Speech Accent Analysis

By Dastan Abdulla

Table of Content

- Goals and Motivation
- The Data
  - Initial Kaggle Data(sub)set
  - Full Dataset (thanks to Dr. Weinberger)
- Phonetic Parsing (Panphon)
- Audio Feature Extraction (Librosa)
- Language Prediction
  - Panphon features, Vectorized Phonemes, and Audio Features
- Country Prediction
  - Simple Approach & Custom Geographic Model.

Goals and Motivation

Research Question: How can variations in phonetic, lexical, and syntactic features in second language English usage be used to predict a speaker's native language and country of origin?

Motivation:

- Understanding Language Variation and Change for Sociolinguistics
- Language Policy and Education for English
- Social Integration and Communication

The Data

Initial Kaggle Dataset 1

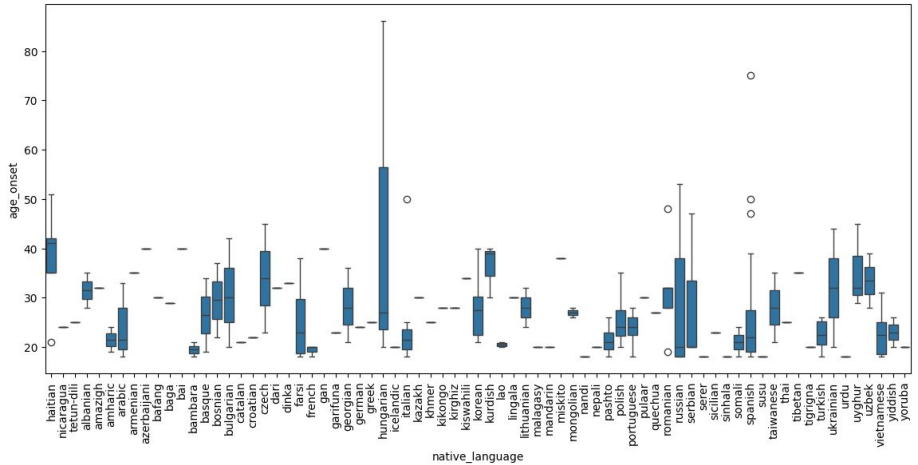
- Drop extraneous columns, sort, and clean entries
- Identify and download missing audio files
- Examine the age of the speakers, and the age they learned english (age_onset)

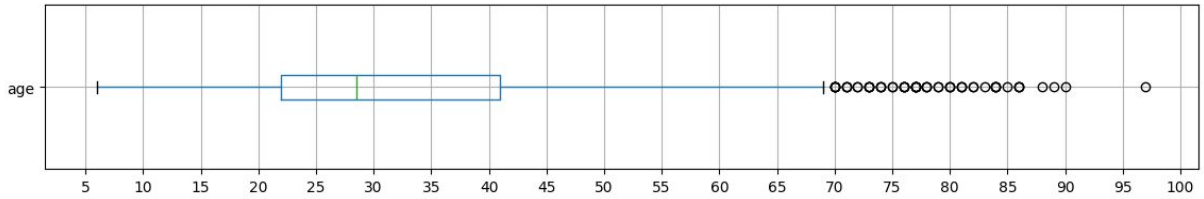
	age	age_onset	birthplace	filename	native_language	sex	speakerid	country	file_missing?	Unnamed: 9	Unnamed: 10	Unnamed: 11
0	24.0	12.0	koussi, senegal	balanta	balanta	male	788	senegal	True	NaN	NaN	NaN
1	18.0	10.0	buea, cameroon	cameroon	cameroon	male	1953	cameroon	True	NaN	NaN	NaN
2	48.0	8.0	hong, adamawa, nigeria	fulfulde	fulfulde	male	1037	nigeria	True	NaN	NaN	NaN

Initial Kaggle Dataset Visuals

Please call Stella. Ask her to bring these things with her from the store: Six spoons of fresh snow peas, five thick slabs of blue cheese, and maybe a snack for her brother Bob. We also need a small plastic snake and a big toy frog for the kids. She can scoop these things into three red bags, and we will go meet her Wednesday at the train station.

