Using Machine Learning to Predict Accents

Round Up of Classifying ML Algorithms

Rob Carter

https://github.com/rob-carter/using-machine-learning-to-predict-accents-final

Context

Goal

- With the data set I’ve selected, I’m interested to see how well machine learning can do in predicting the accent of users.
- I think a solid threshold for predicting an accent would be at least 75%. Definitely better than what I would be able to do (I would imagine). I do think, though, that if this were a production level application, you would need a really high threshold (maybe 95% or better).
- To reach this goal I’ve selected numerous ML classifying algorithms and mainly used accuracy as the determining factor.

Context
Why?

- As English is a global language there are many different ways people pronounce words. It is important to be able to understand them, so that programs can better respond to users. Many current apps struggle with users that have strong accents.
- Being able to parse the accent then applying a specific speech recognition setting would increase usability for apps.
- Personally, I just think the intersection of sound and computers is just super cool, so I think this is just fun. For that reason alone this is worth doing.

Problem Definition

- Many speech recognition software struggle to pickup/understand accents. This is an issue for these users, and they’re unlikely to continue using software that doesn’t work for them.
- Siri rarely understand my mother’s accent. This is an avenue that Apple could explore to better their software in the future.

Project Motivation

Potential real-world applications of the project

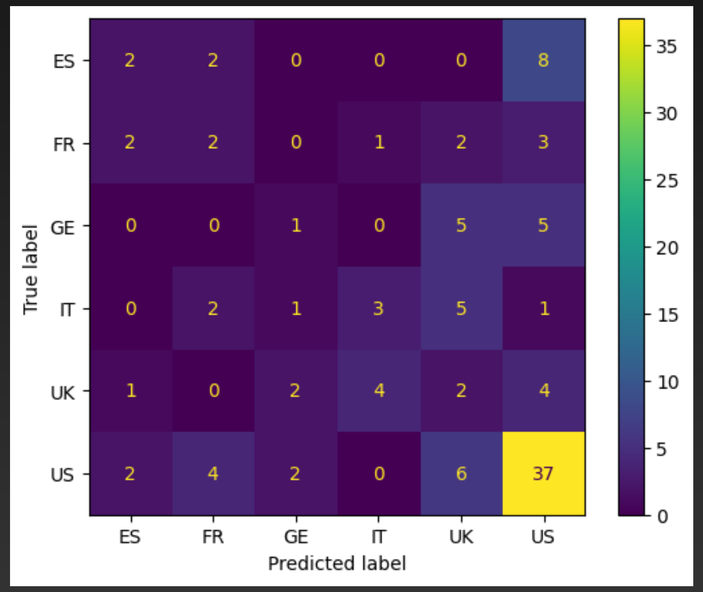
- The most obvious real-world application would be to integrate ML accents recognition into speech recognition software (Siri, Alexa, Cortana, etc...).
- Language learning software would benefit from this as well. It could be used to pickup on accents and help the user reduce them. This could lead to better conversations.
- Potentially increase security. If a program requires voice recognition and the user’s accent suddenly changes, ML could notice this and reject the intruder and save the victim.

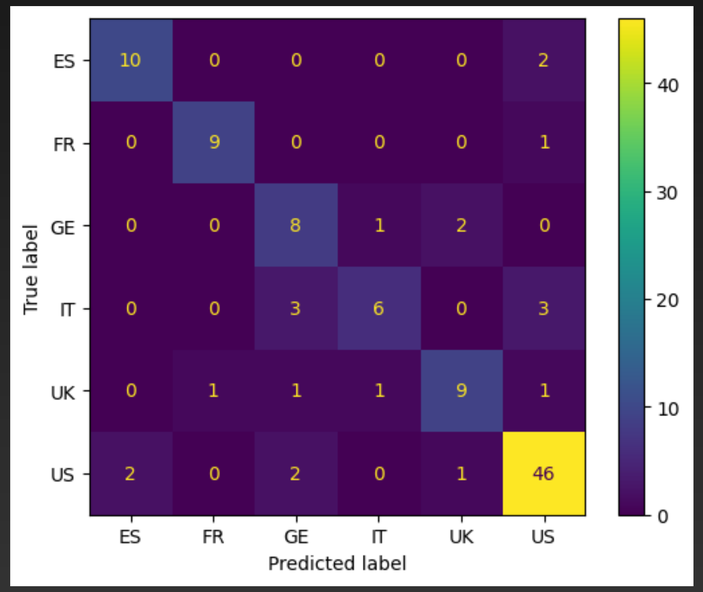
Methodology
The data set

- The data set comprises of a target variable of the possible English accents. The accents are limited to only Spanish, French, German, Italian, UK, and US. This is quiet limited as English is a global language with, probably, hundreds (if not, thousands) of accents. For a learning data set, I think it is fine.
- Issue with the data set: Half of the samples in the data set are US accent. This skews the data. As a result, non-US accents are predicted less accurately.
- Each feature is a segment of an audio signal. This is done with a common speech processing method called MFCC. While I am not too fimilar with MFCC, it seems to be used in industry.
- Data set: https://archive.ics.uci.edu/ml//datasets/Speaker+Accent+Recognition

Methodology
PCA worth doing?

- From my testing, PCA reduced the model's ability to predicted significantly.
- I believe this has to do with the high number of US accents in the dataset compared to all others, and using PCA cut the components essential in getting good accuracy in other accents. PCA0 and PCA1 don't describe the data enough, only 70%.
- Example Confusion Matrices for KNN where left side is PCA and non-PCA is right side. It appears that with PCA, the models can't predict non-US accents at all

