Problems with Machine Learning

lessons learned from learning machine

now with GPU optimized models

Regression Problems in Astronomy

we want to determine parameters

- metallicity, starburst ratio, redshift

... but detailed analysis is too expensive

- observation time / telescope time → spatial problem

high spatial resolution	→	low sky coverage
high spectral resolution	→	long integration time
high time resolution	→	low sensitivity

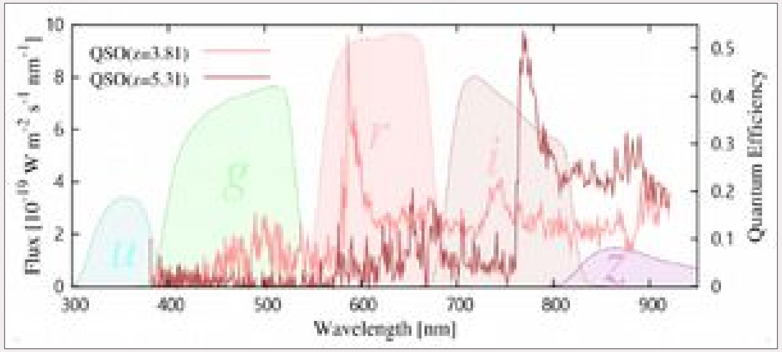
analysis of large catalogs demands solving of regression problems

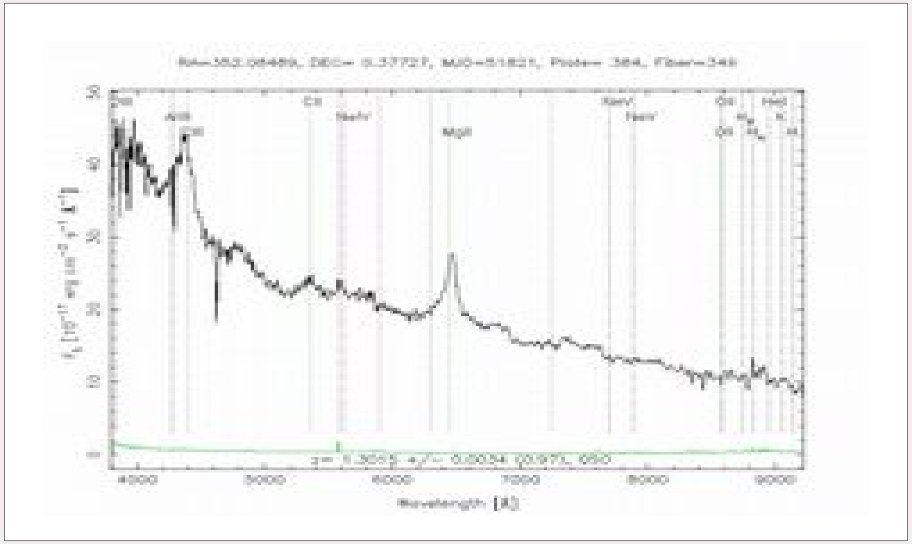
$$f(\vec{x}) \rightarrow y, \text{ where } \vec{x} \in \mathbb{R}^n, y \in \mathbb{R}$$

Photometric Redshifts

galaxy and quasar redshifts

- simple relation $$1 + z = \frac{\lambda}{\lambda_0}$$
- allow to understand early universe and find interesting objects
- SDSS data (DR7), broadband photometry (*u, g, r, i, z*)
  - 10k degree²
  - 0.3 billion objects
  - 1.2M spectra





Nearest Neighbor Models

use k-Nearest Neighbors / local model

$$\hat{Y}(\vec{x}) = \frac{1}{k} \sum_{\vec{x}_i \epsilon N_k(\vec{x})} y_i$$

- works fine in high dimensions (>3 but < 50)
- no physical assumptions required
- good reference samples available
- want to deal with missing values? → change $\boxed{N_k(\vec{x})}$!