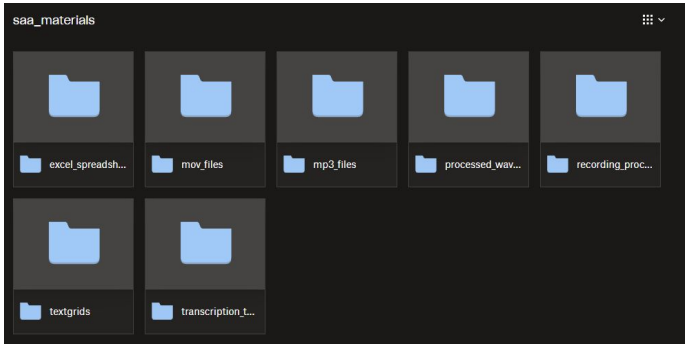
	age	age_onset	birthplace	filename	native_language	sex	speakerid	country
0	24.0	12.0	koussi, senegal	balanta	balanta	male	788	senegal
1	18.0	10.0	buea, cameroon	cameroon	cameroon	male	1953	cameroon
2	48.0	8.0	hong, adamawa, nigeria	fulfulde	fulfulde	male	1037	nigeria
3	42.0	42.0	port-au-prince, haiti	haitian	haitian	male	1165	haiti
4	40.0	35.0	port-au-prince, haiti	haitian1	haitian	male	1166	haiti





Full Dataset (thanks to Dr. Weinberger)

- The Kaggle dataset was lacking
  - No transcriptions, No learning style
- I decided to reach out to the maintainers, Dr. Weinberger.
- Eventually he responded and gave me access to a HUGE dropbox for the dataset.



Processing the Full Dataset

- There was a huge excel sheet that maintained the speaker info's
- First I made sure the audio and transcription files were actually there
  - They were for the most part, but I had to do a TON of corrections
- Besides typos and mistakes, formatting for everything was all over the place
- I decided to use ISO standard conventions for normalization
  - Language names and country names specifically were pretty bad…
  - Used python library pycountry (not really a library)

```
directory = "../data/transcriptions_text/"

for transcription in saa_df[saa_df['phonetic_tr
    file_path = os.path.join("../data/transcrip
    if not os.path.exists(file_path):
        print(f"The file '{transcription}' does
```

```
The file 'czech5.txt' does not exist.
The file 'not' does not exist.
The file 'mandarin42.txt' does not exist.
The file 'arabic195.txt' does not exist.
The file 'portuguese68.txt' does not exist.
```

```
directory = "../data/processed_wav_files/"

for file in saa_df['speech_sample']:
    file_path = os.path.join(directory, file)
    if not os.path.exists(file_path):
        print(f"The file '{file}' does not exist.")
```

```
The file 'estonian15.wav' does not exist.
The file 'wu4.wav' does not exist.
The file 'arabic195.wav' does not exist.
The file 'french81.wav' does not exist.
The file 'catalan6.wav' does not exist.
The file 'thai22.wav' does not exist.
The file 'spanish230.wav' does not exist.
The file 'arabic196.wav' does not exist.
```

Even more processing...

- I had to convert the rtf transcription files to txt files
  - Initially used a python library but that didn't work
  - So I ended up using pandoc as per the instructors recommendation
- After it was all said and done, my data looked like this

speakerid	native_language	country	age	gender	onset_age	english_residence	length_of_residence	learning_style	speech_sample	phonetic_transcription	ethnologue_language_code	transcription	language_name
174	farsi	iran	29.0	male	11.0	usa	2.0	academic	farsi7.wav	farsi7.txt	pes	\[pliz kɔl ə̈stɛlə æsk hˈɛɹ tu bɹĩŋ ḍiə̈s ṯ...	Iranian Persian
522	english	usa	22.0	male	0.0	usa	22.0	naturalistic	english146.wav	english146.txt	eng	\[pʰliz kʰɑlʸ stɪlə æsk hə˞ rə bɹĩŋ n̥iz θĩ...	English
2923	english	usa	18.0	female	0.0	usa	18.0	naturalistic	english641.wav	None	eng	None	English
271	nepali	nepal	22.0	female	5.0	usa	4.0	academic	nepali1.wav	nepali1.txt	npi	\[plis kʰɑl ʂʂlʌ æks hə˞ tu brĩŋ ḍiz θɪŋks ...	Nepali (individual language)
485	english	australia	28.0	male	0.0	australia	28.0	naturalistic	english125.wav	english125.txt	eng	\[pʰliz kʰɔlʸ stɛla ask ə˞ rə bɹĩŋ ðɪz θĩŋz	English

