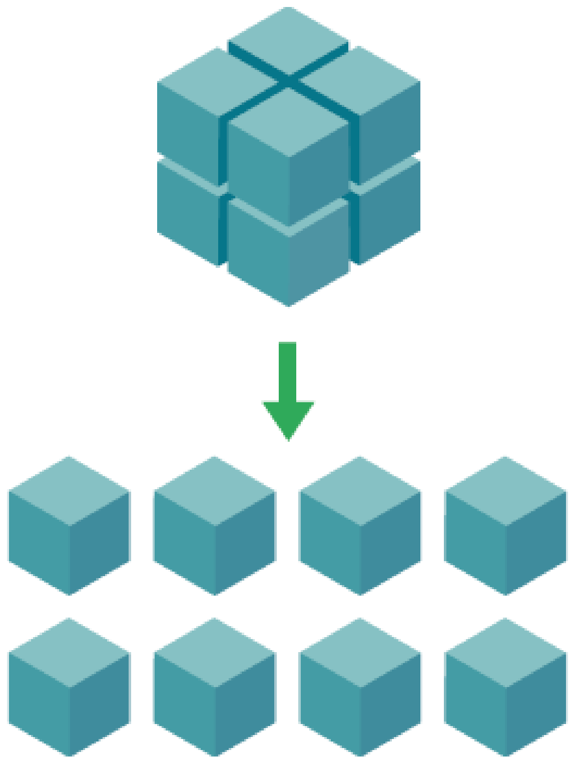
WLW (^3^)

Shiying Li

Tianlu Wang

Yifei Wang

Preprocessing



- The cube represents the 3D MRI image
  - Divide the image into smaller parts

Why?

Research showed that brain age related directly to some specific part of brain. So divide brain into smaller parts would help us to locate this specific part and help to extract high quality features.

So how many parts should we cut the brain up?

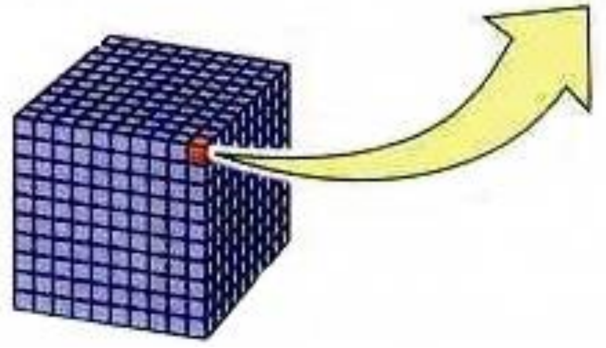
Coarse ↔ Noise trade off

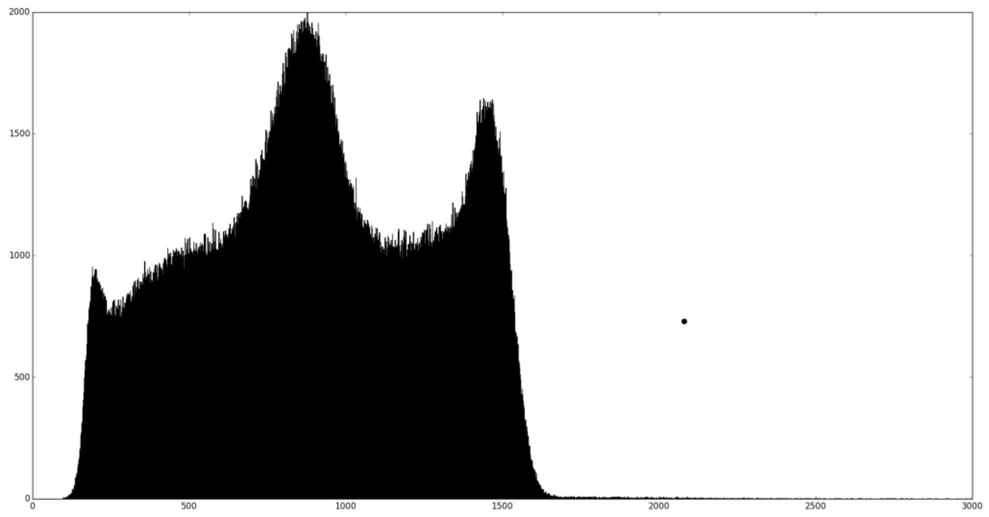
Just try!

We divide the brain equally in 9 parts for each slide.
That is 9*9*9 parts in total.

Feature selection

Use histogram as feature





Divide the histogram (according to pixel value) in several bins (we use 45). Then calculate how many points in this part of the image fall into every bin.

Do the same for the rest of the parts.
So for one image we have 9*9*9 histograms as feature.

Repeat this procedure for the rest 277 training images and 138 test images.