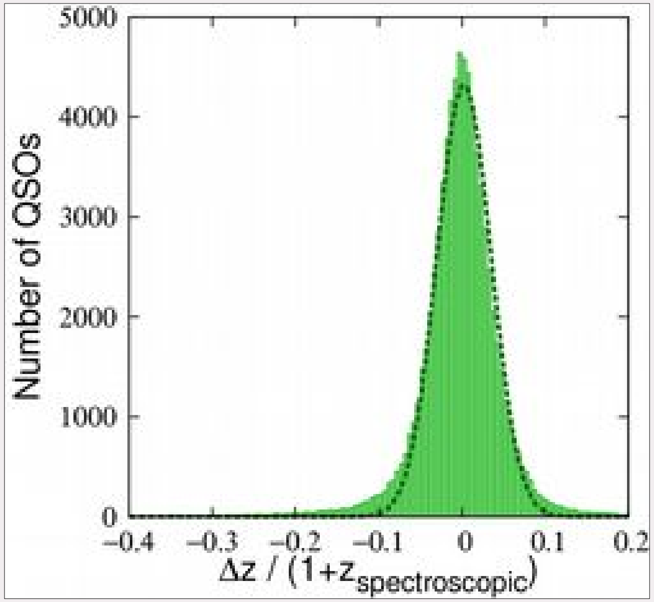
$$\sqrt{\sum_{j=1}^{n} \Delta x_j^{\,2}} = \sqrt{\Delta x_1^{\,2} + \Delta x_2^{\,2} + \ldots + \Delta x_{missing}^{\,2} + \ldots + \Delta x_n^{\,2}},$$

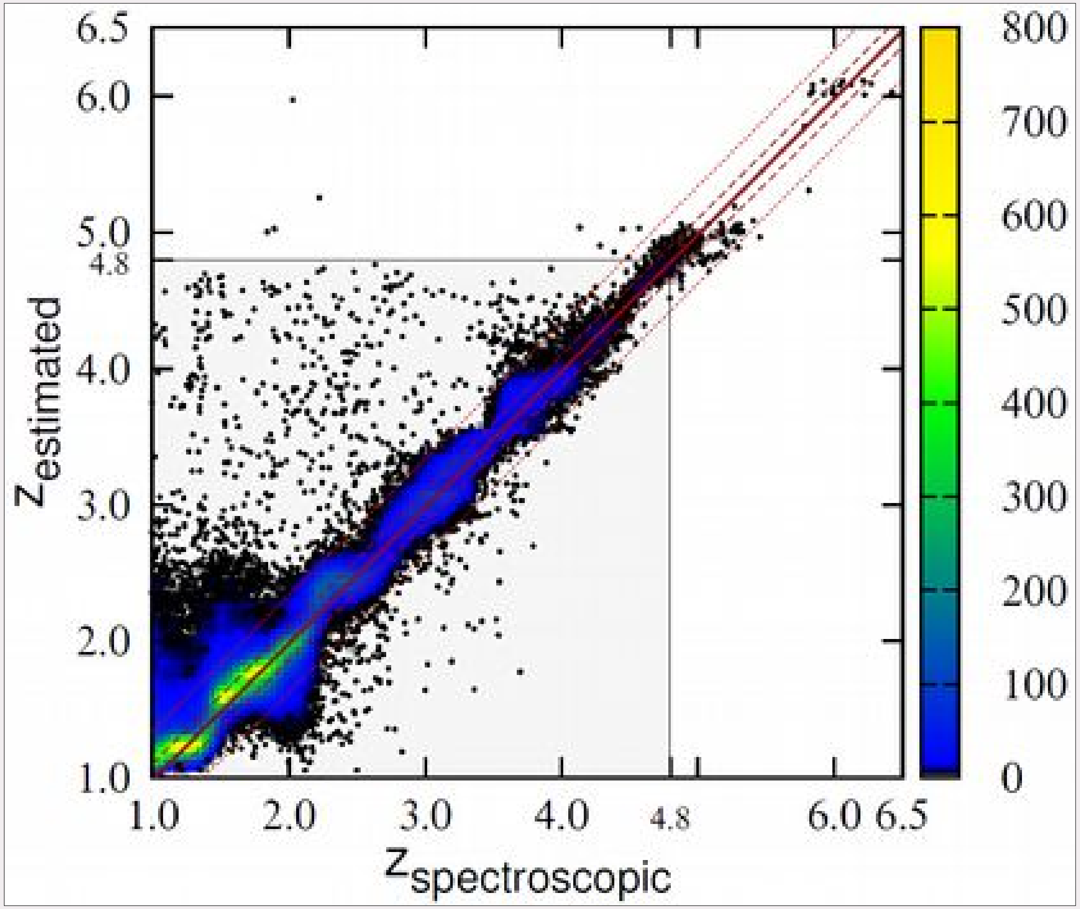
$$with\; \Delta x_{missing}^{\,2} = \lambda \frac{1}{n-1} \sum_{j=1..n,\; j \neq missing} \Delta x_j^{\,2}$$