Glottolog

- I cross referenced the languages Glottolog 5.0 to extract continent information
  - And later on latitude and magnitude information for each country
- This is an extremely powerful source, and you could even trace down dialects and language families with it (which I would've liked to do, but ended up being extremely convoluted)

	glottocode	name	isocodes	level	macroarea	latitude	longitude
18615	tala1285	Talaud	tld	language	Papunesia	4.11846	126.795
12844	natc1249	Natchez	ncz	language	North America	31.75000	-91.330
4045	dats1234	Datshang	NaN	dialect	Eurasia	NaN	NaN
20269	vall1258	Valluno Spanish	NaN	dialect	Eurasia	NaN	NaN
16174	repa1237	Repanbitip	rpn	dialect	Papunesia	-16.32650	167.619

Phonetic Parsing

Panphon

- The ‘transcription’ column needed a ton of clean up as usual
  - Different file formats, word unicode characters, token spacing etc
- After the cleanup, I used Panphon to parse the actual IPA tokens
- Basically for each token encountered,
- Panphon looks up it’s features in this table

1	ipa	syl	son	cons	cont	delrel	lat	nas	strid	voi	sg	cg	ant	cor	distr	lab	hi	lo	back	round	velaric	tense	long	hitone	hireg
2	ɺ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-
3	ɻ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	-
4	ɨ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	ɭ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	+
6	ɭ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	+
7	ʃʶ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	+	0	0	0	0	-	-
8	ʃʶ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	+	0	0	0	0	+	-
9	ʃʶ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	+	0	0	0	0	0	0

```python
import panphon as panphon
ft = panphon.FeatureTable()
print('segment: ', ft.segs(test)[2])
print('features: ')
ft.word_fts(test)[2]
for sign, feature in ft.word_fts(test)[2]:
    print(f"Feature: {feature}, Sign: {sign}")
```

```
segment:  i
features:
Feature: syl, Sign: +
Feature: cg, Sign: -
Feature: sg, Sign: -
Feature: cor, Sign: -
Feature: long, Sign: -
Feature: lab, Sign: -
Feature: voi, Sign: +
Feature: cont, Sign: +
Feature: ant, Sign: 0
Feature: tense, Sign: +
Feature: nas, Sign: -
Feature: back, Sign: -
Feature: distr, Sign: 0
Feature: son, Sign: +
Feature: velaric, Sign: -
Feature: lat, Sign: -
Feature: lo, Sign: -
Feature: cons, Sign: -
Feature: round, Sign: -
Feature: hi, Sign: +
Feature: strid, Sign: 0
Feature: delrel, Sign: -
```

Creating the distribution

- For each sample, I basically collected the number of total present features, and made a column for each
- Probably not the most rigorous way to do it, as there is more room for phonetic analysis but still.

speech_sample	...	long	nasal	round	sonorant	syllabic	velaric	voice	distributed	strident	tense
swedish10.wav	...	1	35	40	136	86	0	161	15	10	35
sundanese1.wav	...	6	27	22	142	87	0	164	9	7	47
english147.wav	...	10	34	23	128	79	0	161	15	9	40
tatar1.wav	...	2	22	36	131	82	0	157	18	11	41
korean12.wav	...	4	26	30	138	89	0	166	15	10	50

Audio Processing

Extracting Audio Features

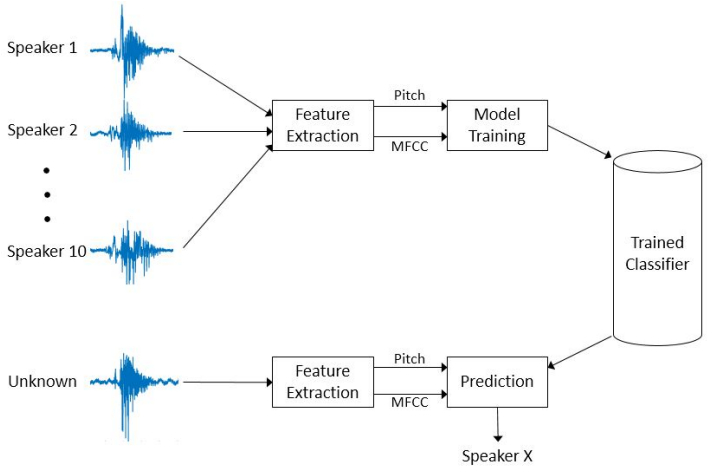
- Advantage: remove any human error or thoroughness factors during transcriptions
- I used a library called librosa to extract the mfcc and pitch features for each sample

```
Processing audio files: 100%|██████████| 3016/3016 [09:14<00:00,  5.43it/s]
```

```
features_df = pd.DataFrame(features_list, columns=['features', 'file'])
features_df.to_pickle('../data/audio_features.pkl')
```

```
features_df.head()
```