Model and Optimization

- We adopted Bayesian Ridge regression model;
  - Why? It is better because of its prior setting;
  - Actually, linear and nonlinear regression are both tried and Bayesian ridge has shown its advantage when tested online.

- Feature selection from Sklearn package is also used to lower the dimension of features from 32,805 to 1,500.

Tips we might benefit from

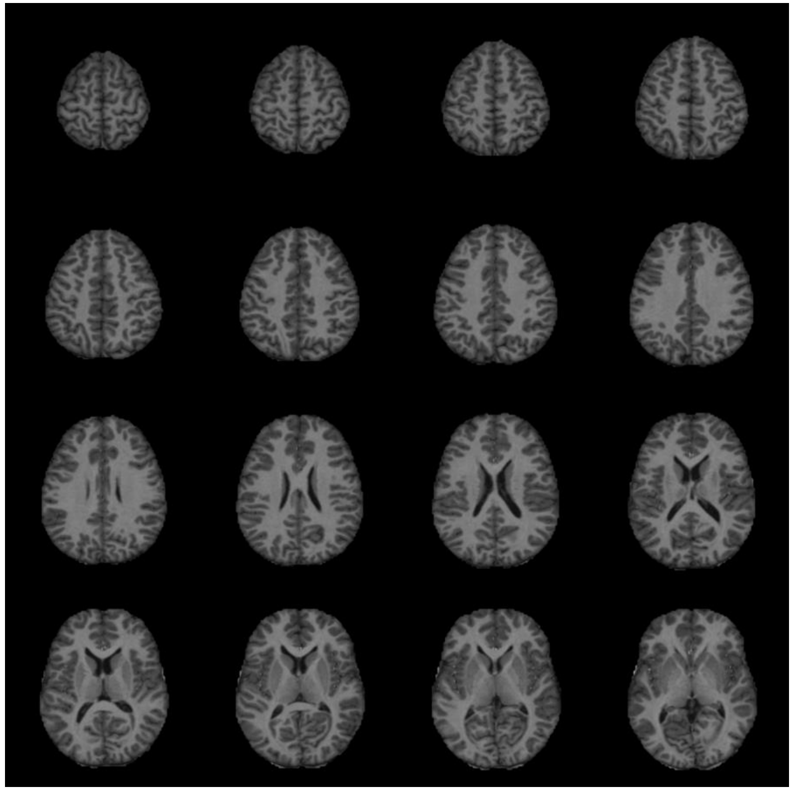
- Try everything that might be useful;
  - For the regression model, we have tried Linear Regression, Lasso, Bayesian Ridge, Gaussian Process and…;
  - Do not stick to one method just bacause of its theoretical possibility.
- It is good to pursue accuracy but please not to be overfitting;
  - It is hard;
- Please record everything that you have tried!!!
  - Just in case that you waste your time trying again;
  - More convenient for communication among team members.