Nearest Neighbor Models

applied to all quasars with spectra, SDSS DR7

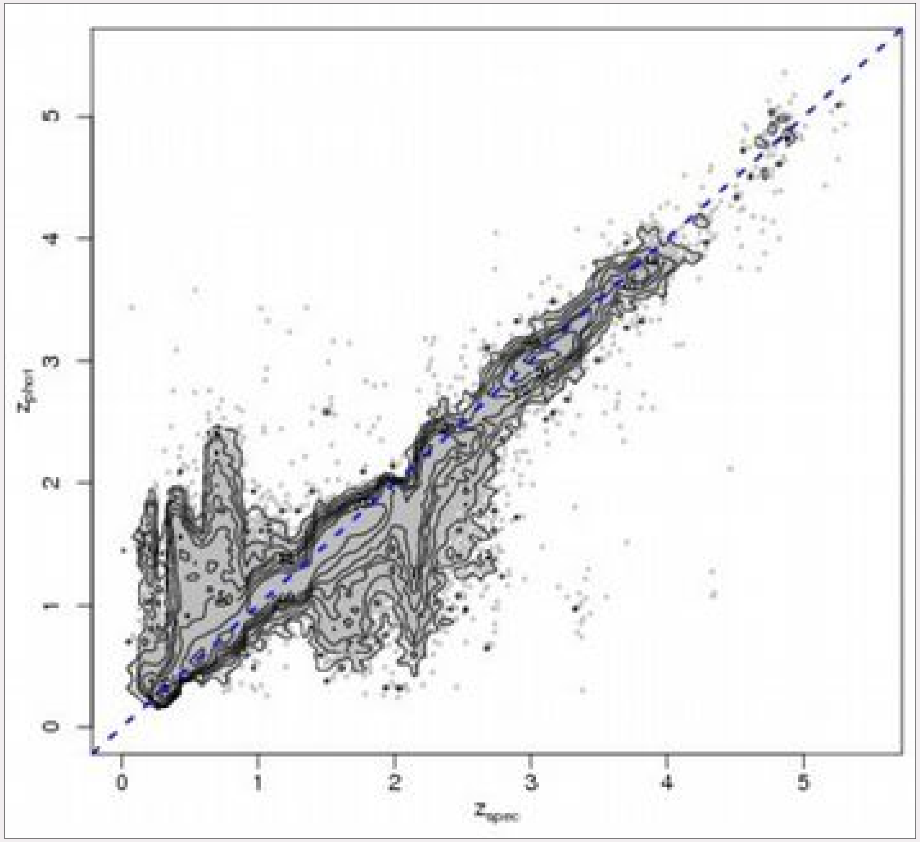
$$\sigma \frac{\Delta z}{1+z} = \sigma \Delta z_{norm} = 0.033$$