	features	file
0	[-320.2046203613281, 113.08561706542969, 13.68...	afrikaans1.wav
1	[-293.78033447265625, 124.66605377197266, -6.4...	afrikaans2.wav
2	[-338.8608703613281, 132.85476684570312, 6.337...	afrikaans3.wav
3	[-285.78668212890625, 136.85784912109375, 8.95...	afrikaans4.wav
4	[-371.3587646484375, 134.38897705078125, 3.720...	afrikaans5.wav



Language Prediction

Note: I did perform significance testing for the features and labels before making these.

The good the bad and the ugly

- Before I did anything I tested for significance
- I looked at three different models
  - One using the Panphon derived features (22 numeric columns)
  - One using tfidf vectorization (the transcriptions directly)
  - One using the audio features (to remove any human error during transcriptions)
- Overall they all did equally horribly (26% accuracy at best), but here's the thing….
  - Our baseline ‘guessing’ accuracies were about 0.08%
  - Our data was extremely imbalanced and limited (only 1269 entries for the non audio and english making up about a fourth of those)

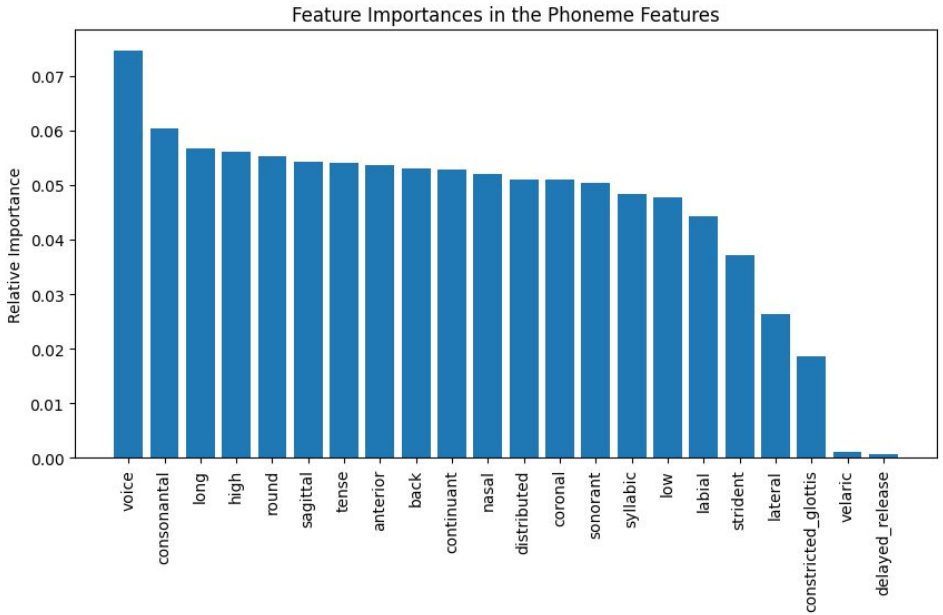
```
Feature:  continuant            Significant: p =  7.543576669811675e-15
Feature:  coronal               Significant: p =  1.0857911765596683e-25
Feature:  delayed_release
Kruskal-Wallis Test stat=153.15821381473737, p=0.593901970325533
ANOVA:
                df     sum_sq   mean_sq         F    PR(>F)
C(country)   158.0   1.181354  0.007477  0.706041  0.996922
Residual    1110.0  11.754816  0.010590       NaN       NaN
Feature:  high          Significant: p =  5.441580079261381e-19
Feature:  lateral               Significant: p =  9.3839392155854e-13
```

