Metis Project 3 - Classifying Whether an Audio Sample Contains a Human voice

Gregory LuI

Project Design

When I started this project I was originally unsure of what to do, for the first couple of days I just browsed aimlessly on Kaggle looking for something I thought would be interesting. I had an idea that I wanted to analyze audio samples, which I thought would be too difficult for me at this point in the bootcamp. However, following Cliff's suggestions and some of my classmates examples I decided to try an and analyze human speech.

I used a dataset I found online while browsing for 'human voice datasets'. The dataset from CHiME [1] contained audio samples that were pre sliced into 4 second clips which I thought would be more convenient to work with than arbitrary length audio clips. The data contained only information on which audio clips had human speech so I extracted features on my own via the Librosa library [2]. The features were Mel Frequency Cepstral Coefficients [3], which in a nutshell relate to frequencies that a human being would hear. For a 4 second audio clip the MFCC defaults gave me 4000+ features, since my dataset only had 2000 observations I needed to reduce my dimensions while retaining as much information as possible. Robert and Cliff both suggested PCA, which we hadn't learned yet at the time, but there were ample easy-to-understand resources online [4].

When I started modeling for this project the MVP was using logistic regression, specifically LogisticRegressionCV. It wasn't until later that I found out this implementation had a major flaw which wouldn't scale my data correctly when it was split into different folds (there may have been an option for it, but I did not find it at the time). This prompted me to try and switch into using pipelines which were very powerful. It so happens that Flora did an investigation around the same time on automating machine learning, which really inspired me to try and automate my process as well, i.e. the kitchen sink approach.

I looked up online how to use GridSearchCV combined with Pipelines [5] and was able to create a pipeline I am really proud of. To insert a new classifier into the pipeline would only take about 8 lines of code, and if I had more time I would've added more models than the 4 I used: KNN, logistic regression, SVM.SVC, randomforest. Looking up information on how/what to tune for the parameters took a long time, and I think because I was tuning the variance-explained for the PCA each pipeline fitting/transforming took a long time. This was something I should've memoized.

In the end the results showed that KNN was my "best" model using an F1 score to compare across models. Surprisingly the PCA parameter only needed 20% of the variance-explained, which was just about 5 components (down from 2000) [6]. I took a look at some of the other models which used closer to the 40-50% variance-explained, so about 200-500 components.

Tools

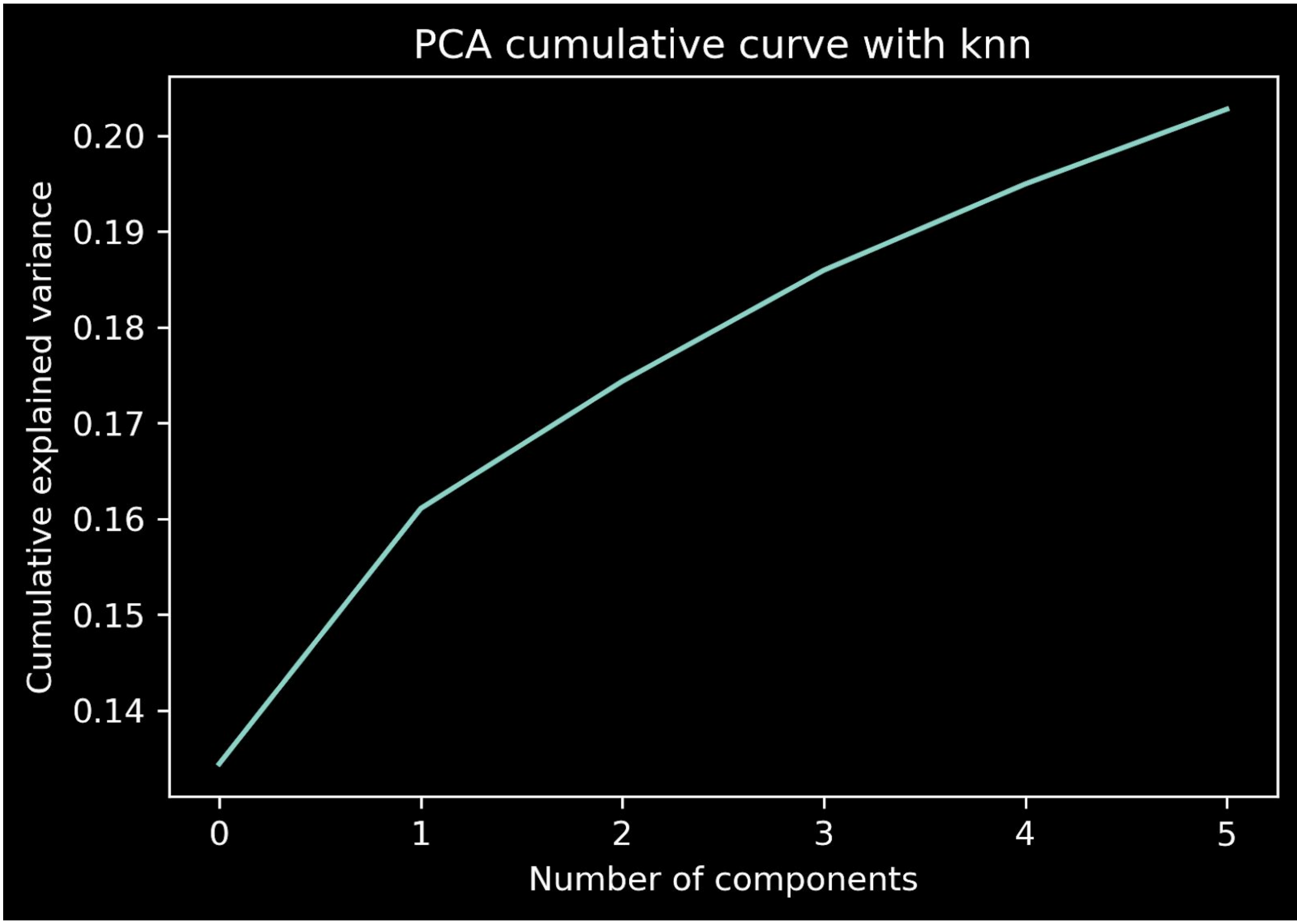
- Python
    - Pandas, scikit-learn, jupyter notebook, vscode with python interpreter
- Google slides, Google docs

Conclusion

The F1 score for a naive model that just guesses was ~80%, and my model was at ~87%, so there wasn't much of an improvement. One of the issues was that my dataset was slightly imbalanced, with 65% of it having human voices and 35% not. From the class examples I thought that because it wasn't an extreme imbalance, e.g. 90:10 I didn't have much to worry about, but I should've spent some time adjusting the balance via sampling. The overall conclusion is that machine learning could definitely help humans to identify speech in audio samples, and next time I would love to use a larger and more diverse dataset to model with.

Appendix

1. CHiME data homepage http://spandh.dcs.shef.ac.uk/chime_challenge/CHiME5/
2. Librosa library https://librosa.github.io/librosa/
3. MFCC explanation https://medium.com/@jonathan_hui/speech-recognition-feature-extraction-mfcc-plp-5455f5a69dd9
4. https://www.youtube.com/watch?v=FgakZw6K1QQ
5. GridsearchCV and pipelines https://towardsdatascience.com/a-simple-example-of-pipeline-in-machine-learning-with-scikit-learn-e726ffbb6976

6. 

PCA cumulative curve using KNN

7.