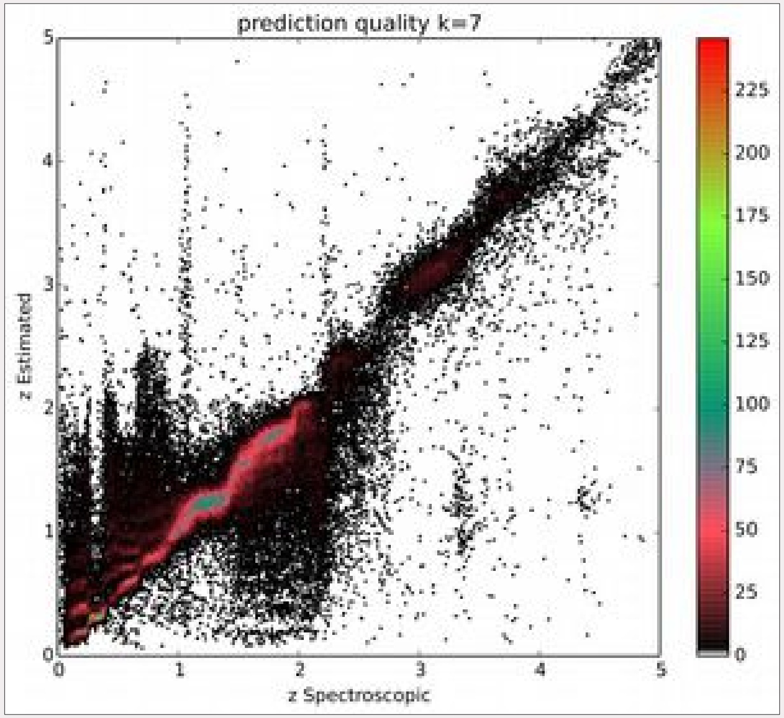
Polsterer et al. 2013

Existing Models



Laurino et al. 2011

$$RMSE(\Delta z_{norm}) = 0.19$$
$$MAD(\Delta z_{norm}) = 0.041$$



$$RMSE(\Delta z_{norm}) = 0.25$$
$$MAD(\Delta z_{norm}) = 0.048$$

Parallel Feature Selection

missing error values in model ...

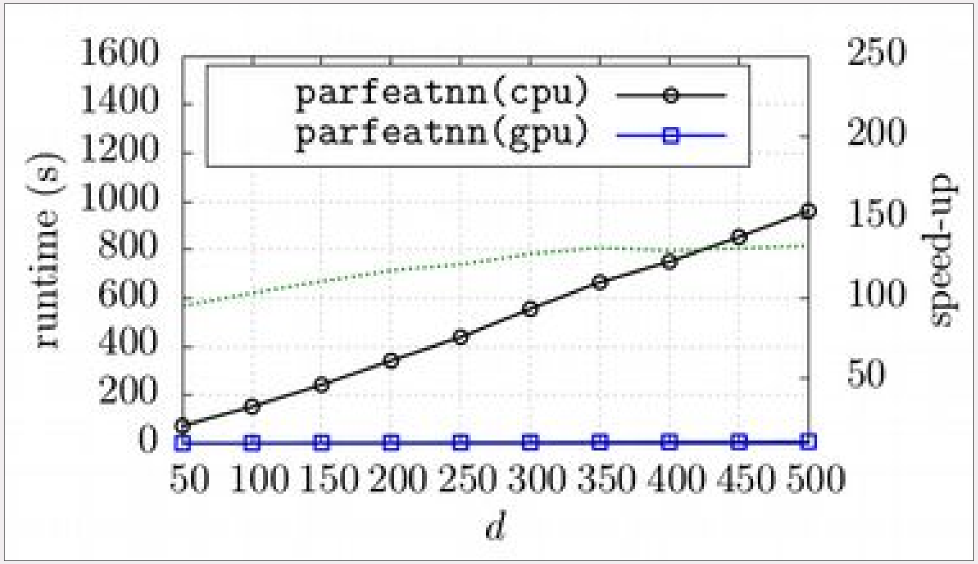
... test different feature combinations!

- building and testing one model = 100 sec.
  - huge dataset + bad python implementation

do it in parallel on a GPU

- used openCL
- matrix update operation

improved reference sets



Gieseke et al. 2014

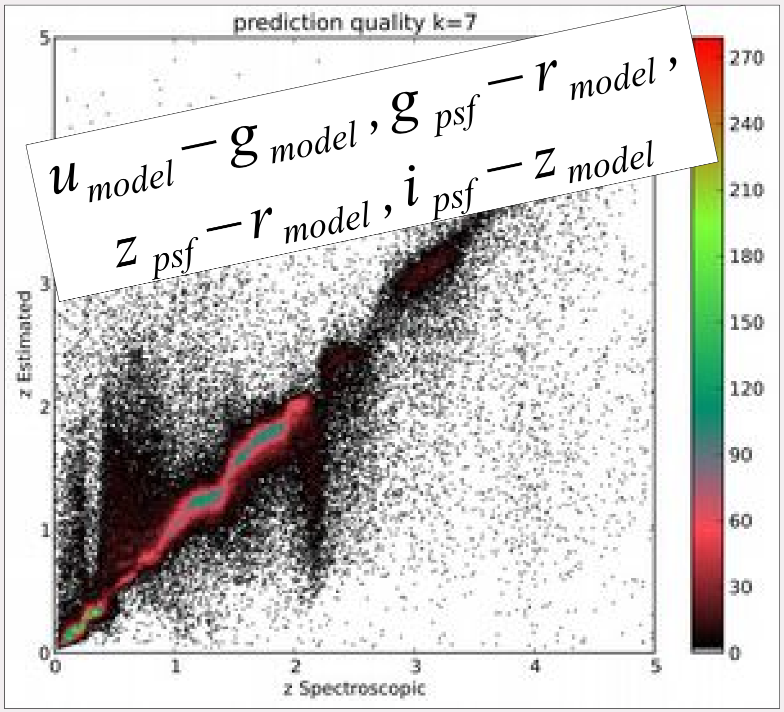
Complete Test

what are the best 4 features?