The models

- For the Panphon features: `RandomForestClassifier(n_estimators=100, random_state=42)`
- For the tfidf Vectorizer: `MultinomialNB()`
  - `TfidfVectorizer(max_df=0.5, max_features=1500 , tokenizer=phoneme_tokenizer, lowercase=False, token_pattern=None, ngram_range=(1, 3))`
    (this is looking at trigrams)
- For the audio features: `RandomForestClassifier(n_estimators=100, random_state=42)`

Most informative features (Panphon)

- Voice came out to be one of the most informative features
  - Suggests that vocal cord vibration was informative for distinguishing the language(s)
- The velaric and delayed release were the least informative.



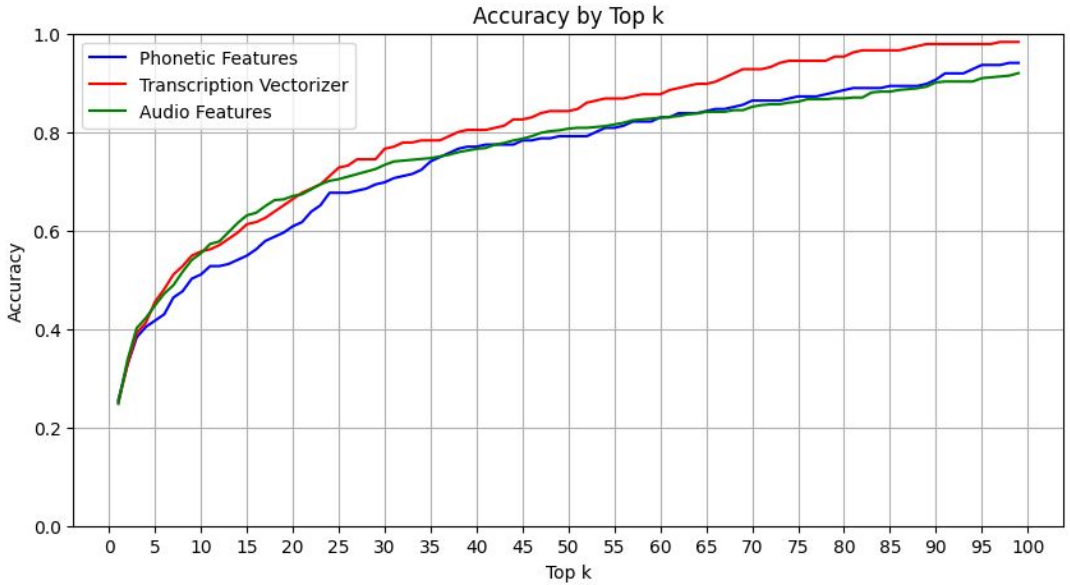
Most informative features (Tfidf trigram)

- Note that this only covers some of the labels because there were over 100+ possible labels for each language represented in the data.
- I didn't look at informative features for the audio as they are mostly numeric heavily math operated on features.

```
Top features for class 0:  ['h ɜ', 'ˈ', 'ɛ ˈ', 'ˈ ̃n', 'ɛ ɪ', 'z θ', 'i ŋ', 'ə ʊ', 'ə n', 'ŋ z ˌ']
Top features for class 1:  ['ɛ ː', 'ɛ n', 'h ɛ', 'l ɣ', 'ɣ', 't ʊ', 'ː s', 'i ː s', 'ˈˈ', 'ɔ l ɣ']
Top features for class 2:  ['r', 's t ˌ', 'ˌ ɪ', 'ɪ s', 't ˌ ɪ', 'ɜ', 'h ɜ', 'ˌ ɪ ˈ', 'ɪ s t', 'ɔ f']
Top features for class 3:  ['ə ɹ', 'h ə ɹ', 'ɜ', 'h ɜ', 'r', 'h ə', 'ə ɹ b', 'f ɔ ɹ', 'ɹ b', 'ŋ z ˌ']
Top features for class 4:  ['s ˌ', 'r', 'ʔ', 'e s', 'i k s', 'ɜ ɹ', 'z t', 'k i d', 'i d ˌ', 'k i']
Top features for class 5:  ['ˌ', 'æ ˌ', 'a ˜n', 'a ˈ', 'z ˌ s', 'ɪ ˈ', 'ɔ v', 'z ˌ ɔ', 'k a', 'ˌ ɔ']
Top features for class 6:  ['r', 'x', 'x ɜ', 'ɛ s t', 'ˈ', 'ɛ s', 's t ˌ', 's ɔ', 'r b', 'x ɛ']
Top features for class 7:  ['r', 'h ɛ r', 'ɛ r', 'd ˌ e', 't u ˈ', 'h ɜ', 'ˌ e', 'd ˌ', 's ɪ ˈ', 'u ˈ']
Top features for class 8:  ['t', 'r', 'h ɜ', 'ˌ i', 'ˈˈ', 't ˌ ɪ', 'ˌ', 'l a', 'ʰ i', 'd ˌ i']
Top features for class 9:  ['ˌ ɪ', 'r', 'v i', 't ˌ ɪ', 'h ɜ', 'ɜ', 'v ɪ', 'ɛ r', 'n d', 'ː']
```