Yohan Thibault

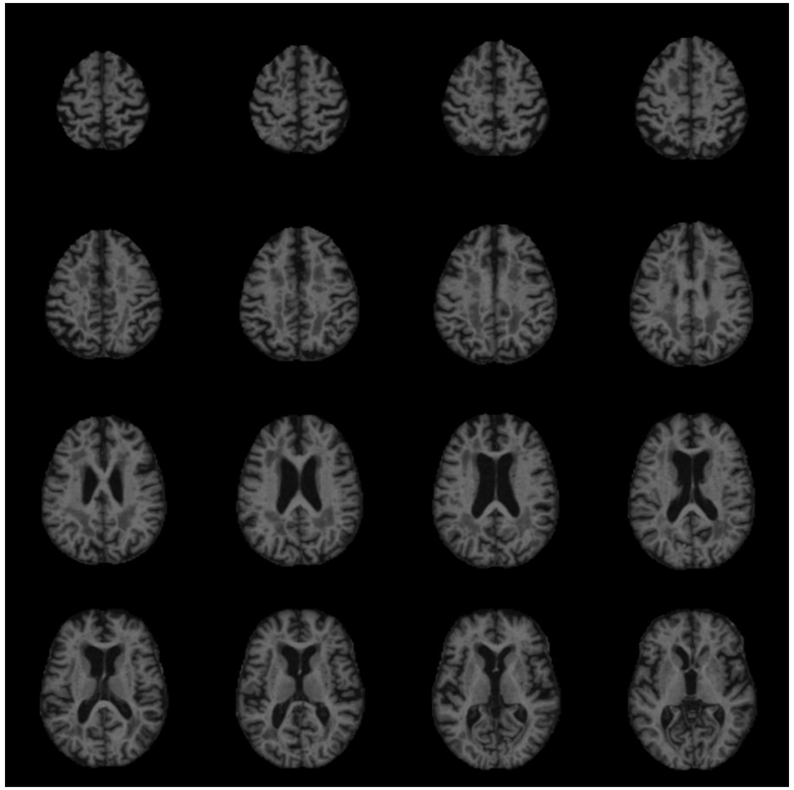
Yohan Thibault

Data analysis

Age of the brain: 20

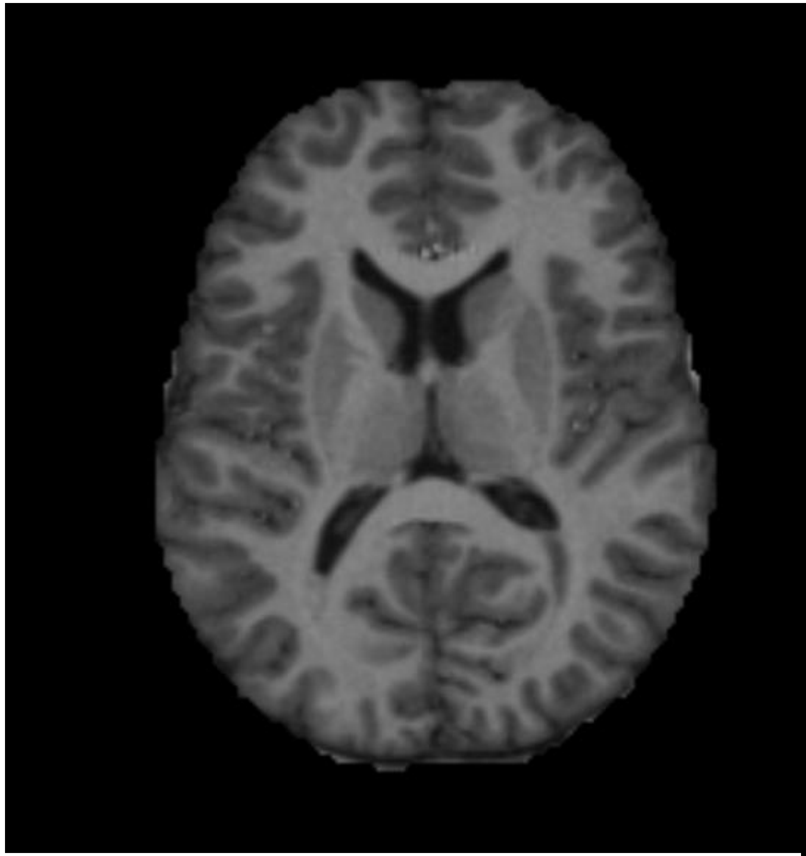


Age of the brain: 84

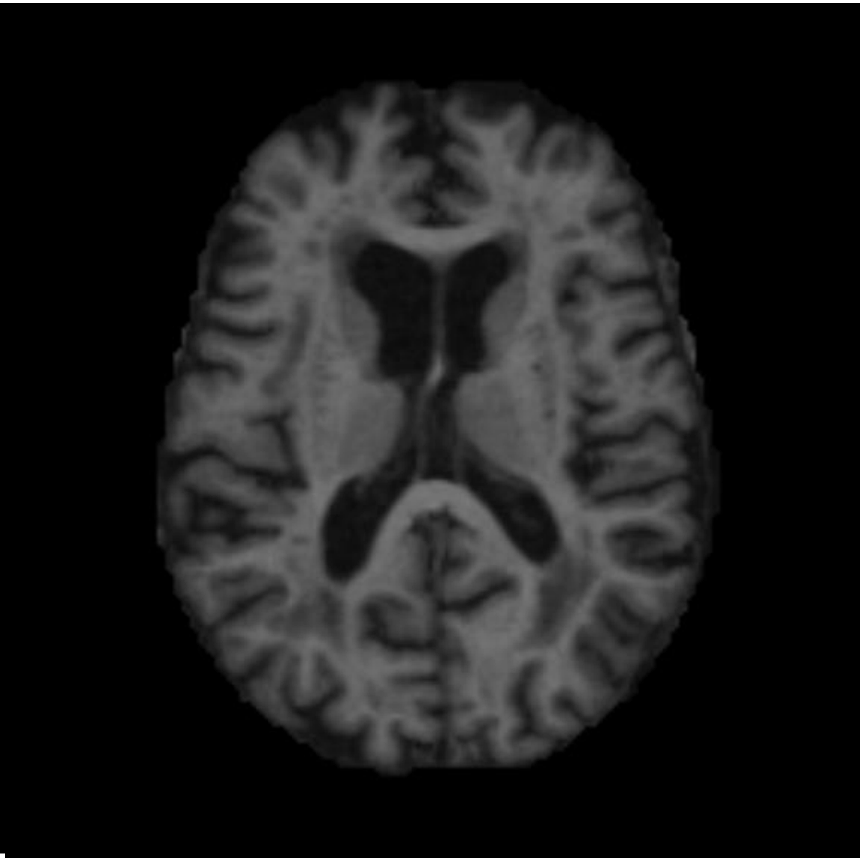


Data analysis

Age of the brain: 20



Age of the brain: 84



Preprocessing/features extraction

- About 1 600 000 non zeros voxels
  - To much information for the number of sample.
  - Reduce dimension by feature extraction
- First idea:
  - Mean, standard deviation and median
  - Strong against noise and very low dimension.
- Improvement:
  - Splitting the first image into 27 sub-images to capture different variation in different part of the brain.
- Second idea:
  - Canny filter: perfect to capture the strong local variation.
  - Simply counting the number of edges per areas.

Canny filter

- An edge detection algorithm based on intensity gradient.
- Look on wikipedia, it is very basic.

Input image Low threshold Medium threshold High threshold

Learning process

- Using the python library ‘sklearn’ and the linear model: LassoLars.
  - Weak to noise (but my feature are strong to noise)
  - Good when dimension > sample.
- Use the precomputed vectors of features.
- Manually search for the best parameters.
  - This has to be done automatically for the next project (cross validation)
- Hope that my C++ code has produced non corrupted files.

Postprocessing

- Predict the result for several sets of features
    - Features from mean, standard deviation, median.
    - Features from Canny edges detector.
    - Different numbers of sub-image (27, 64 and 125)
- Average the obtained results.
- Submit

Remarks

- There exists lots of complex image processing that would give better features:
  - Image segmentation (watershed).
  - Template matching.
  - Fourier transform (wavelet)
- Time spend on finding good feature is well spend.

