SPEECH RECOGNITION CHALLENGE

Kiran Bacsa
Manuel Vonlanthen
Adrian Löwenstein

January 25, 2018

Content

- Objective
- Feature Extraction
- Classification Pipeline
- Spectral Clustering
- Semi-Supervised Classification
- Conclusion

OBJECTIVE

Objective

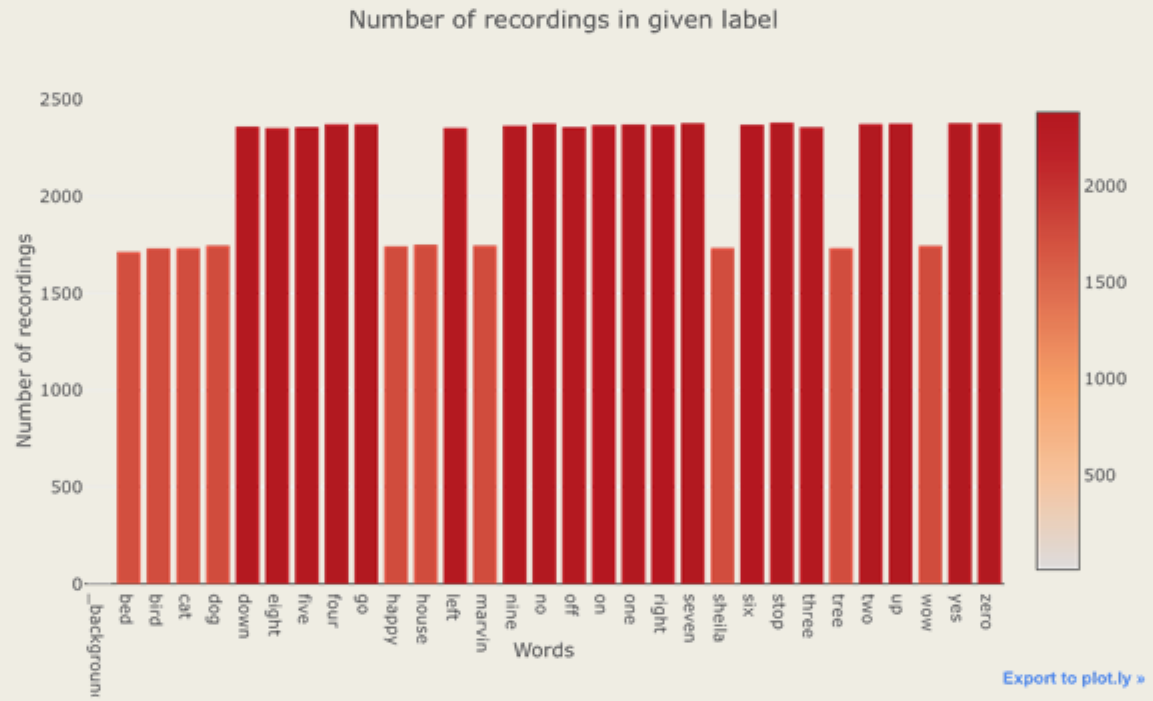
- Kaggle Competition : “TensorFlow Speech Recognition Challenge”
- Design a speech recognition algorithm that is able to recognize simple speech commands.
- Our Approach : Use graph methods of NTDS to classify the commands

Commands to be recognized	"Yes", "No", "Up", "Down", "Left", "Right", "On", "Off", "Stop", "Go", "Zero", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine"
Auxiliary words	"Bed", "Bird", "Cat", "Dog", "Happy", "House", "Marvin", "Sheila", "Tree", and "Wow"

FEATURE EXTRACTION

Description of the Dataset

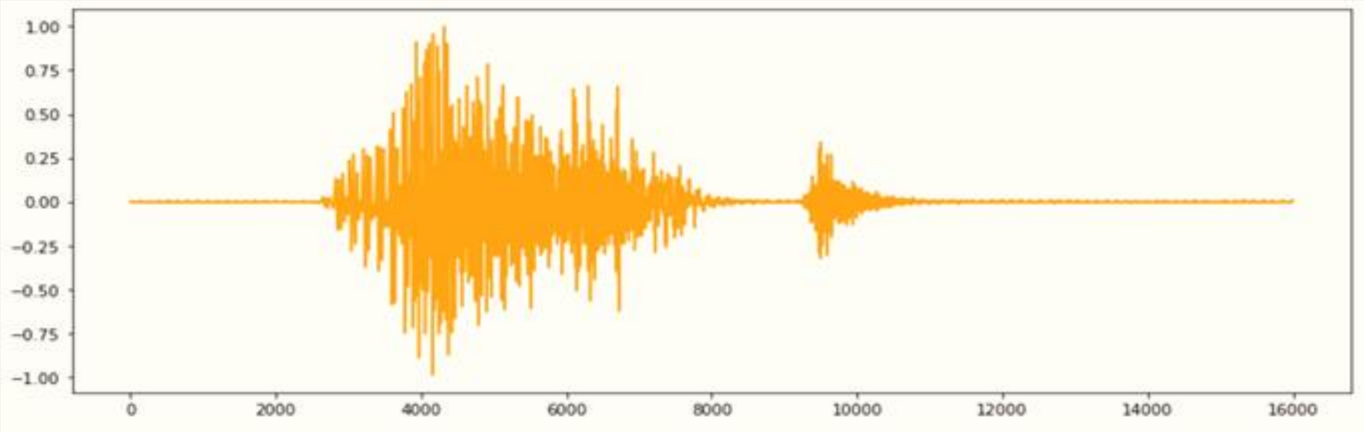
- Used only the Test set from the Kaggle competition.
- 64’720 audio files
- 1 second audio files with different speakers
- 30 Different words
- Only 20 need to be classified
- Data is not perfect :
  - *Empty Audio Files*
  - *Noisy Audio Files*