Model Performances

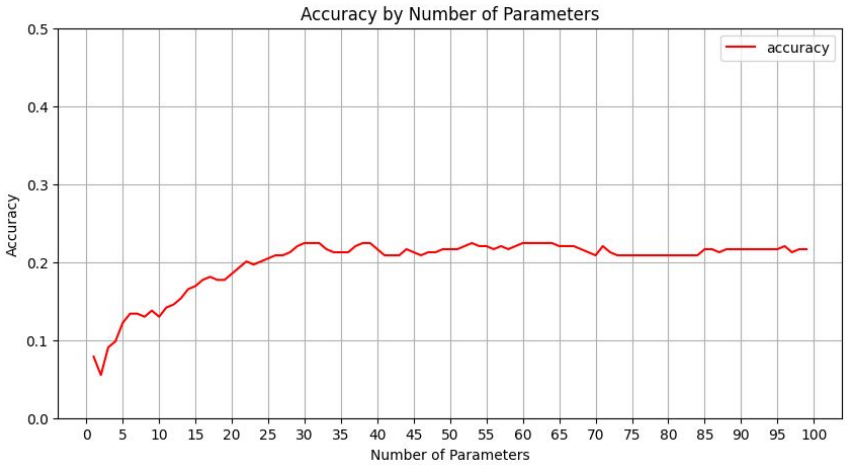
- Basically all 3 of them performed relatively the same
- However… the space representation is very different for each…
  - Panphon features: (1174, 22)
  - Tfidf vectorizer: (1174, 1500)
  - Audio features: (2923, 21)
    - The audio features didn't really care about transcription.



Country Prediction

The simple method

- I treated the countries as tags, and used a classifier. I know very boring
- Again `RandomForestClassifier(n_estimators=100, random_state=42)`
- Accuracy was about 21% which is not too bad considering baseline accuracy is 1/126=0.07%
- Did parameter tuning for the number of estimators, accuracy plateaued around the 30 parameters mark



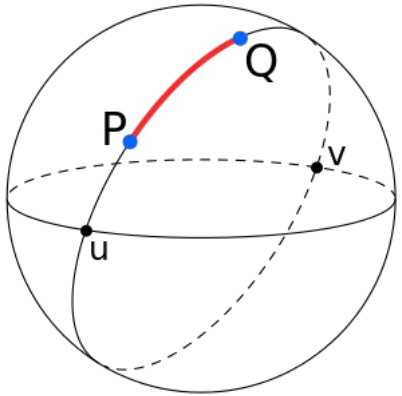
The more fun method

- I leveraged Google's geographic datasets to get the longitudes and latitudes
- Used the Panphon features to train 2 regression models for each longitudes and latitudes
- Then took the pair predictions, found the closest countries in our data (haversine distance) to it
- Converted back to a string label which would be our prediction.