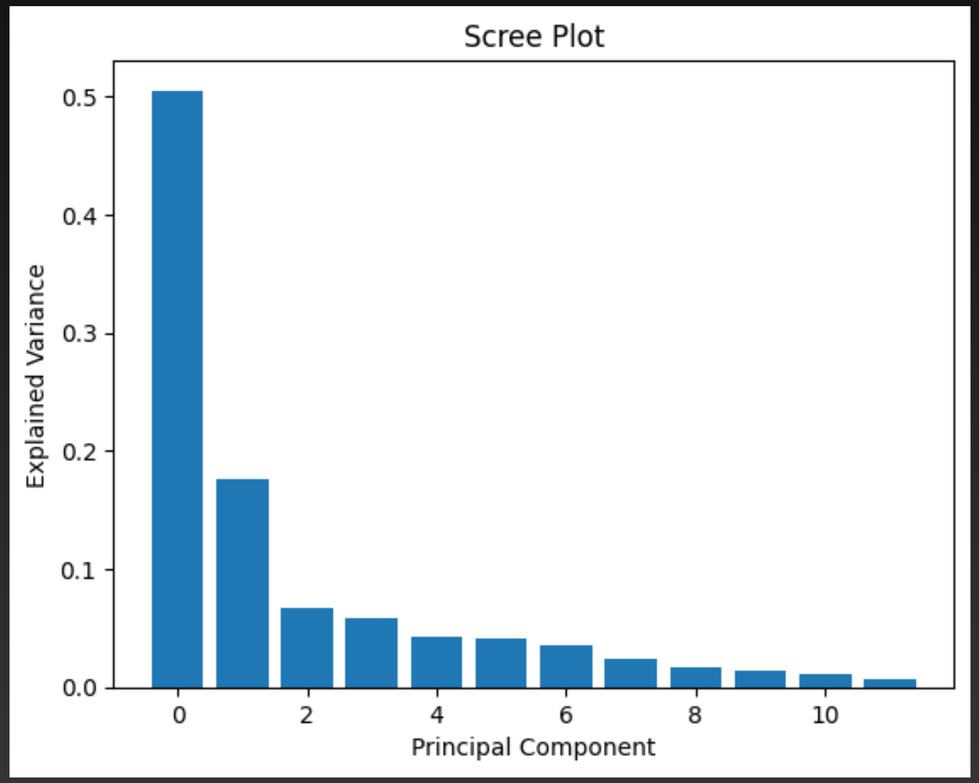


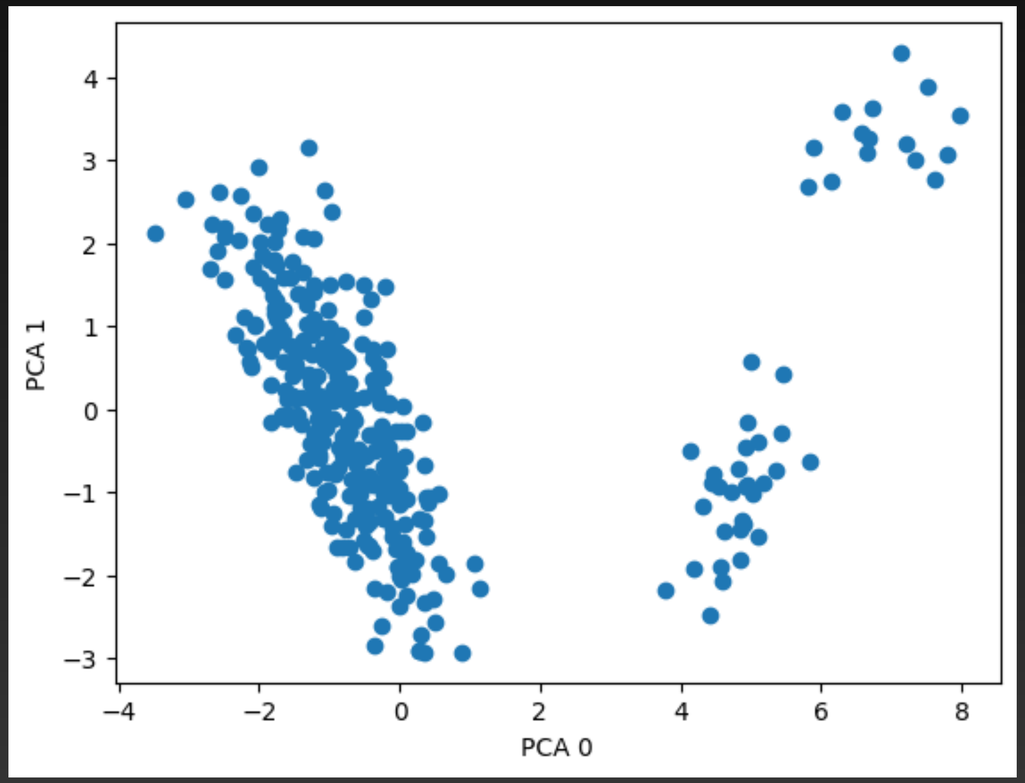


- I opted out of using PCA

Methodology

PCA Graphs for those interested





Methodology

What the data looks like

	language	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12
0	ES	7.071476	-6.512900	7.650800	11.150783	-7.657312	12.484021	-11.709772	3.426596	1.462715	-2.812753	0.866538	-5.244274
1	ES	10.982967	-5.157445	3.952060	11.529381	-7.638047	12.136098	-12.036247	3.491943	0.595441	-4.508811	2.332147	-6.221857
2	ES	7.827108	-5.477472	7.816257	9.187592	-7.172511	11.715299	-13.847214	4.574075	-1.687559	-7.204041	-0.011847	-6.463144
3	ES	6.744083	-5.688920	6.546789	9.000183	-6.924963	11.710766	-12.374388	6.169879	-0.544747	-6.019237	1.358559	-6.356441
4	ES	5.836843	-5.326557	7.472265	8.847440	-6.773244	12.677218	-12.315061	4.416344	0.193500	-3.644812	2.151239	-6.816310
5	ES	6.267089	-4.227738	7.227741	10.069418	-6.953871	9.384771	-15.341538	6.718533	-0.450952	-7.177585	2.538713	-3.550536
6	ES	10.421391	-4.261053	7.472472	9.172961	-7.419083	11.388772	-14.508959	6.557093	0.014110	-8.559910	2.596415	-4.763508
7	ES	8.787293	-5.524667	8.752215	10.001286	-8.810647	11.891619	-13.265491	6.968578	0.903085	-5.969508	1.760507	-6.101706
8	ES	6.754669	-4.621569	10.172625	14.311938	-7.485512	10.126099	-12.335403	5.022380	-0.454547	-9.487129	1.313203	-3.971762
9	ES	9.803577	-4.485605	9.092753	12.236940	-9.276524	9.972296	-14.003346	6.099329	-0.420892	-9.935106	2.556851	-5.887197
10	ES	9.607115	-5.856414	6.804578	13.318068	-8.356424	9.999290	-13.649587	3.190674	1.093630	-8.628709	0.903910	-3.572472
11	ES	10.148804	-3.633411	5.470405	11.264079	-6.624555	12.251751	-12.346634	6.734823	1.264493	-6.930236	2.982636	-5.512447
12	ES	7.598763	-9.123252	8.857363	10.440621	-8.513084	12.140179	-14.739480	4.682152	1.213757	-5.518448	2.276325	-4.331024
13	ES	7.077973	-7.265477	9.474136	9.068401	-8.417684	12.104569	-13.272939	3.462327	-1.225247	-5.040444	2.291653	-5.847239
14	ES	13.744606	-9.938531	3.068951	13.795041	-6.007463	14.317395	-9.658241	0.302305	1.699647	-10.990290	5.381063	0.124498

Methodology

Steps used to incorporate data

- Since I'm not using PCA, the dataset is already cleaned and prepped for usage.
- Import the dataset to a data frame.
- Separate the target from the rest of the data.
- Generate testing data using a third of the dataset: train_test_split(X, y, test_size=0.33, random_state=42)

Methodology

How the models were trained

- For each model I first created the specified class (all from sklearn excluding the boosts)
- Fit it with the data
- Ran prediction against target
- Mess with hyper parameters to see if any increased the accuracy score
- Use accuracy score to determine its value in this project
- Print the confusion matrix to get a better understanding of what is going on

- Note: I tried many different hyper parameters. If I don’t mention them it is likely they had little to no effect on the accuracy score.