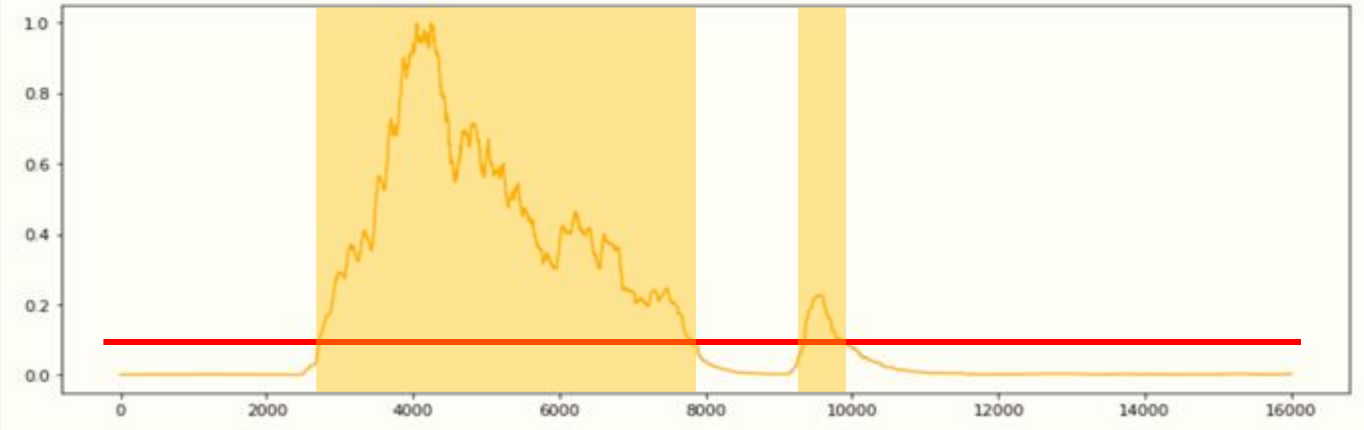
Smart Cutting the Audio Files

- Audio Files are only 1s long, but the word pronunciation itself takes around 40% of the duration of sample only.
- Computing Directly the Features over the full time => Not precise features
- Idea :
  - *Cutting the silence and noisy parts of the signal.*
  - *Keeping only the word itself*
  - *Get better features*