Orbuculum

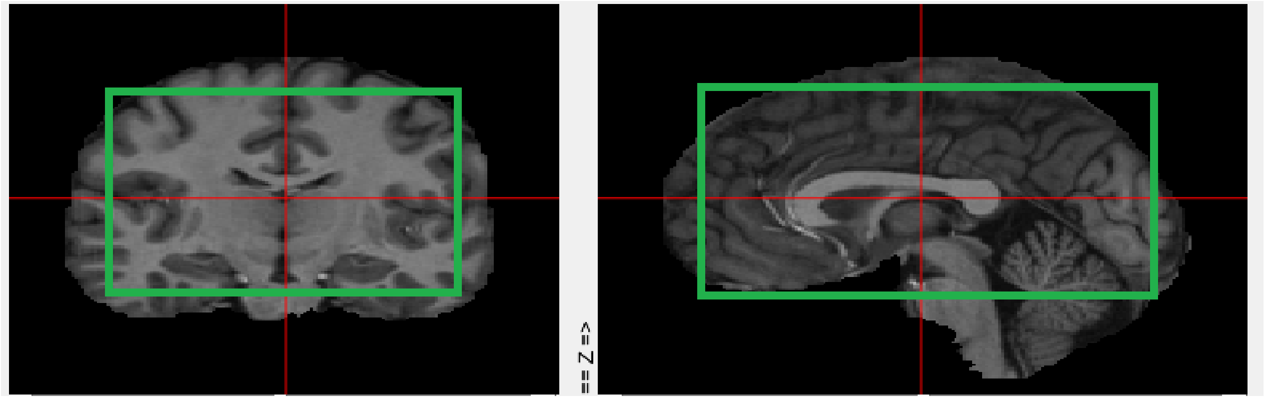
Nikolaos Kolitsas

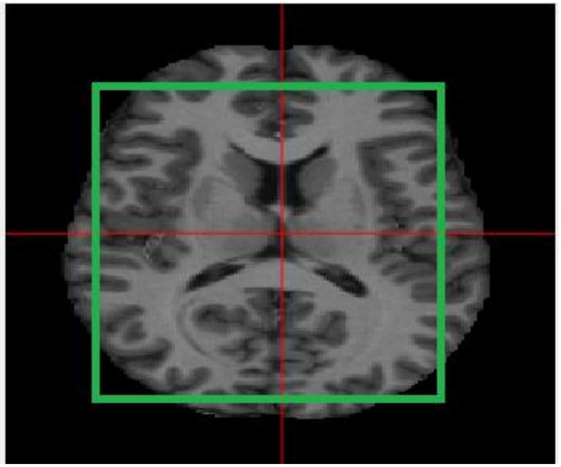
Dejan Mircic

Ingo Schilken

Preprocessing

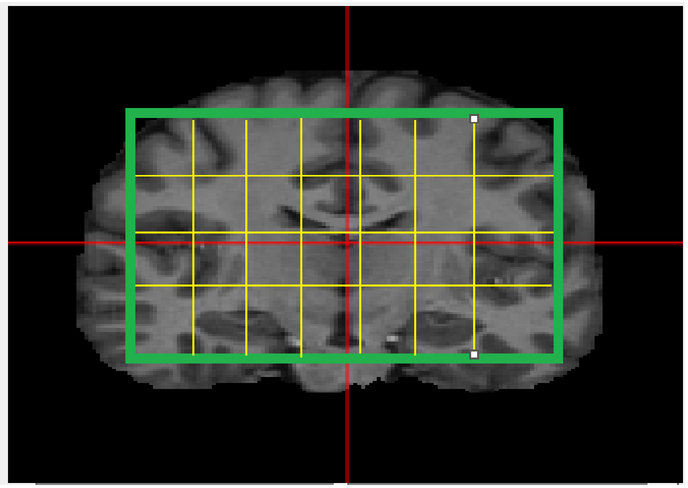
- Cut away the outer voxels to remove the black areas and outer areas of the brain





Feature extraction

- Partition the remaining volume into small cubes
- Create intensity histograms and concatenate them



Model / optimization

- Gradient boosting
- Least squares loss function

- K-Fold cross validation
- Grid search

Things to consider

- Non uniform histograms bin widths
- Avoid overfitting by looking at the CV mean and standard deviation

Other attempts

- PHOG features
- Bag of visual words using SURF features
- PCA dimensionality reduction
- Pixel averaging over small cubes