Methodology
ML models

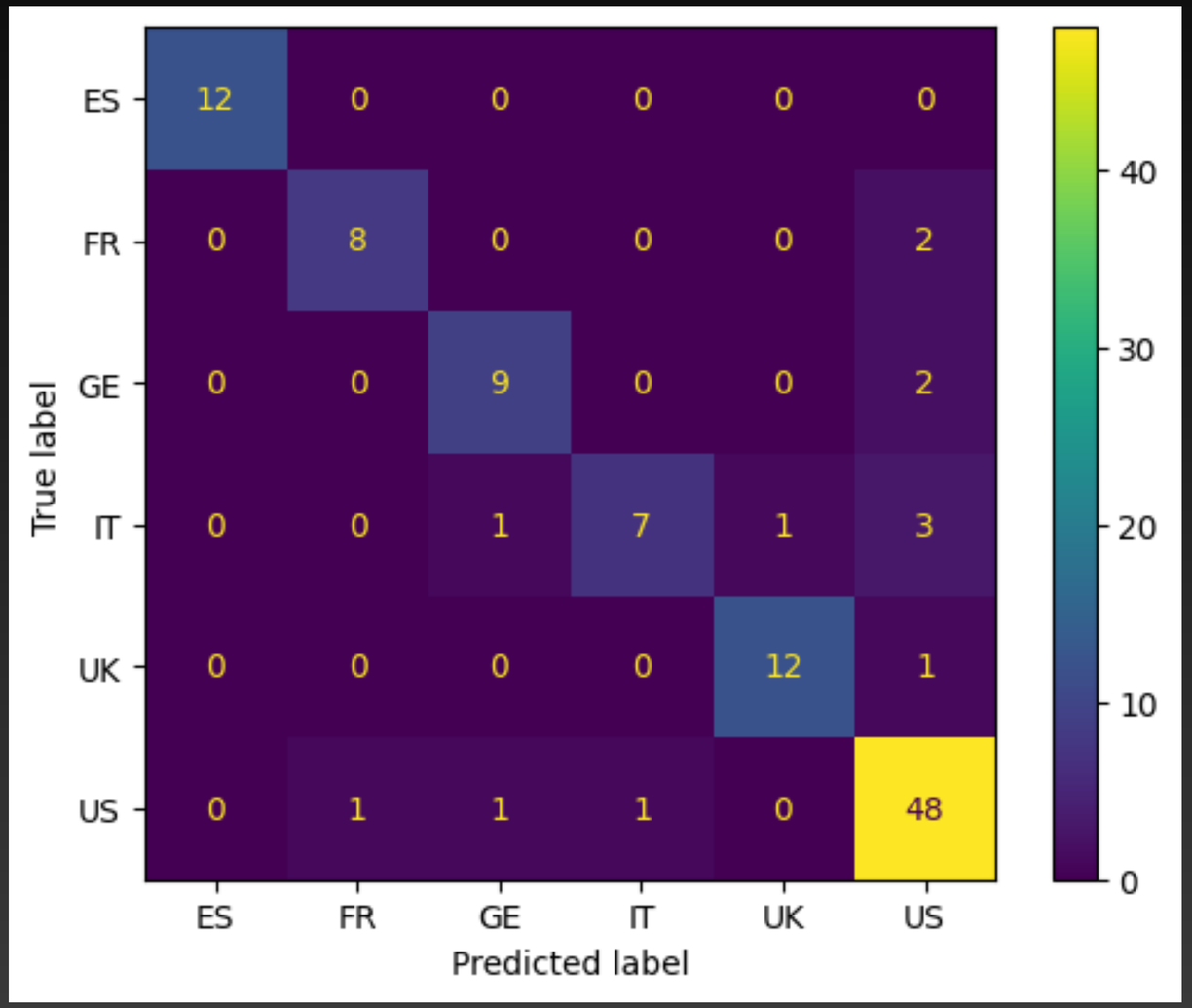
- I selected various classifying machine learning algorithms, this includes:
- Support Vector Machine
- K-Nearest Neighbors
- Naive Bayes
- Decision Tree
- Random Forest
- Various Boosting algorithms

Findings
SVM

- Running the model with hyper parameters set to default settings gave an accuracy score of 0.52. This is well below the threshold I selected.
  - Initially I found that a C of 15 returned a accuracy of 0.84. This is a huge improvement!
- I was curious though, and looped from 1 to 10000 to see what C would get me the highest accuracy. I tried to calculate up to a million, but the time to run this was too long.
- I’ve found that C=49 was the best at an accuracy of 0.88!! This is an amazing result

Findings
SVM visualized

- This is the confusion matrix of the SVM with C set to 49
- It honestly did amazing. I didn't expect anything to reach this high of an accuracy.
- It appears that other accents are predicted to be US when they're not. This is a common occurrence in the other models.
- I believe this is mostly due to the fact that a majority of elements being US and it overfits the data so that there is a US accent bias.

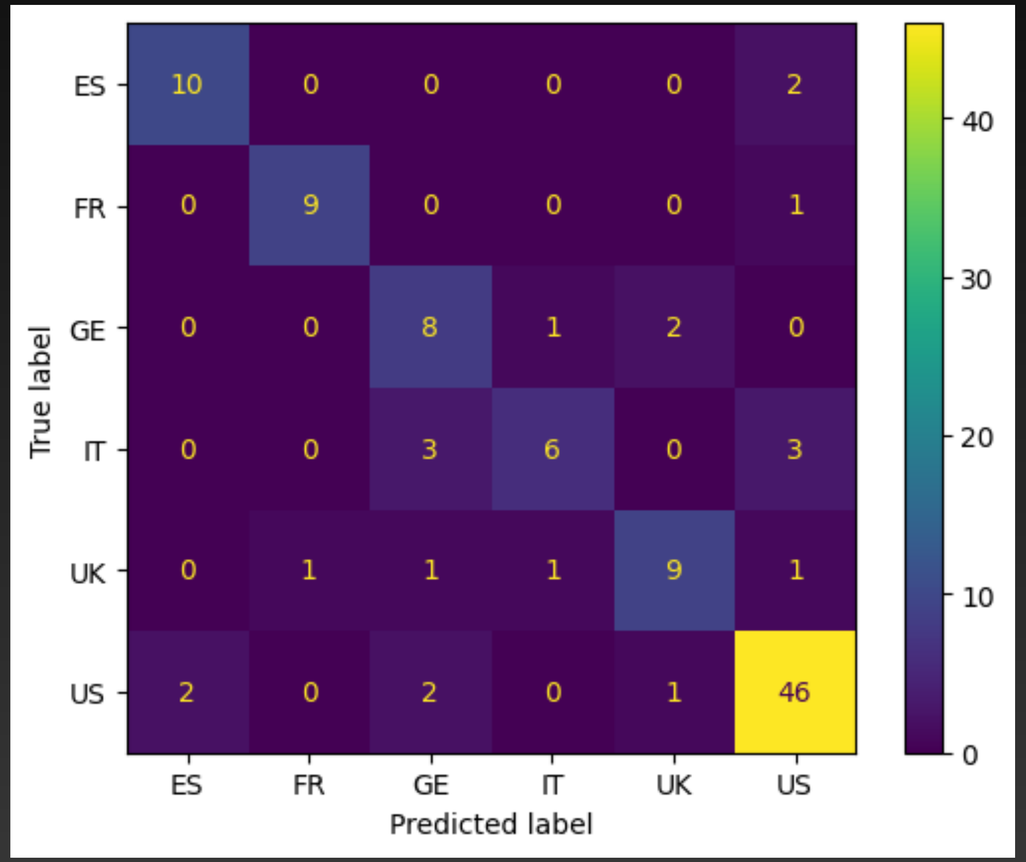


Findings
KNN

- Default hyper parameters gave an accuracy of 0.78
- Setting the hyper parameter weights to distance increased it to 0.81
- This is above the threshold, so it is a good contender.
- Not as impressive as SVM.
- I also messed with the n_neighbors, p, metric, leaf_size, and got worse results.