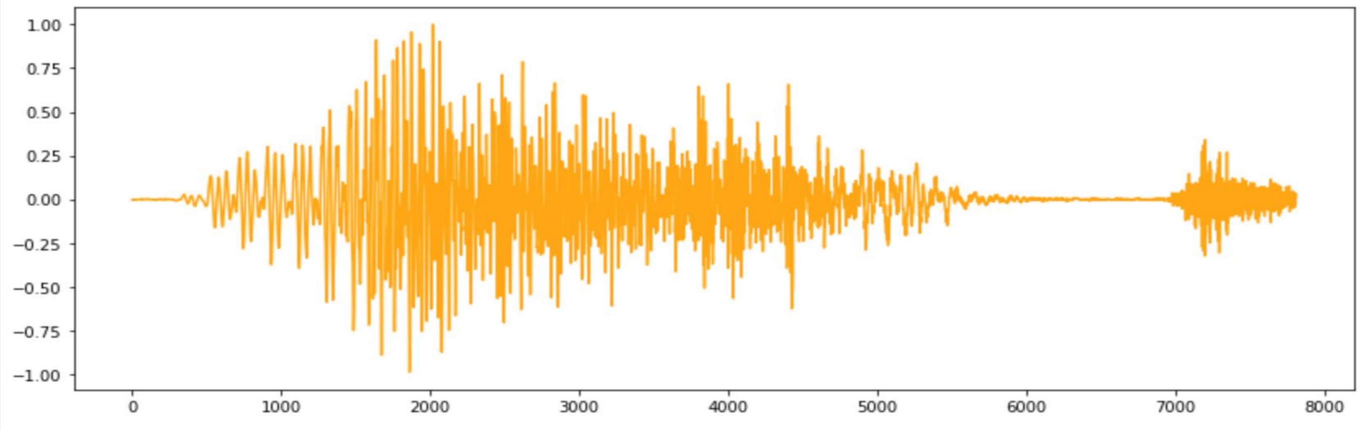
- Raw Audio Sample



- RMSE of the Sample
  - *Threshold*
  - *Lobe Selection*

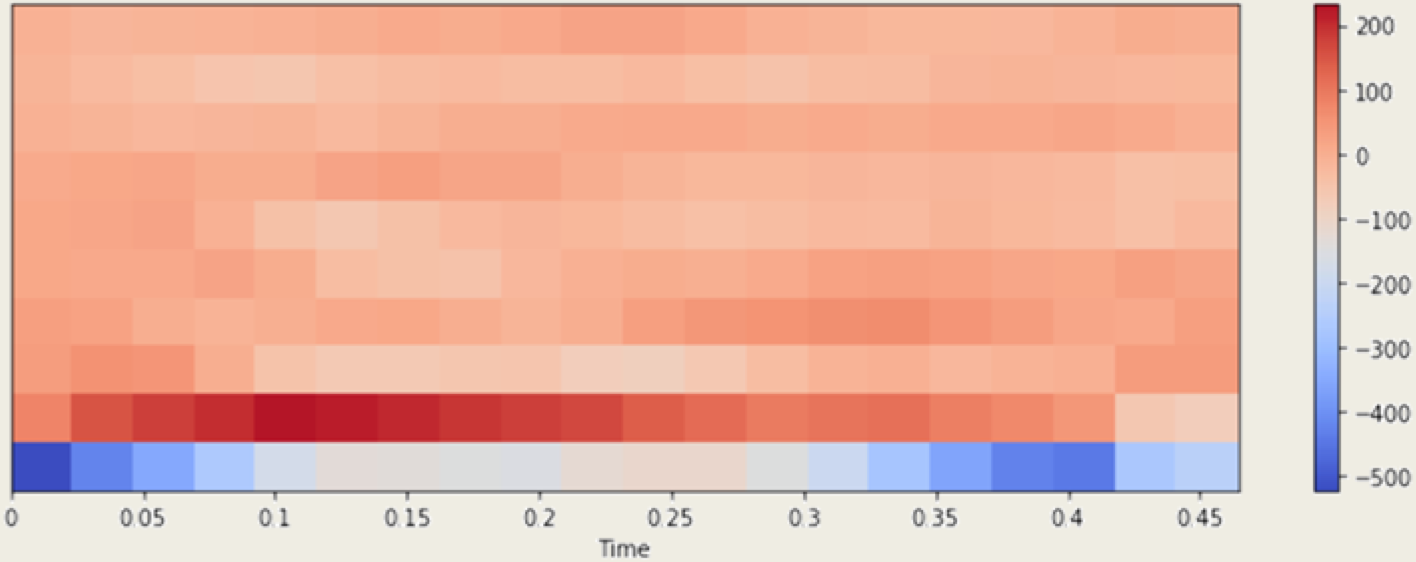


- Final Audio Sample



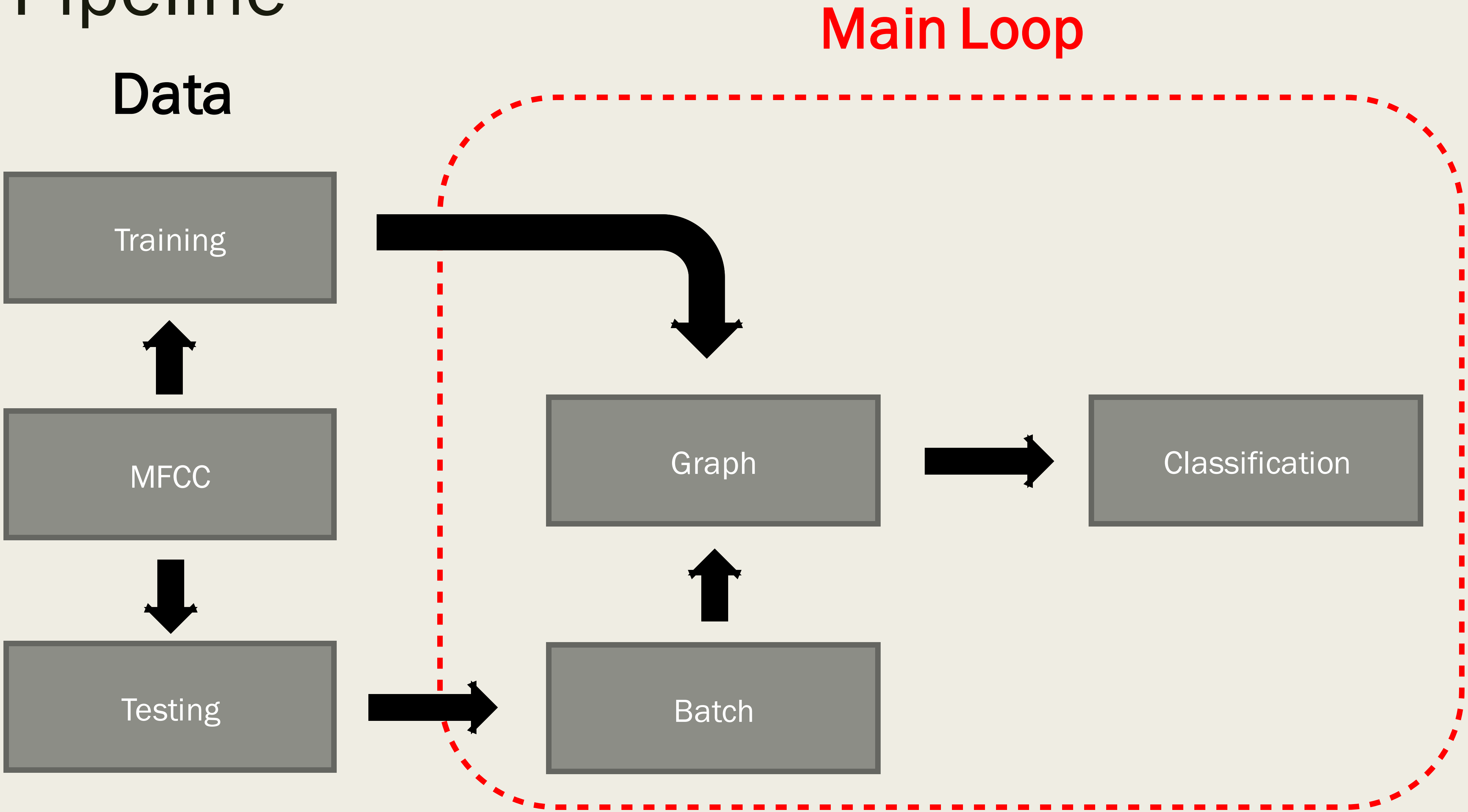
Choice of the Features : MFCC

- Mel-frequency cepstral coefficients (MFCC) are a classical but robust way of creating features from the audio.
- The dimension of MFCCs is set to 10 : this is enough as we are analysing speech
- We adapt the STFT dynamically, to get a the same number of MFCCs : 20



CLASSIFICATION PIPELINE

Pipeline


Main Loop
Data
Training
MFCC
Testing
Graph
Batch
Classification

Splitting training and testing data

Problem :

Both training and testing need the same proportion of each class in order to avoid biases during training.

- Sample each class separately for training and test set

- Split test set into batches

- Build similarity graph based on training and current batch

Using graph theory for feature extraction

- Compute cosine distance between each point

$$d(\mathbf{x_i},\ \mathbf{x_j}) = \frac{\mathbf{x_i}^T \mathbf{x_j}}{\|\mathbf{x_i}\|_2 \|\mathbf{x_j}\|_2}$$

- Build similarity graph with RBF kernel

$$\mathbf{W_{i,\ j}} = exp\left(\frac{-d(\mathbf{x_i},\ \mathbf{x_j})^2}{\sigma^2}\right)$$