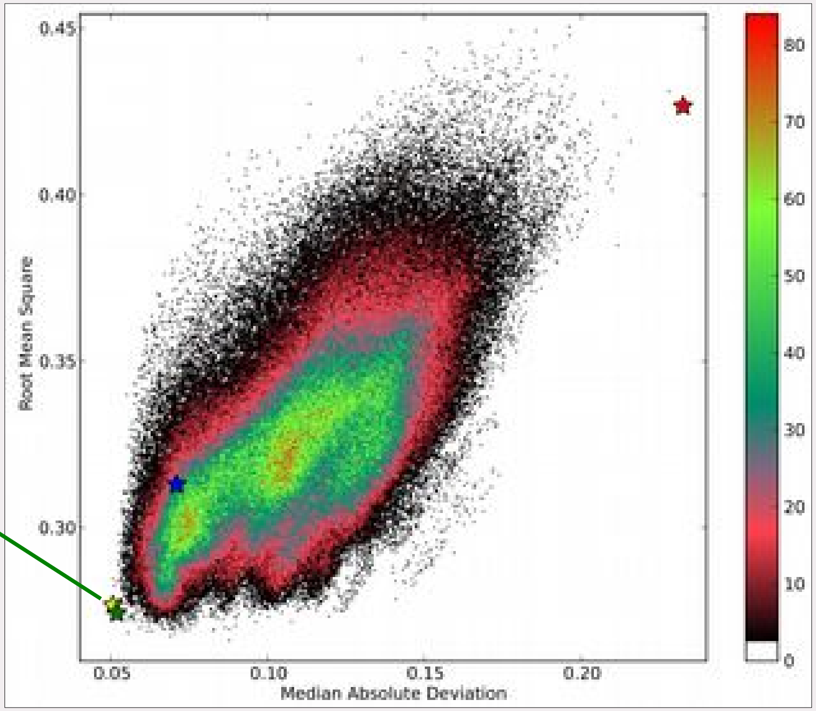
- psf and model magnitudes in (*u, g, r, i, z*)
  - 10 raw features + 45 colors = 55 features
- 395 days with old code → now just 3 hour, on 1 GPU

$$\frac{n!}{(n-r)!\,r!}, \textit{with } n=55, r=4$$

$$\rightarrow 341{,}055 \textit{ combinations}$$



$$u_{model} - g_{model},\ g_{psf} - r_{model},$$
$$z_{psf} - r_{model},\ i_{psf} - z_{model}$$



Forward Selection

can we be even better?

- psf, model and petrosian magnitudes in (*u, g, r, i, z*) and errors
- extinction