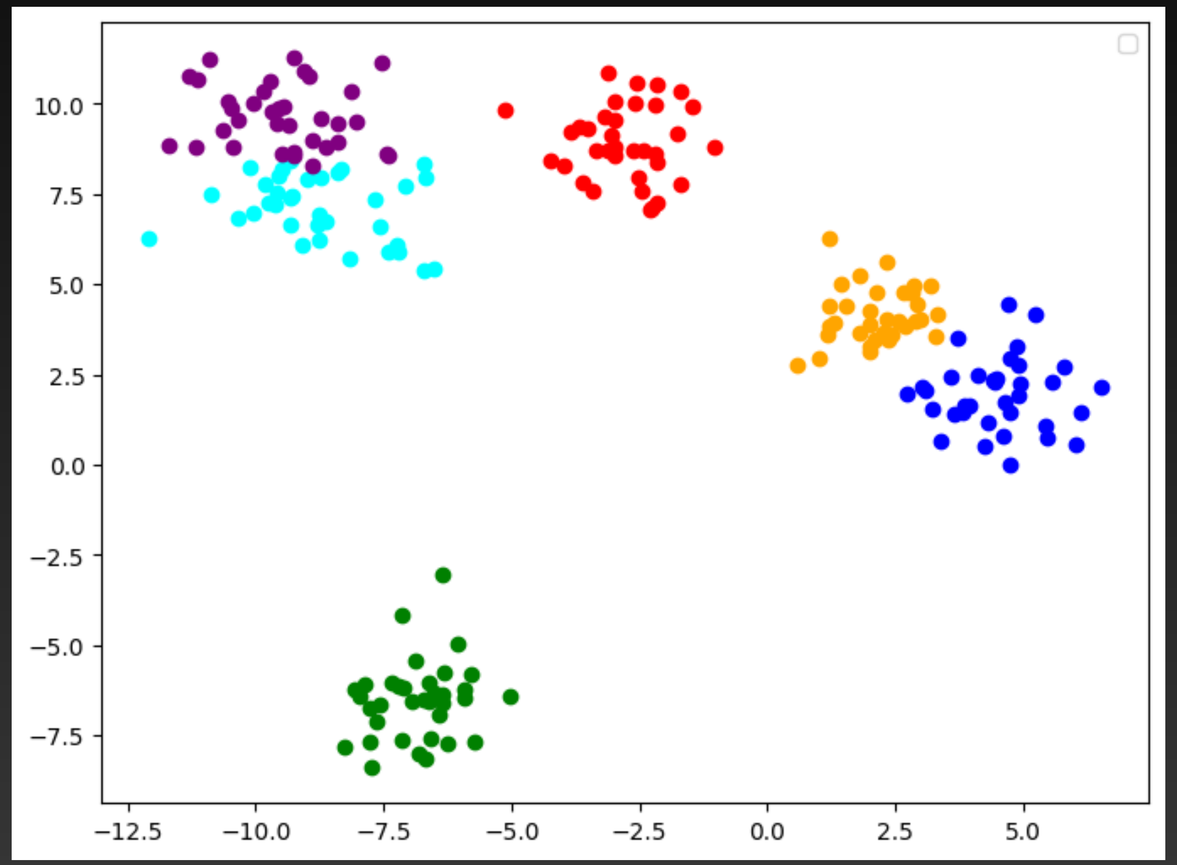
Findings
KNN

- It appears this model struggles the most with differentiating German with Italian and US accents.
- The US accent bias isn’t too large in this model.



Findings
KNN

- This only accounts for the first 2 features (I don’t know how to graph more features without PCA), so it is not an accurate representation of all 12 features.
- I did think it was worth including to visualize some of the over lap seen between the different accents.

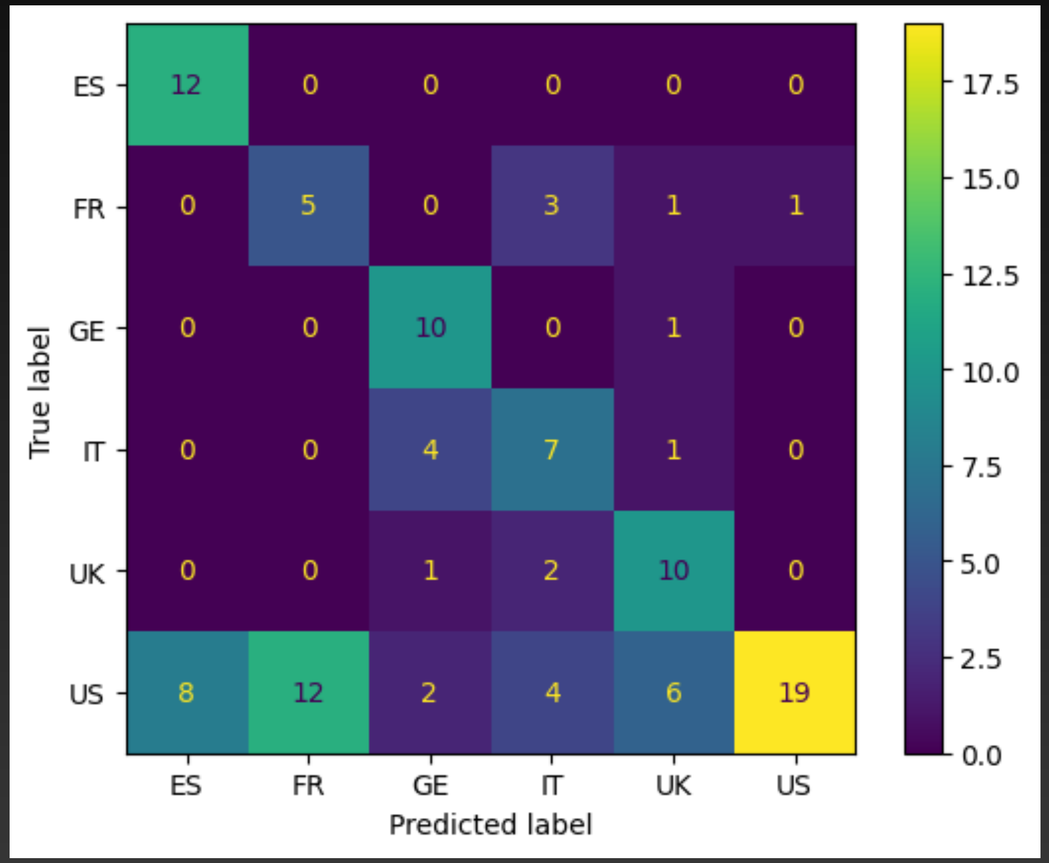


Findings
Naive Bayes

- Using the default parameters gave an accuracy of 0.58. This isn’t a good model for the data.
- This was a disappointing result, and the lowest accuracy of all the models.
- There are numerous reasons I think this might be such a weak model. It could be that the dataset is too small for bayes to work well, or the imbalance in number of US classes.