- Extract normalized Laplacian

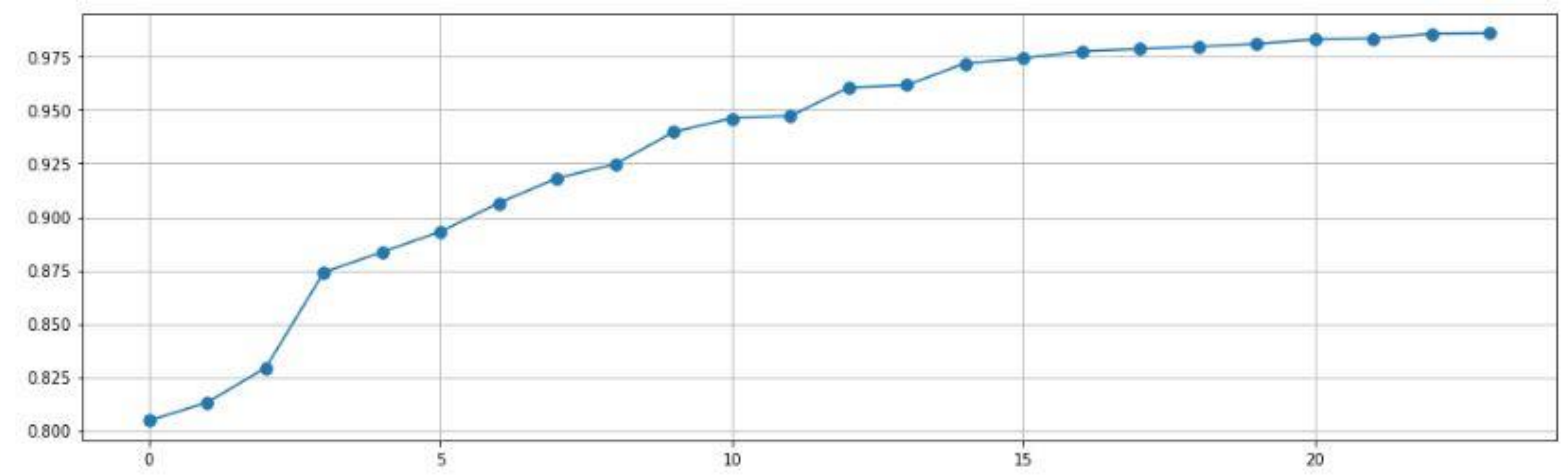
$$\mathbf{L} = \mathbf{D} - \mathbf{W}$$

