EE 257 - Machine Learning for Electrical Engineers

Term Project

Under the Guidance Of

Prof. Birsen Sirkeci

Spring - 2022

Speech Accent Recognition Dataset (2020)

By

Bhavin Patel (015954770)

Chapter 1: Dataset Description

1.1 Data Set Origin

The Speaker Accent Recognition Data Set(2020)[1] from the UCI Machine Learning Repository contains data extracted from 329 audio recordings of speakers from six different countries. The objective of this dataset is to classify each audio into 6 different accents. Out of all the speakers, 165 are from the United States, 45 from the United Kingdom, 30 from Italy, 30 from France, 30 from Germany, and 29 from Spain. Each audio recording contains an English word spoken by a native speaker. Each audio recording was pre-processed and the resulting 12 Mel-frequency Cepstrum Coefficients (MFCC) are provided in the dataset. The dataset does not include all the 329 audio recordings and only includes their MFCC coefficients. It also includes some sample audio for each accent.

1.2 Understanding the MFCC Features

This is an informative section describing MFCC in brief. Although this section may not directly help in developing a classification model, It is useful in understanding what our input attributes X1 to X12 represent.

The raw time-domain speech signal is difficult to analyze and interpret, so we extract the frequency component of the audio signal for further analysis. The MFCC further processes this frequency component and provides coefficients that mimic the human-like perception. The idea behind MFCC is that our ears find it easier to differentiate between lower frequencies than higher frequencies. So we use Mel-Scale to map actual frequency to the frequency bands that the human ear will perceive and then we calculate coefficients for each band.

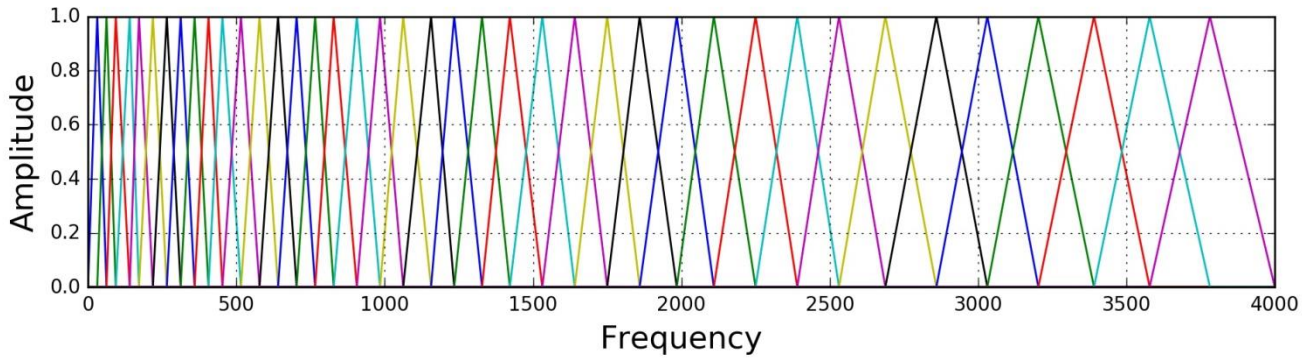


Fig 1. Mel Filterbank

In the above, each triangle represents a digital filter of the bandwidth determined using Mel-Scale. Here lower frequencies are given higher resolution by having more narrowband filters to support the main idea behind MFCC. After that, we calculate how much energy is present in each filter and then take their logs because we don't hear loudness on a linear scale. Finally, we take its DCT(Discrete Cosine Transform) which gives us MFCC coefficients. Generally, we calculate 40 coefficients but for speech processing, we take only the first 12 as they contain most of the information of a speech signal.

1.3 Properties of Input Variable

The 12 MFCC coefficients described in the previous section are the 12 input attributes X1 to X12 in our dataset. There are a total of 329 sets of X1 to X12 MFCC, one set for each audio. So we have a total of 3948 MFCC in the dataset. There are no missing values in the dataset.

Generally MFCC is produced by windowing the audio sample and generating a set of MFCC for each window period. Therefore each audio results in an array of MFCC sets. However, to summarize the whole audio in just one set of MFCC, all such sets produced by windowing are combined using some summary statistics such as mean and std. deviation.

Each MFCC in this dataset is a floating-point value which typically varies between -20.0 to 20.0. Below are the summary statistics.