Findings
Naive Bayes

- It appears to greatly under predict the US accent.
- It is the only model to predict Spanish or French to be US by a significant amount.
- If it ignored US, it did quiet alright. However US accent is important and a part of the data set.

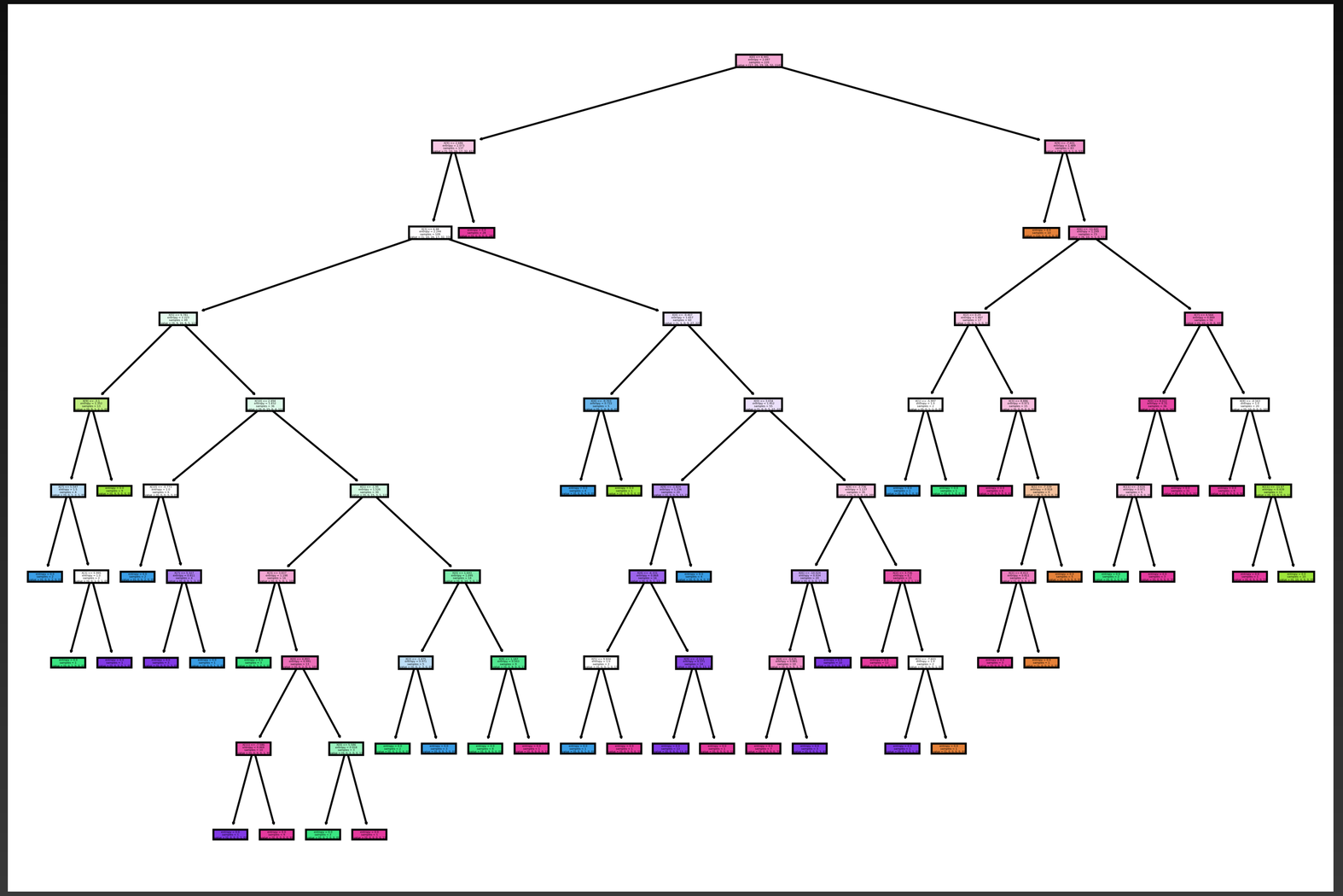


Findings
Decision Tree

- I knew that the decision tree would perform worse than a random forest. I was curious though to see that in action.
- Default settings had the accuracy score come out to 0.60
- Setting the criterion hyper parameter to entropy greatly increased the the model’s accuracy to 0.70
- Too low to be a good result with the 0.75 threshold.
- Tried a bunch of different hyper params. Messed with splitter, max_depth, and min_samples_split. Didn't see appreciable improvement over just setting the criterion to entropy