$$\mathbf{L}_{\text{norm}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$$

SPECTRAL CLUSTERING

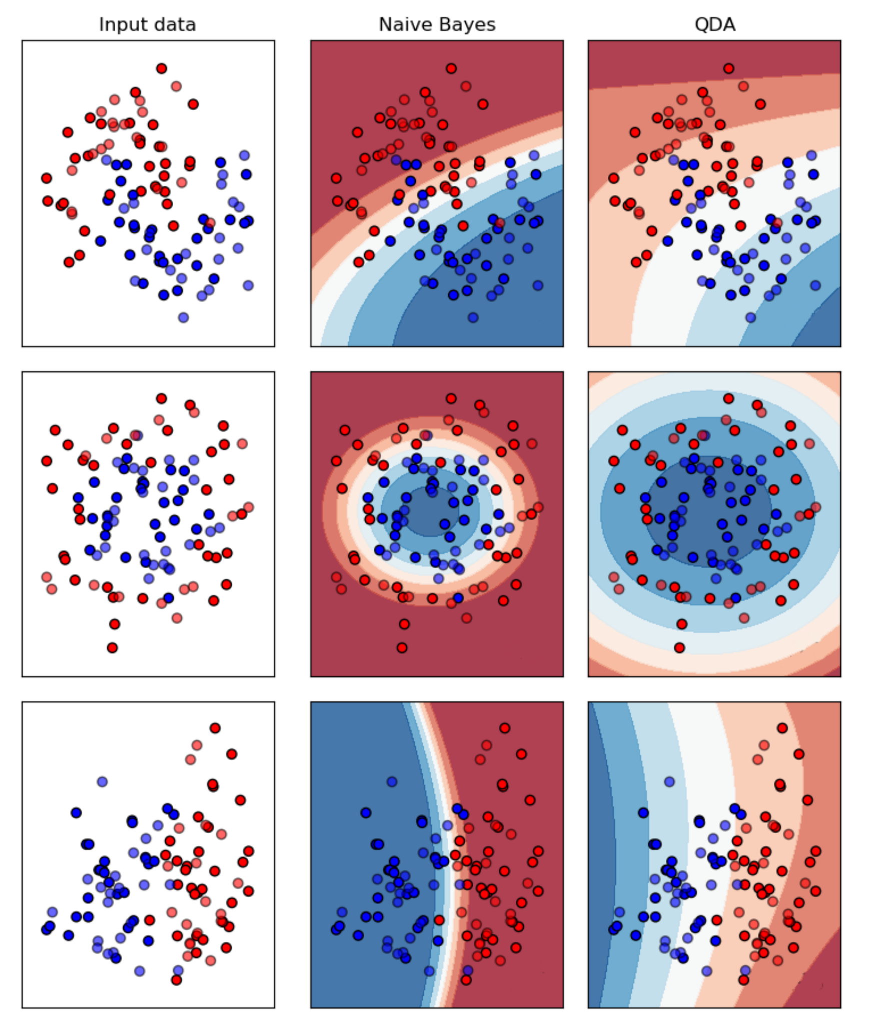
Use eigenvectors as features

Eigenvalues of normalized laplacian



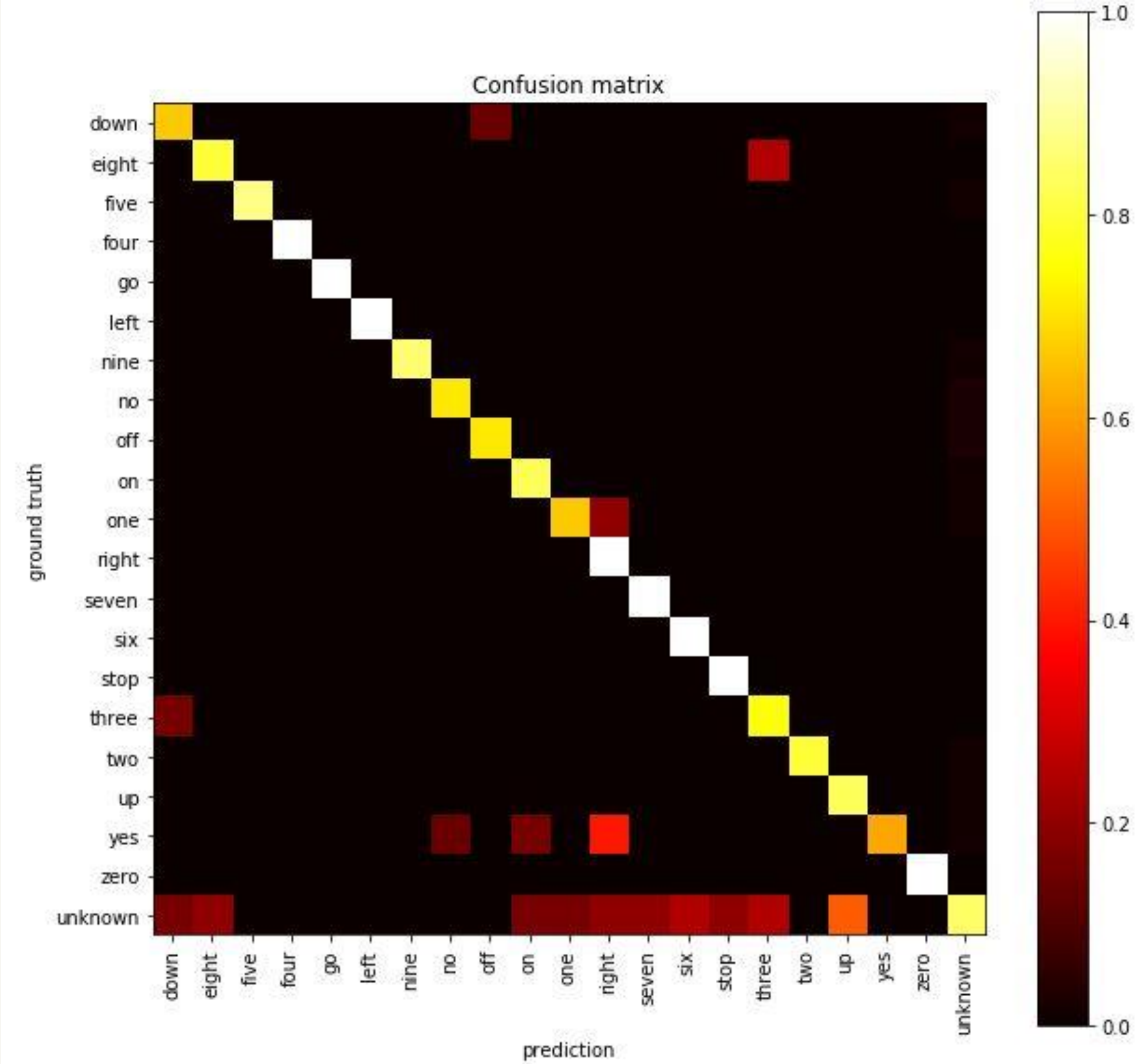
Classification

- Gaussian Naive Bayes : 67%
- Quadratic Discriminant Analysis : 78 %



http://scikit-learn.org

Confusion matrix



SEMI-SUPERVISED CLASSIFICATION

Semi-Supervised Classification

- Build graph using the 5000 data points (audio feature vectors)
  - N = 4800 training data points
  - K = 200 validation data points
  - Similarity graph using cosine distance
- Sparsify weight matrix **W** using k-Nearest Neighbours (k = 120)
- Define class indicator vectors **yi** as signals on the graph:
  - K nodes still unlabelled

$$\mathbf{y}_{\mathrm{i}} = \{0, 1\}^{N+K} \qquad y_{i,j} = \begin{cases} 1 & \text{if node j is in class i} \\ 0 & \text{otherwise} \end{cases}$$