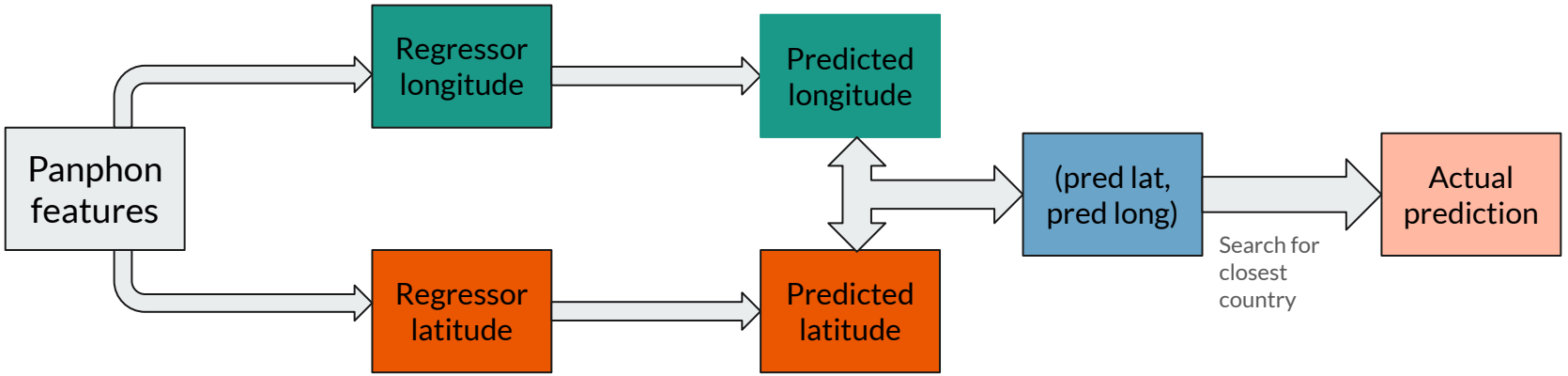
$$\operatorname{hav}(\theta) = \operatorname{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\operatorname{hav}(\lambda_2 - \lambda_1)$$

where

- φ_1, φ_2 are the latitude of point 1 and latitude of point 2,
- λ_1, λ_2 are the longitude of point 1 and longitude of point 2.

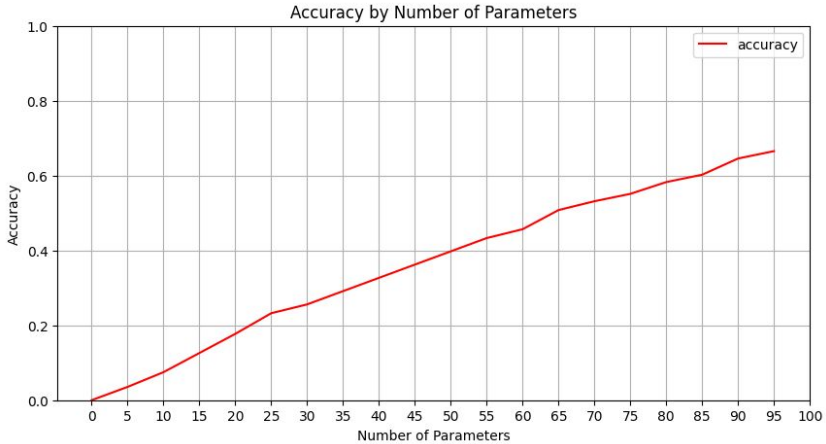


What?? English please??



How did it perform you may ask?

- Not great, in fact very badly, compared to the simple method.
  - Even worse than baseline guessing
- The ME for lat is about 2331 kilometers
- The ME for long is about 6986 kilometers
  - * at the equator
- The R2 coefficients are very close 2 zero indicates almost no correlation and hence the high error margins.




```
Latitude        MSE: 433.87571918958554 R2: 0.053577876284328085
Longitude       MSE: 3937.92735784482   R2: 0.223798986191754 76
Country Prediction Accuracy: 0.0039370078740157 48
```

Final Thoughts and Takeaways

- I tried dabbling with classifying native vs non-native and predicting the continent
  - Both of these tasks were very easy with +80% accuracy.
- Considering the the data imbalances, and the small size, we were already at a disadvantage
  - I tried to implement stratification in the splits to slightly overcome it but it wasn't significant
- With that being said, however, the task of pinpointing the language and/or country from just phonetic features is very difficult inherently but not impossible considering…
  - We can see that when we used tfidf and the audio features as well directly.
- Our models were able to perform dramatically better than baselines (except my crazy experiment) despite the skewed data and the smaller size.
- If our data was purposely designed for the tese prediction tasks, our model performances would improve dramatically.
- When it comes to the experiment, if the latitudes and longitudes were more spread out (instead of the centers of the countries)