Findings

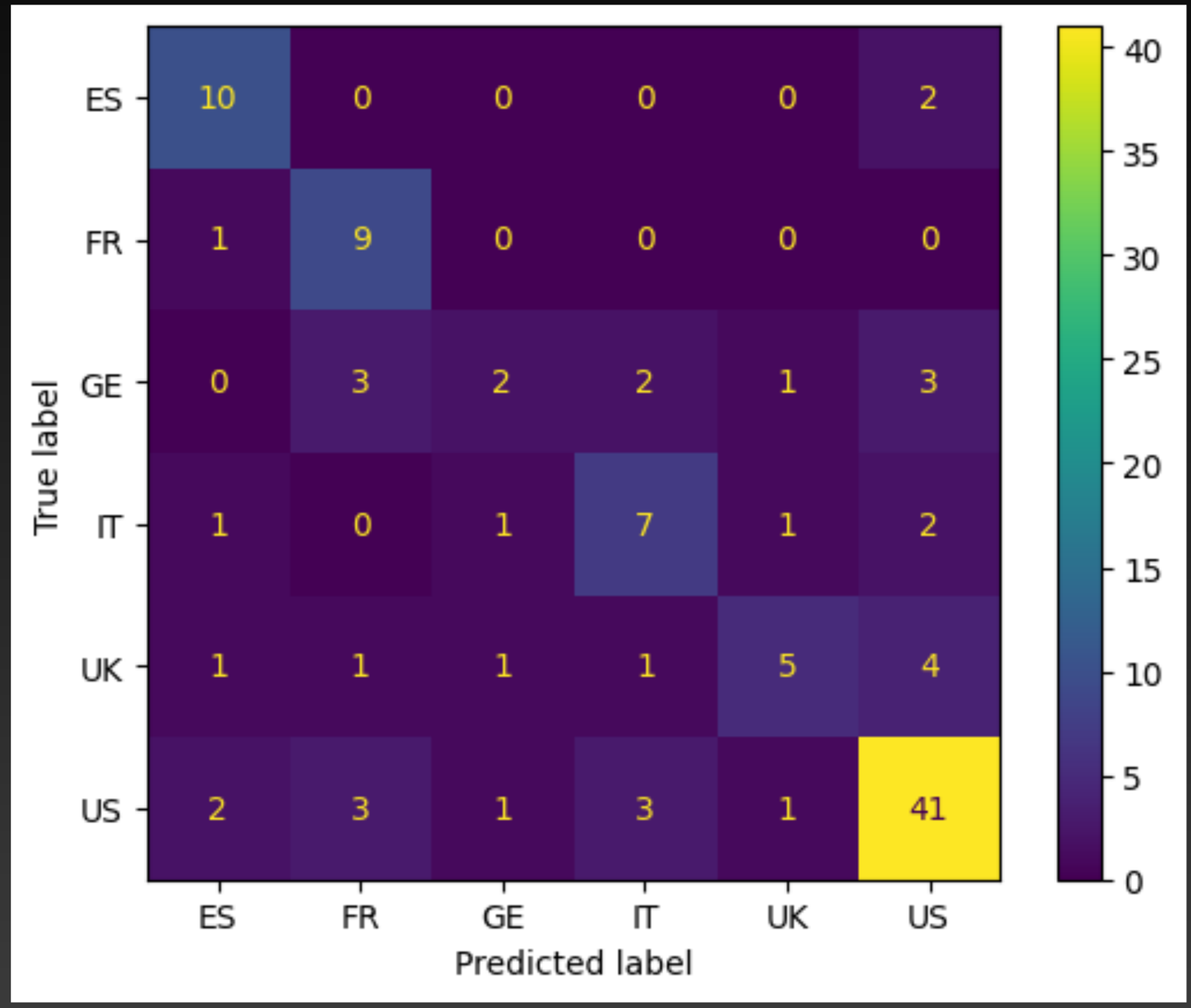
Decision Tree Visualized



Findings

Decision Tree Visualized

- This model is average... it does alright.
- It appears to struggle with German accents. It guessed French and US accents to be German.
- Nothing else too interesting to comment on here. Not amazing, but not half bad.



Findings

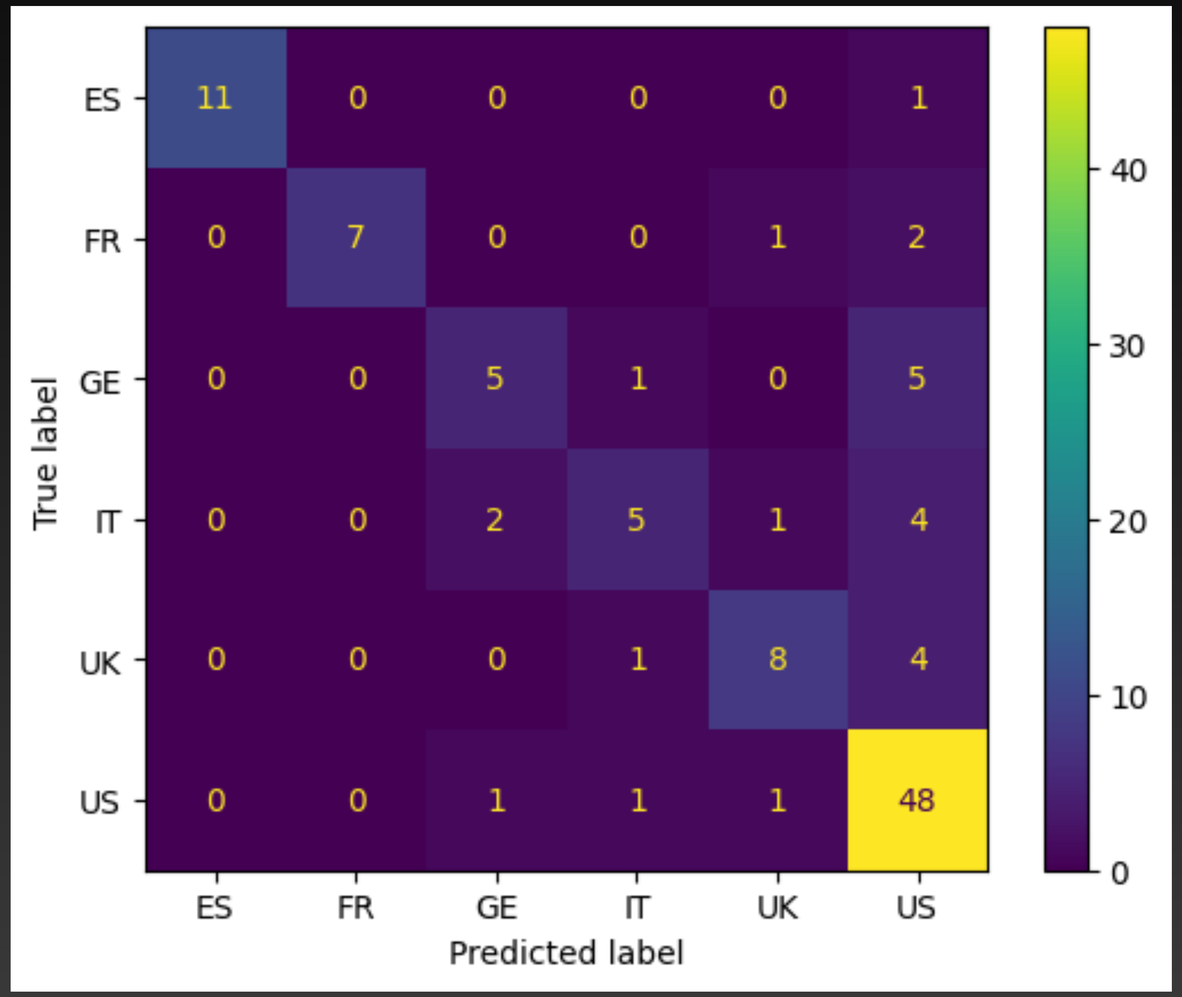
Random Forest

- Default hyper parameters gave am accuracy score of 0.75
- Setting criterion to entropy increased accuracy score of 0.78
- Right on the edge of the threshold.
- As expected these are better results than a decision tree.
- Like the decision tree, messing with the other hyper parameters only had negative effect to no effect.

Findings

Random Forest Visualized

- With the confusion matrix, we can see that the model is overfitted
- It is predicting US accents too much. We can see that it is predicting German, Italian and UK accents as US accent.
- I am sure that this could be improved with more work.