Semi-Supervised Classification

- Estimate unknown node labels for each of the 30 classes **i**, by solving:

$$\underset{\hat{\mathbf{y}}_{\mathbf{i}} \in \mathbb{R}^{(N+K)}}{argmin} \quad \frac{1}{2}\|\mathbf{M}_{\mathbf{i}}(\mathbf{y}_{\mathbf{i}} - \hat{\mathbf{y}}_{\mathbf{i}})\|_2^2 + \frac{\alpha}{2}\hat{\mathbf{y}}_{\mathbf{i}}^{T}\mathbf{L}\hat{\mathbf{y}}_{\mathbf{i}} + \frac{\beta}{2}\|\hat{\mathbf{y}}_{\mathbf{i}}\|_2^2$$

- **Term 1:** Fidelity Term
  - Known labels should stay the same
- **Term 2:** Estimated signal should be smooth on graph
- **Term 3:** Tikhonov regularization
  - keeps signal energy low
  - Keeps problem from being ill-posed

Semi-Supervised Classification

- Optimization solved explicitly:

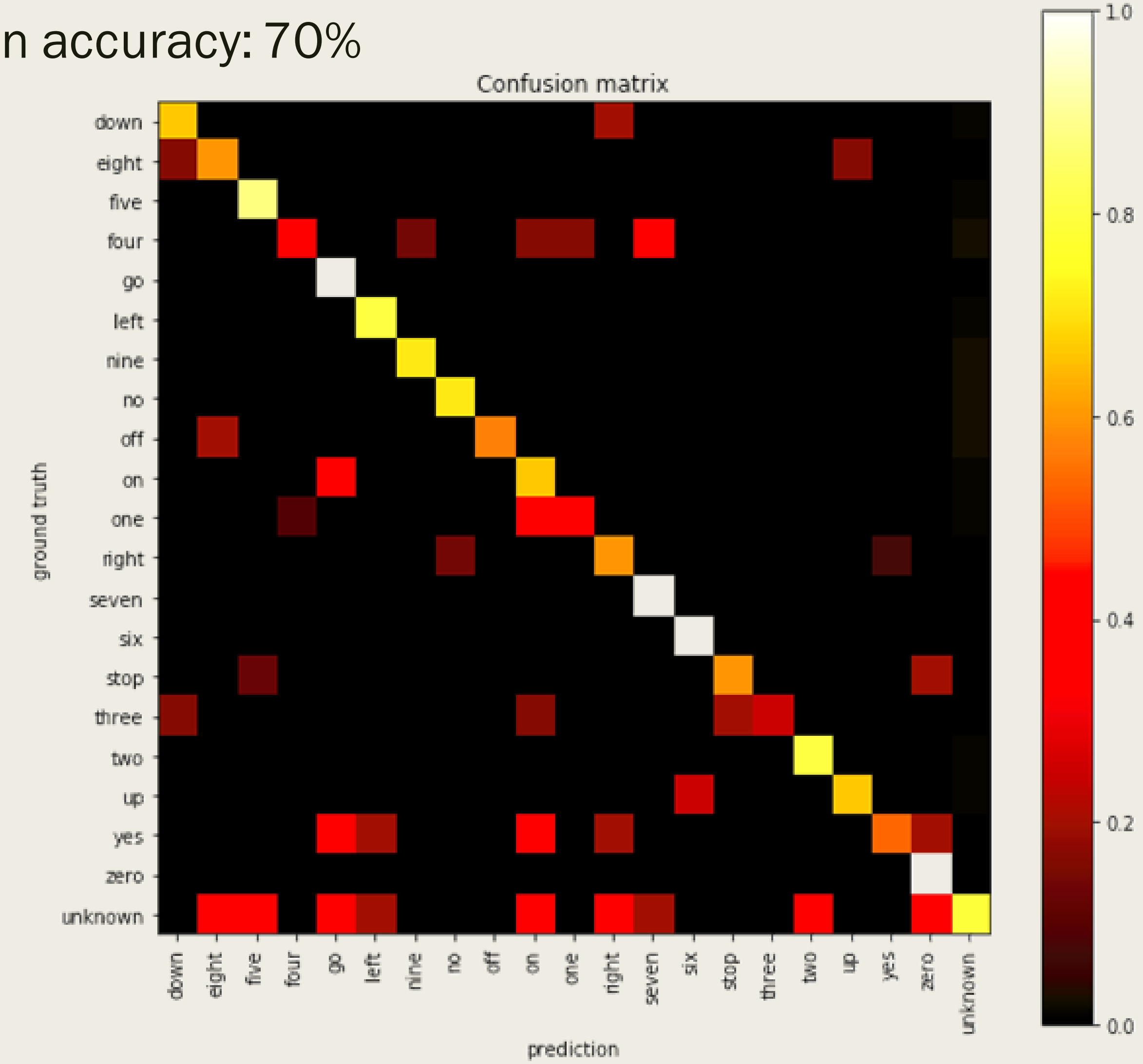
$$\mathbf{y_i} = (\mathbf{M_i} + \alpha\mathbf{L} + \beta\mathbf{I}_{(N+K)(N+K)})\hat{\mathbf{y}}_{\mathbf{i}}^{*}$$