- SVM
- LASSO
- Bayesian ridge regression

General Lessons from MLP1

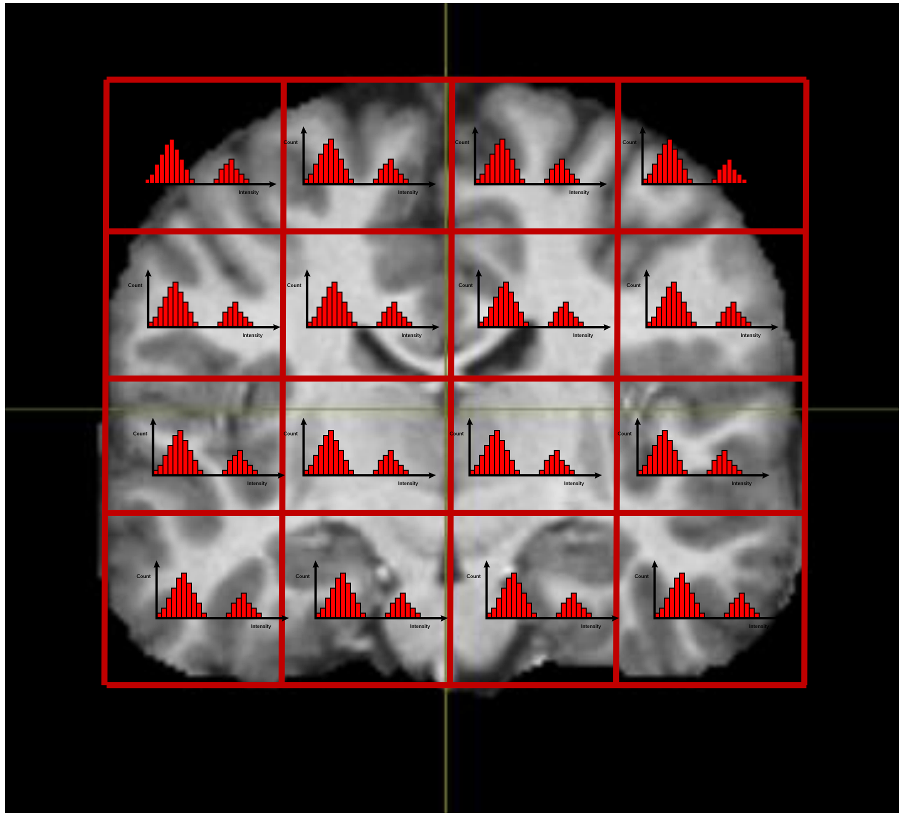
Top 10

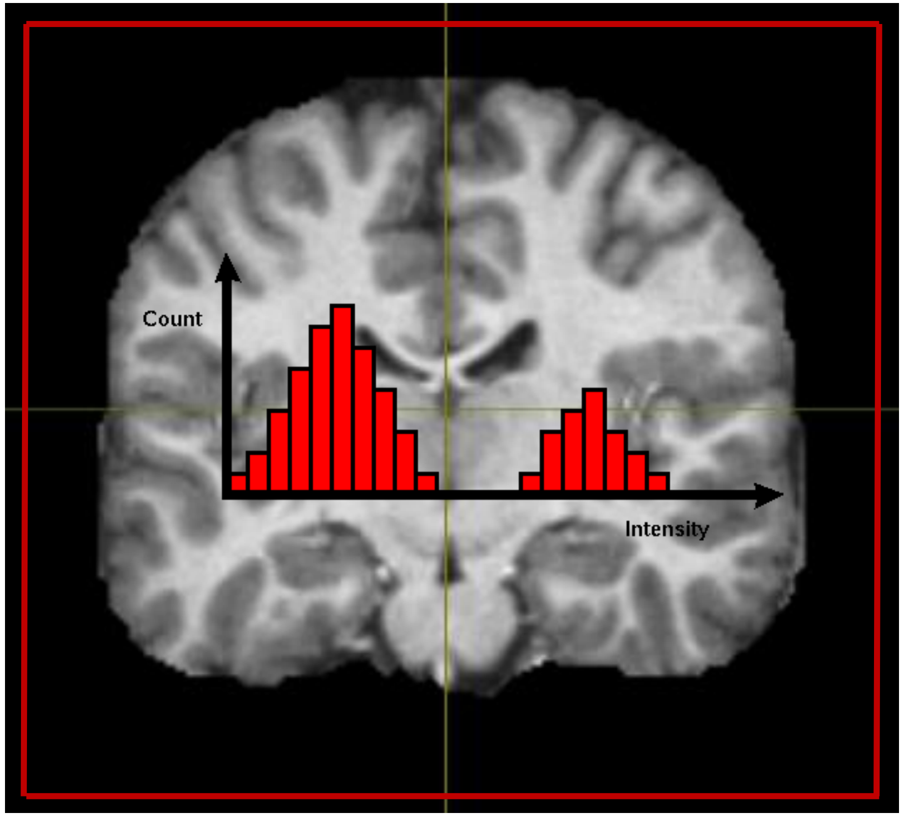
	PREPROCESSING	FEATURES	MODEL	OPTIMIZATION
1	crop, subsample	histograms	gradientboosting	crossvalidation
2	crop, subsample	multiscale, cannyfilter	lasso	
3	subsample	histograms, fscore	bayesian ridge	
4		histograms		crossvalidation
5			3d conv net	
6		histograms, multiscale	SV regression	crossvalidation
7		pca	linear regression	
8	subsample	histograms, pca	linear regression	
9		histograms	linear regression	
10	normalize	histograms, pca, vscore	ridge	crossvalidation

Bottom 10

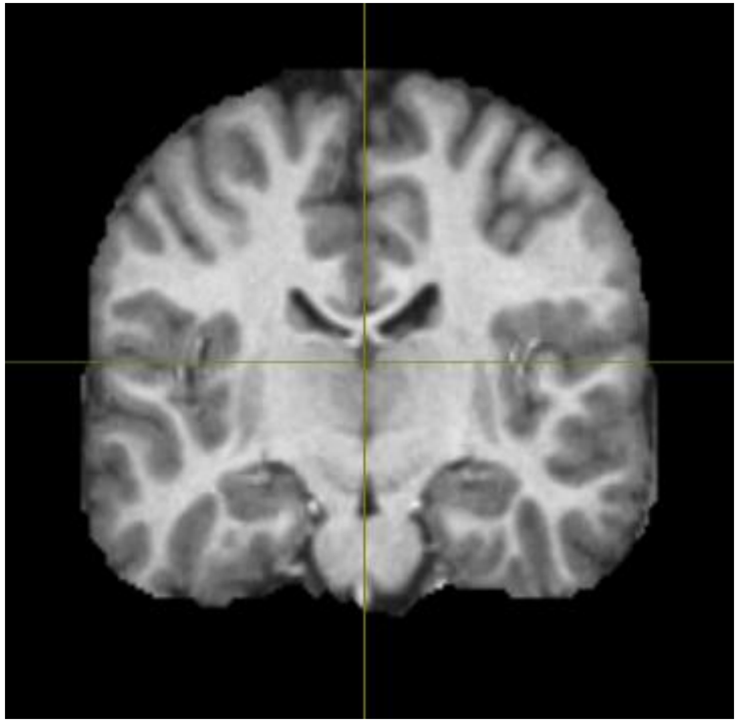
	PREPROCESSING	FEATURES	MODEL	OPTIMIZATION
-1	subsample	histograms, pca	ridge, lasso, SV regression	crossvalidation
-2	crop	histogram	gradientboosting	
-3				
-4		pca	neuralnet	
-5			ridge	crossvalidation
-6	crop	histogram, ascore	randomforrest	
-7	crop	threshold, mean	randomforrest	
-8	subsample		linearregression	
-9	filter, maxintensity	histogram	kNN	
-10	subsample		ridge	