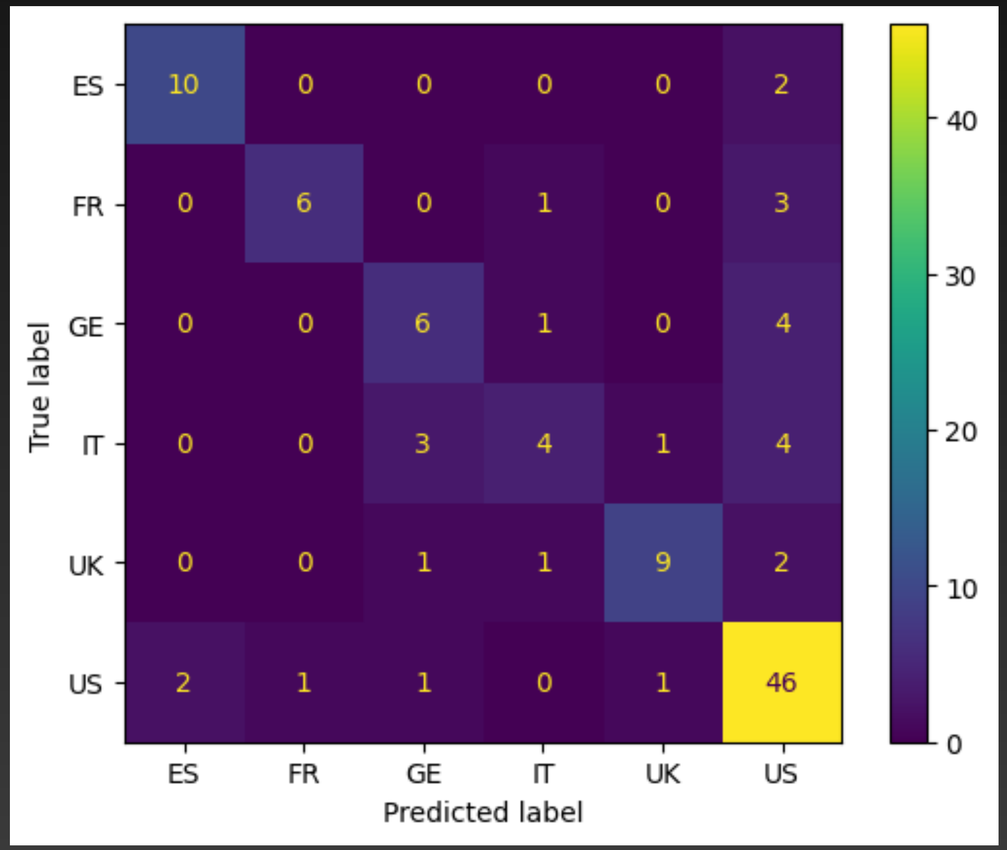
Findings
Boosting

- Boosting algorithms I used:
- XGBoost
- Gradient Boost
- CatBoost
- Using an ensemble voting method to get better results

- Note: I tried to use lightgbm, but the install kept failing. I couldn’t fix it.

Findings
Boosting - XGBoost

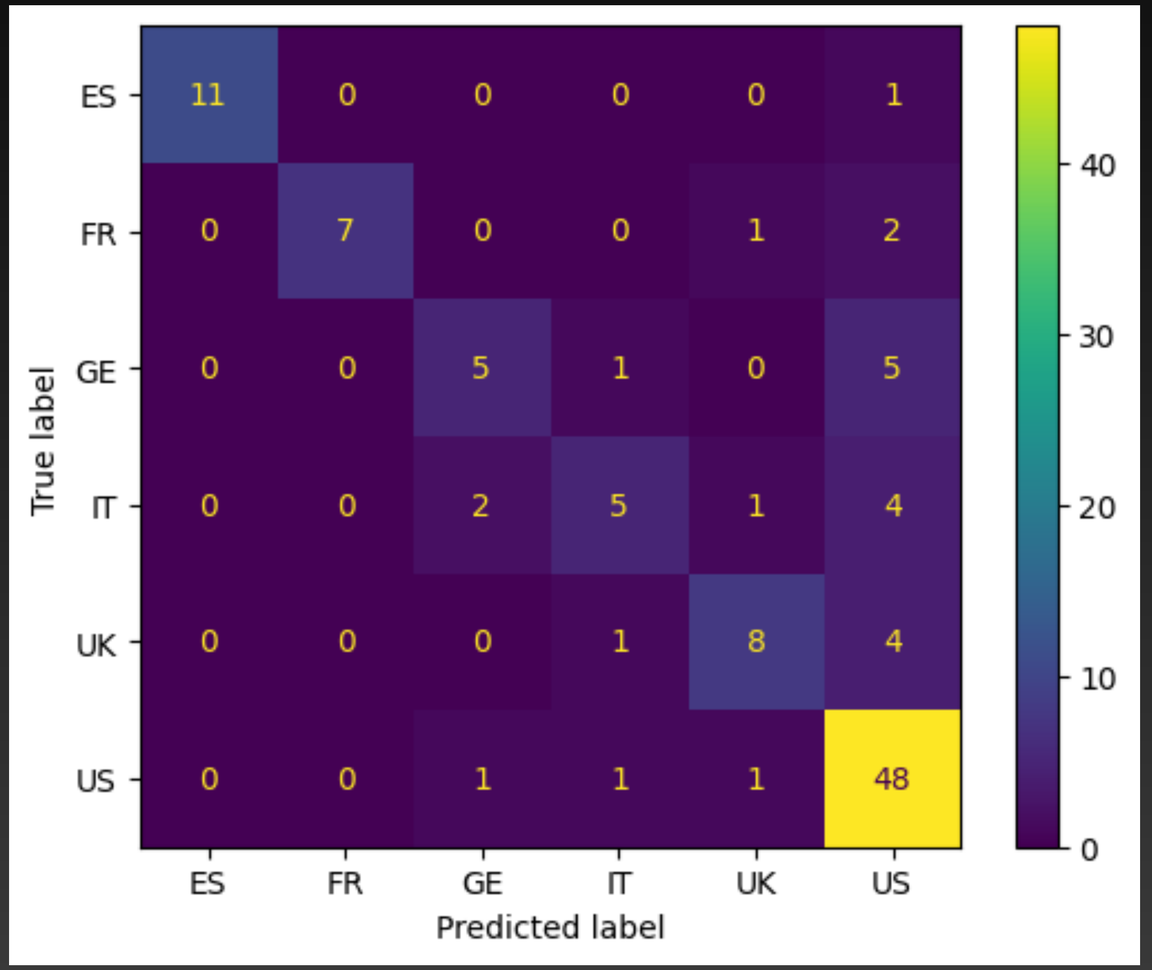
- Required a bit of finesse to get it to work. Finally I was able to get it to work and got an accuracy score of 0.74.
- Just barely below the threshold!
- Appears to over predict Italian, German, and French as US accent.
- This is similar to what we've seen in random forest where it has a US accent bias.