- Labels of unknown nodes are determined with:

$$i_j^* = \underset{i\in\{1,2,\ldots,30\}}{argmax} \quad \hat{\mathbf{y}}_{\mathbf{i,j}}$$

Semi-Supervised Classification

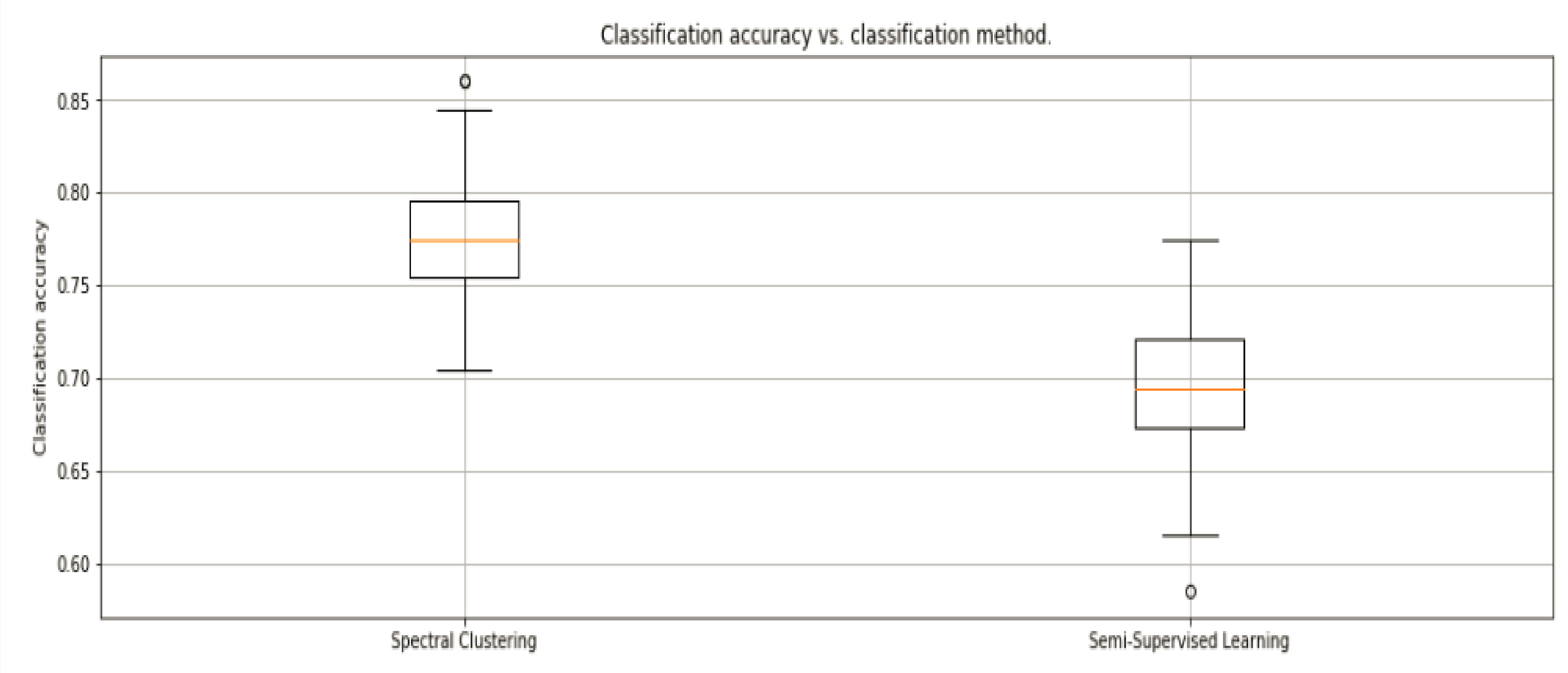
- Achieved mean accuracy: 70%



CONCLUSION

Conclusion

- Achieved accuracies for the two classification methods:



Conclusion

- Spectral clustering (78%) achieved better results than Semi-Supervised Classification (70%)
  - SSC has potential to be faster than SC because it uses sparse matrices
  - SSC could probably be improved by tuning hyperparameters further
- Best results on kaggle competition achieve 90% accuracy
  - 78% is in the top 50% of competition
- Most effective steps during project
  - Extracting words from audio files
  - Going from statistical mfcc's to raw, dynamic mfcc's
  - Sparsify graph for SSC, don't sparsify for SC
- Further improvements
  - More extensive parameter tuning (especially for SSC)
  - Analyse/Include other methods (e.g. Graph Inference)

Questions