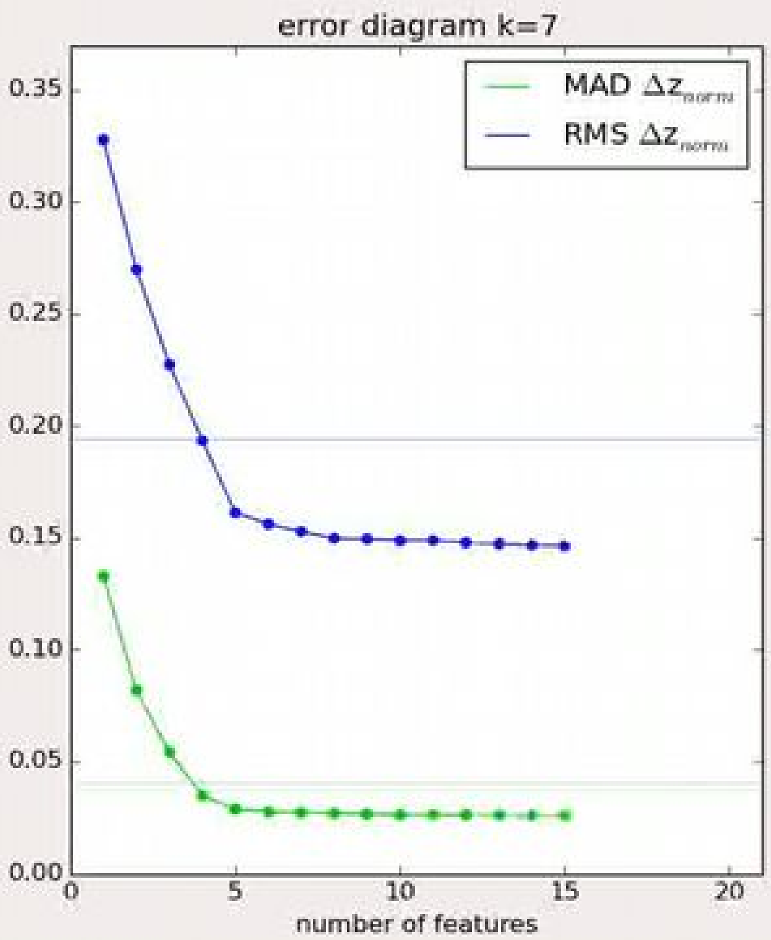
585 features

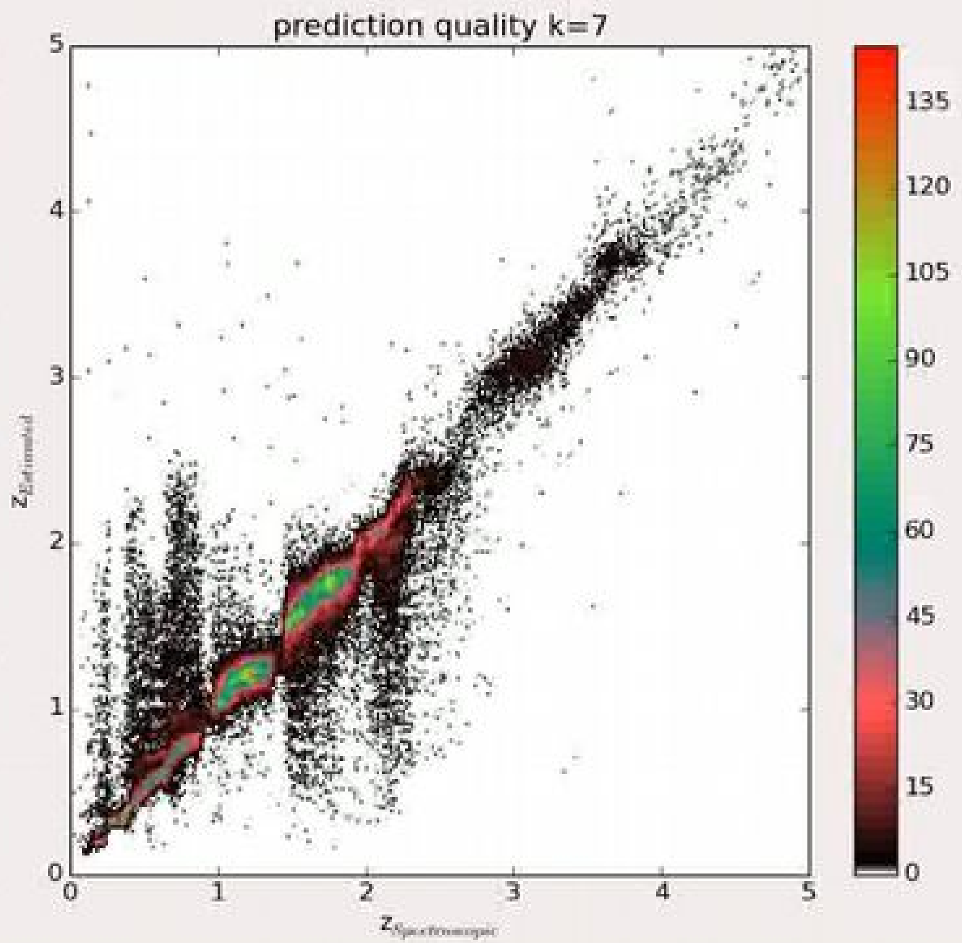
best 10 out of 585 → 1,197,308,441,345,108,200,000

we need a better strategy!