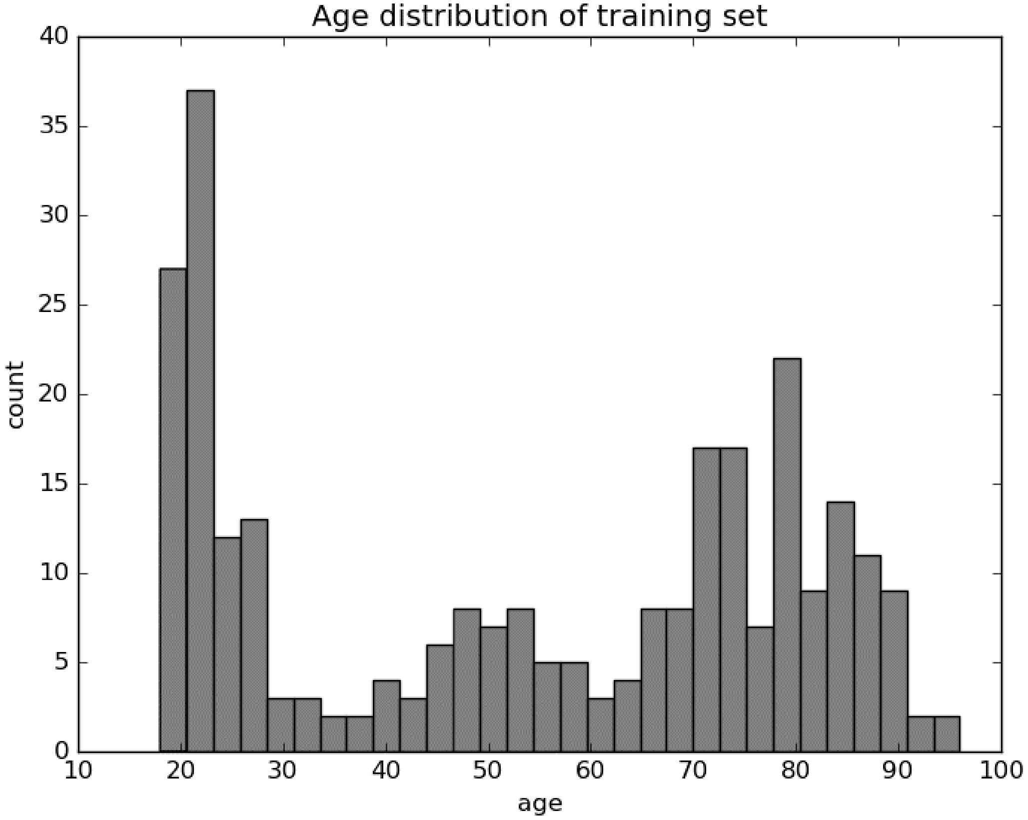
Top 10 vs. Bottom 10





Look at the data





Model Selection

A **or** B **or** C

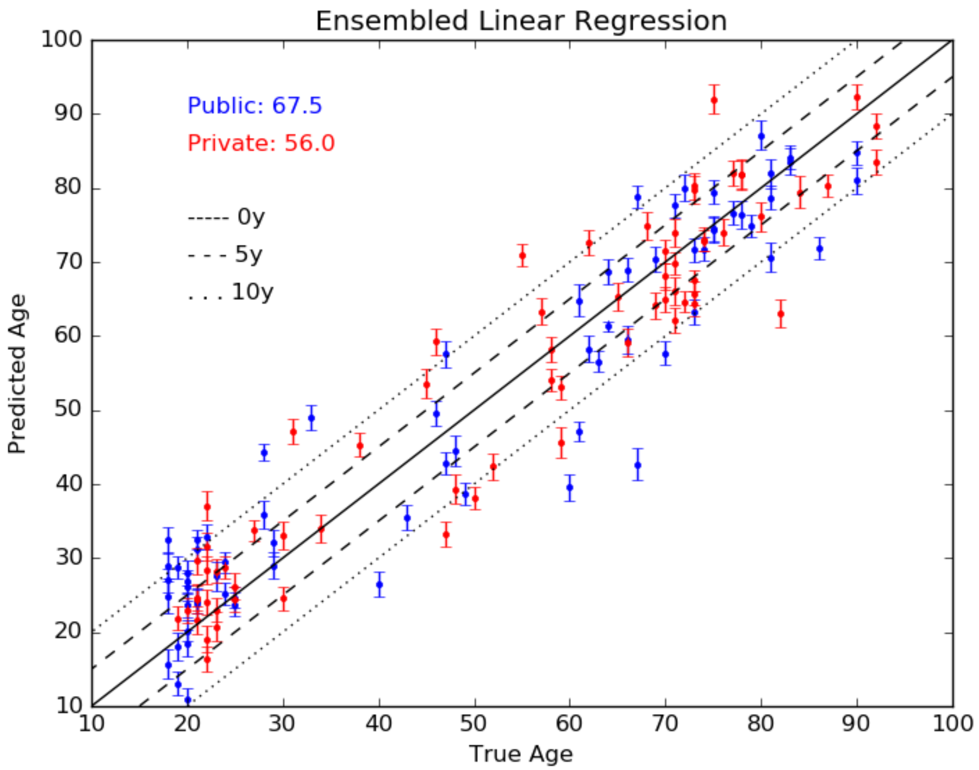
?