Findings

Boosting - Gradient Boost

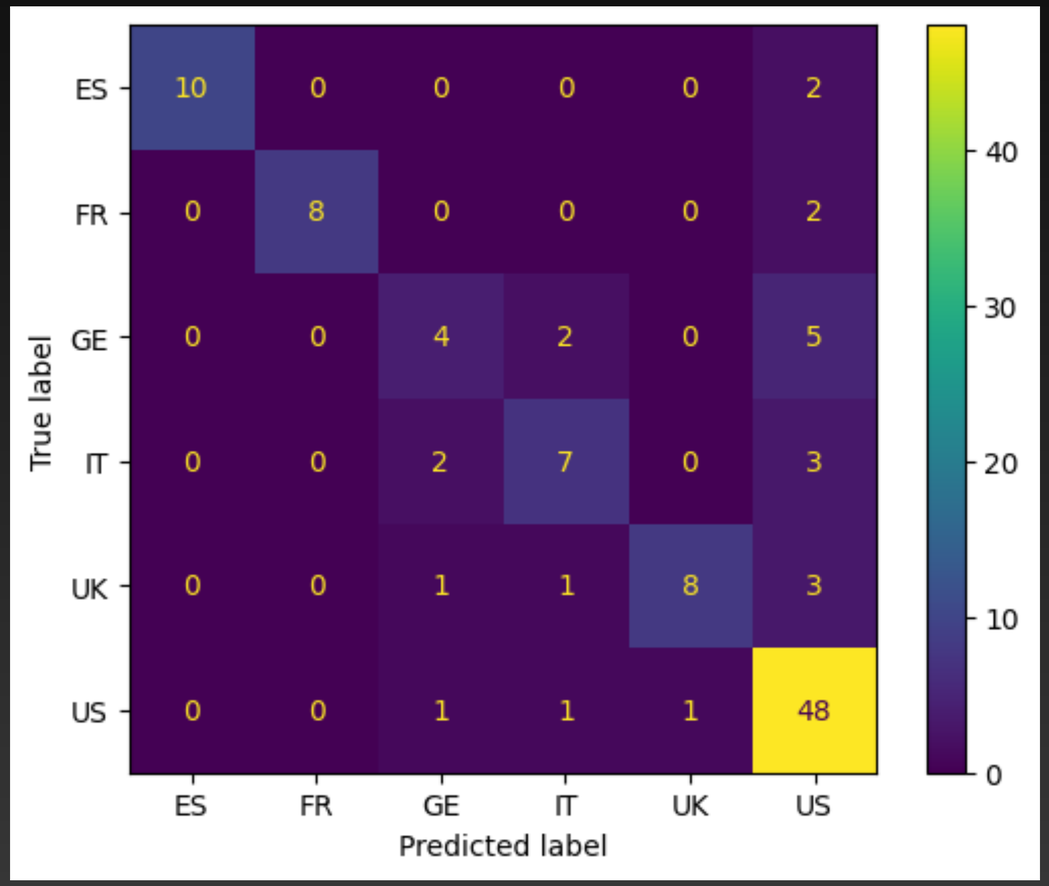
- Accuracy score of 0.67.
- Not a great result and also well below the set threshold.
- Also has a greater US accent bias than XGBoost.



Findings

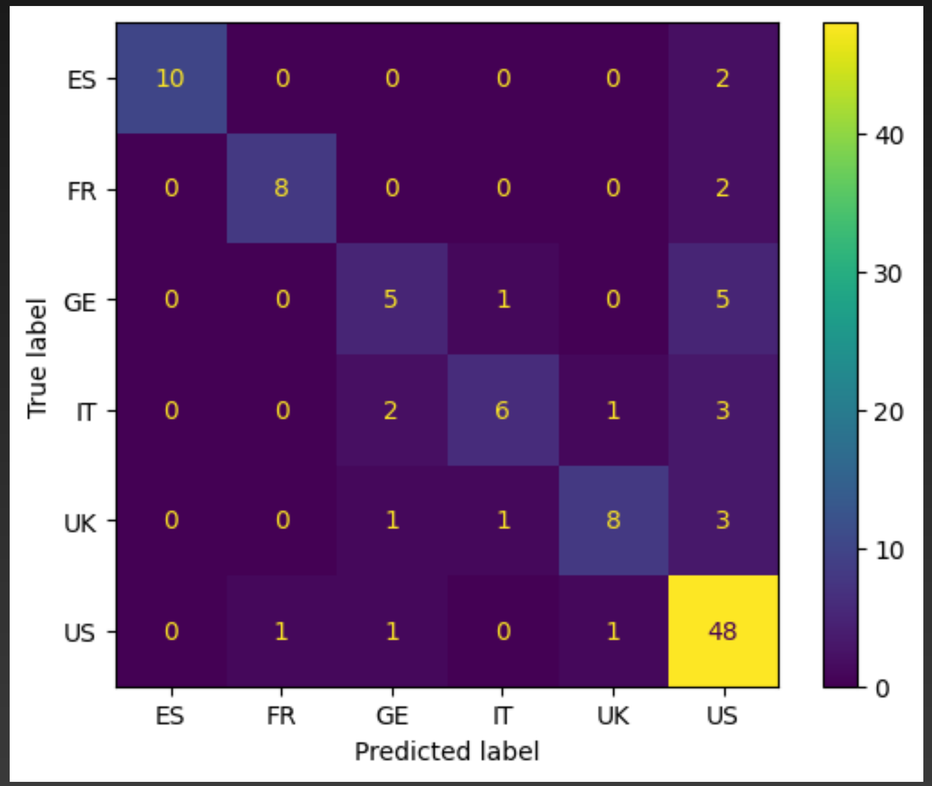
Boosting - Cat Boost

- Accuracy score of 0.78
- Unsurprisingly the best result of the boosts
- Also appears to has a US bias



Findings
Boosting - Voting

- I used the voting ensemble method to combine all the boosting models to try to get a better result. This worked a little, as it increased the accuracy score to a steady 0.80.
- Was better than the self imposed threshold.
- I was a little disappointed by this result.



Conclusion

Summary of Accuracy Scores (from best to worst)

- SVM with C set to 49: 0.88
- KNN with weights set to distance: 0.81
- Voting with Boosts: 0.80
- Random Forest with entropy: 0.78
- Cat boost: 0.78
- XGBoost: 0.74
- Decision Tree with entropy: 0.70
- Gradient Boost: 0.67
- Naive Bayes: 0.58

Conclusions

- From the models performance, we can see that the SVM with C set to 49 gave us the best prediction potential at 88%. I think this is really cool, and I am excited that it works so well.
- KNN and Voting were close behind, but slightly less accurate. The voting ensemble was fun to implement.
- I didn’t initially expect the Naive Bayes to perform so poorly, however upon reflection it makes sense.
- It is clear from all the confusion matrices that the models suffer from a US accent bias.

Conclusions

What model to use?

- For accent recognition the best model to use is SVM with C set to 49.
- I still believe that the 2nd and 3rd place could reach or even surpass SVM if they were modified in some way. Maybe different Boosting algorithms for the ensemble.

Conclusion

Improvements and Limitations

- In future, this project could improve from using more ML algorithms. There could be a better one I've yet to try.
- Although I did a through job messing with the hyper parameters to try to increase the prediction accuracy, I'm sure someone else could get better results than I did.
- There is clearly a US bias in the dataset. This is visible in the many confusion matrices, and the fact the dataset is 50% US accents. Might be interesting to just test if the accent is US or not. I would think the model's accuracy would increase, however that is much less useful for real world applications.
- The data set it self is limited in the number of accents, and they're specifically only European accents. This is fine for this project, but if it were to ever be applied to major project it would be insufficient. A better dataset would make for a better end product.
- The US accent as a singular classification is also some what dubious. A Bostonian is very distinct from a Southern one. UK is well known for many different accents as well.

Ethical Implications

- I originally didn’t realize that there could be potentially be negative uses for something like this.
- The current data set isn’t made to work with many of the accents found through out the world. If it were carelessly implemented it could lead to accidental racism. It would very specifically only work with the previously mentioned accents.
- Being able to identify an accent in general could have negative implications. An application could deny use if it detects certain accents, or worse be used to track/attack minority groups.

Appendix
Code!

- I created a GitHub with my juypter notebook, dataset, and power point
- URL: https://github.com/rob-carter/using-machine-learning-to-predict-accents-final