Forward Selection

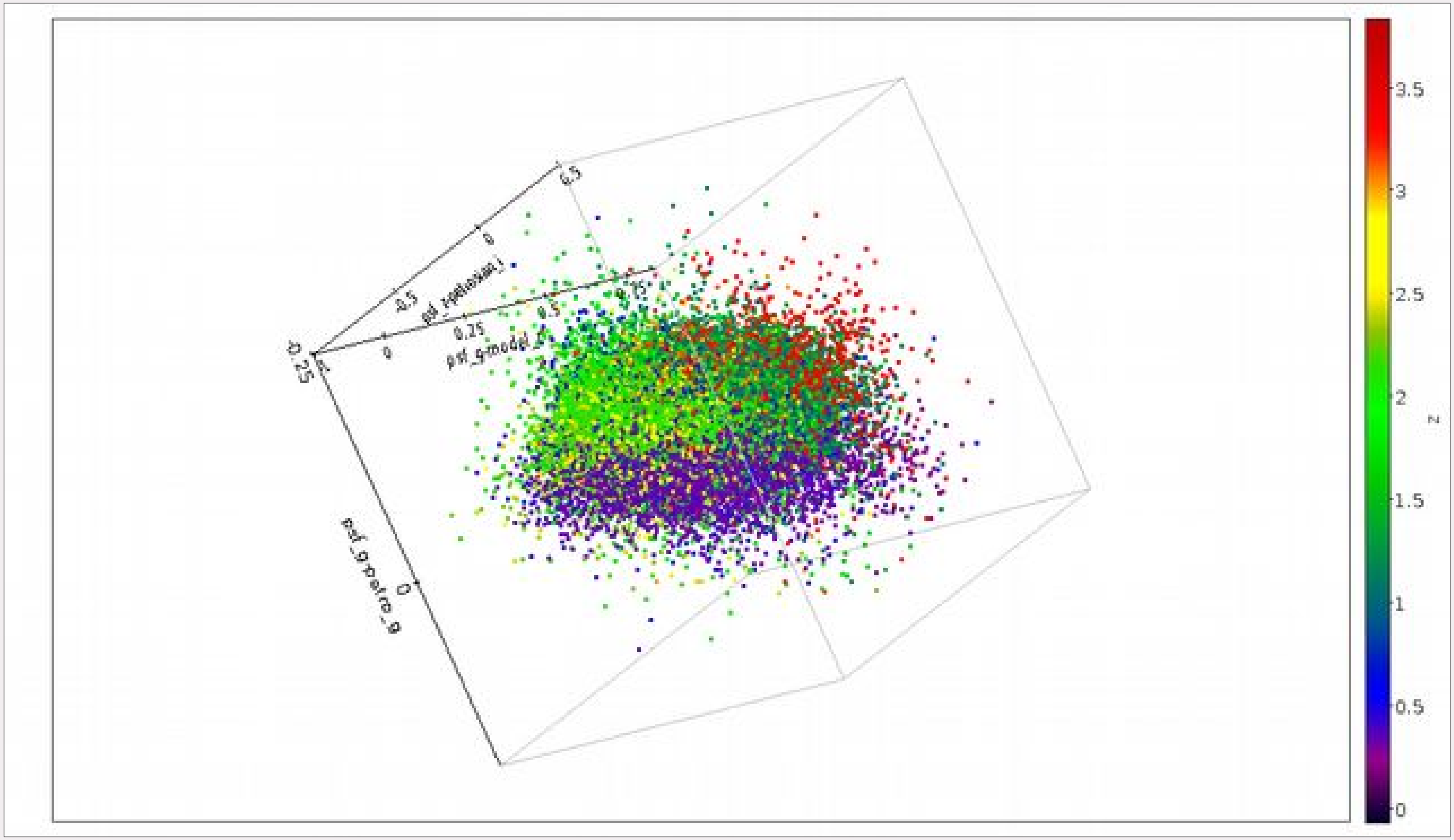
apply greedy forward selection





Forward Selection

resulting features:



Lessons Learned

new features …

… optimized for machine learning

evaluate features …

... to optimize instrumentation, surveys

comparable sets to have competition …

… publish data and method

different database access …

... optimized for machine learning


thanks for a great week in Sydney