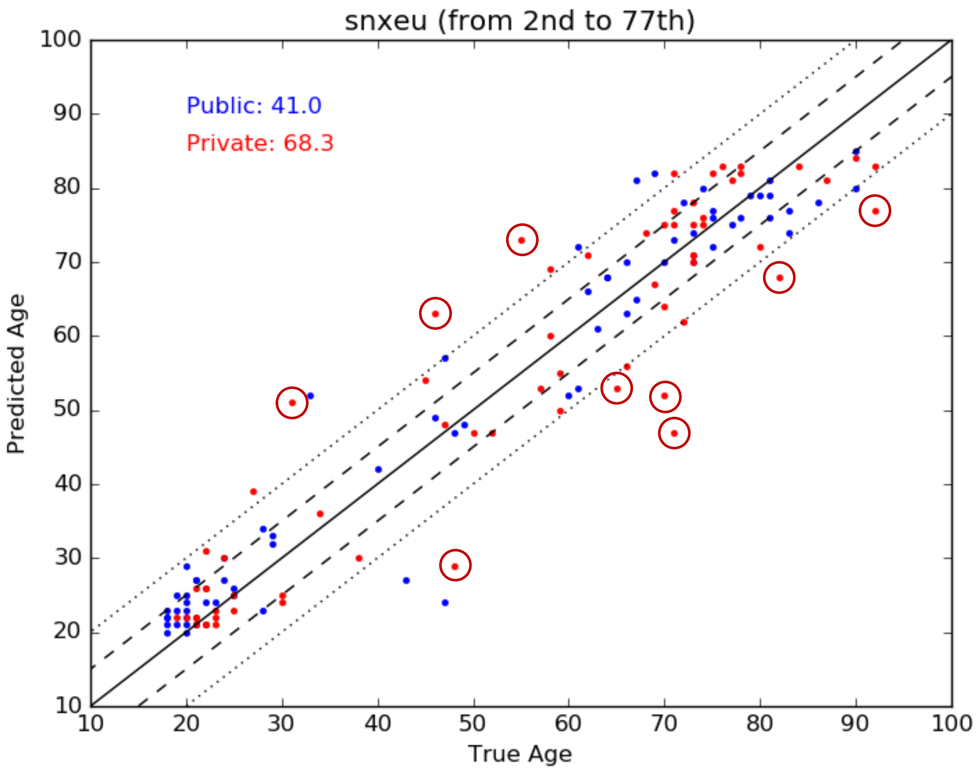
A **and** B **and** C

!

$$y = \omega_A y_A + \omega_B y_B + \omega_C y_C$$

Avoid Overfitting





Cross Validation!

Lessons		
preprocessing	filtering helps	
	brightness & contrast helps	
	centering not needed	
feature selection	selection better than rand	
	pca too crude	
	lasso too selective	

Lin. Regression		masked	equalized	centered
no filter	selected	74.5	69.0	69.0
	random	82.4	79.0	79.0
	pca			**91.3**
filter	selected	68.1	**62.6**	62.6
	random	79.9	75.7	75.7
	pca			88.0

Ridge		masked	equalized	centered
no filter	selected	74.4 (0.01)	69.0 (0.01)	69.0 (0.01)
	random	82.8 (0.01)	79.1 (0.01)	79.1 (0.01)
	pca			**87.1** (50k)
filter	selected	68.1 (0.01)	**62.6** (0.01)	62.6 (0.01)
	random	79.6 (0.01)	75.1 (0.01)	75.1 (0.01)
	pca			83.4 (50k)

Lasso		masked	equalized	centered
no filter	selected	109.8 (100)	105 (0.1)	105.0 (0.1)
	random	98.7 (250)	96 (0.1)	96.0 (0.1)
	pca			**83.8** (30)
filter	selected	95.4 (250)	98.8 (0.1)	98.8 (0.1)
	random	**102.5** (250)	98.8 (1)	98.8 (1)
	pca			87.4 (30)

filter: 3x3x3 mean filter

mask: remove all zero voxels

equalize: image mean = 0, standard deviation = 1

center: subtract sample mean from each image

selected: top 100k voxels by mutual information

random: 100k randomly selected voxels

pca: 400 components