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12
count	164.000000	164.000000	164.000000	164.000000	164.000000	164.000000	164.000000	164.000000	164.000000	164.000000	164.000000	164.000000
mean	4.597671	-3.537878	1.878479	7.953284	-6.794339	10.657298	-10.186938	5.291842	-0.381418	-4.416863	3.204355	-4.612296
std	4.602879	3.319799	2.990142	3.247887	2.612983	1.905411	1.950640	1.905319	1.496844	2.835827	2.005803	2.200917
min	-6.067831	-11.483628	-4.776980	-0.725240	-12.908842	6.361658	-15.341538	0.302305	-4.820865	-11.429178	-1.606830	-9.614959
25%	1.675449	-5.896806	-0.229998	5.508858	-8.566801	9.243085	-11.383370	4.206731	-1.346694	-6.021932	1.951646	-5.976913
50%	4.058216	-3.542518	1.271110	8.035463	-6.924507	10.668017	-10.027093	5.180216	-0.319201	-3.733782	3.376090	-4.870921
75%	7.287269	-0.924009	3.753992	10.018319	-5.499433	11.892141	-8.967664	6.594619	0.595746	-2.218678	4.618655	-3.214954
max	15.195717	3.526736	10.172625	16.178942	1.411211	14.851000	-3.884775	9.567757	3.866759	0.707209	9.166066	1.079622

Table 1. Summary Statistics of MFCC Features

The above summary statistics give some idea of dispersion and central tendency of our dataset. Most of the X5, X7, X10, and X12 coefficients are negative whereas most of the X4, X6, X8, and X11 coefficients are positive according to min and max. The X9 is narrowly centered around zero.

Range, standard deviation, and interquartile range indicate that X1 to X4 are spread across a wider range and X5 to X12 are concentrated around their mean. The mean and median suggest that X1, X3, X5, X7, X11, and X12 are slightly right-skewed and X8, and X10 are left-skewed. The remaining X2, X4, X6, and X9 are almost symmetrical. There are a few outliers mostly from X5 and X7.

1.4 Properties of Output Variable

There is only one output attribute with the name ‘language’. The output variable is a label that can have 6 possible values ES, FR, GE, IT, UK, or US for Spanish, French, German, Italian, British, and American accents respectively. Each audio must be classified into one of these accents. About half of the output in the dataset is from the US and the remaining are spread across each accent. From general observation, there seems to be no simple and visible relation between a particular MFCC and a particular accent. For example, the UK accent being dependent on X5 MFCC. One major issue with this dataset is that it is unbalanced. It has half of the sample with the US accent and the other half is divided equally amongst the 5 remaining classes.

Chapter 2: Dataset Visualization

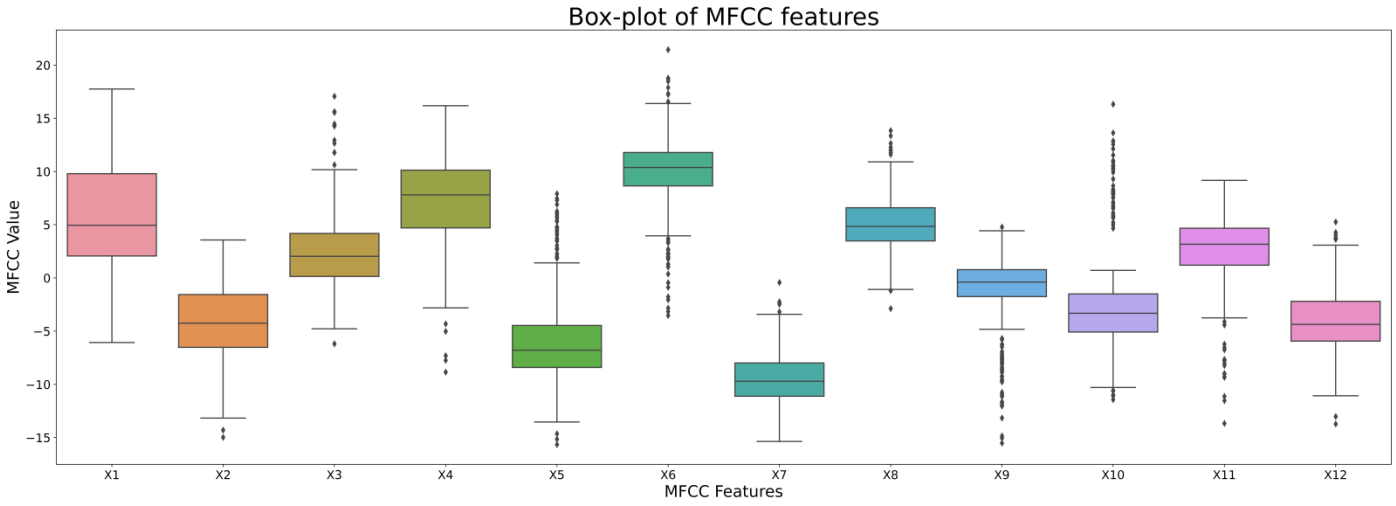


Fig 2. Box-Plot of MFCC Features

The above box-plot of MFCC features helps us to understand how the input variable is spread. According to the box plot, most of the X5, X7, X10, and X12 coefficients are negative whereas most of the X4, X6, X8, and X11 coefficients are positive. Also, X1 to X4 are spread across a wider range compared to X5 to X12. One notable feature is X9 which is very narrowly centered. Therefore it may contribute the least to the prediction and may get eliminated with dimension reduction if required.

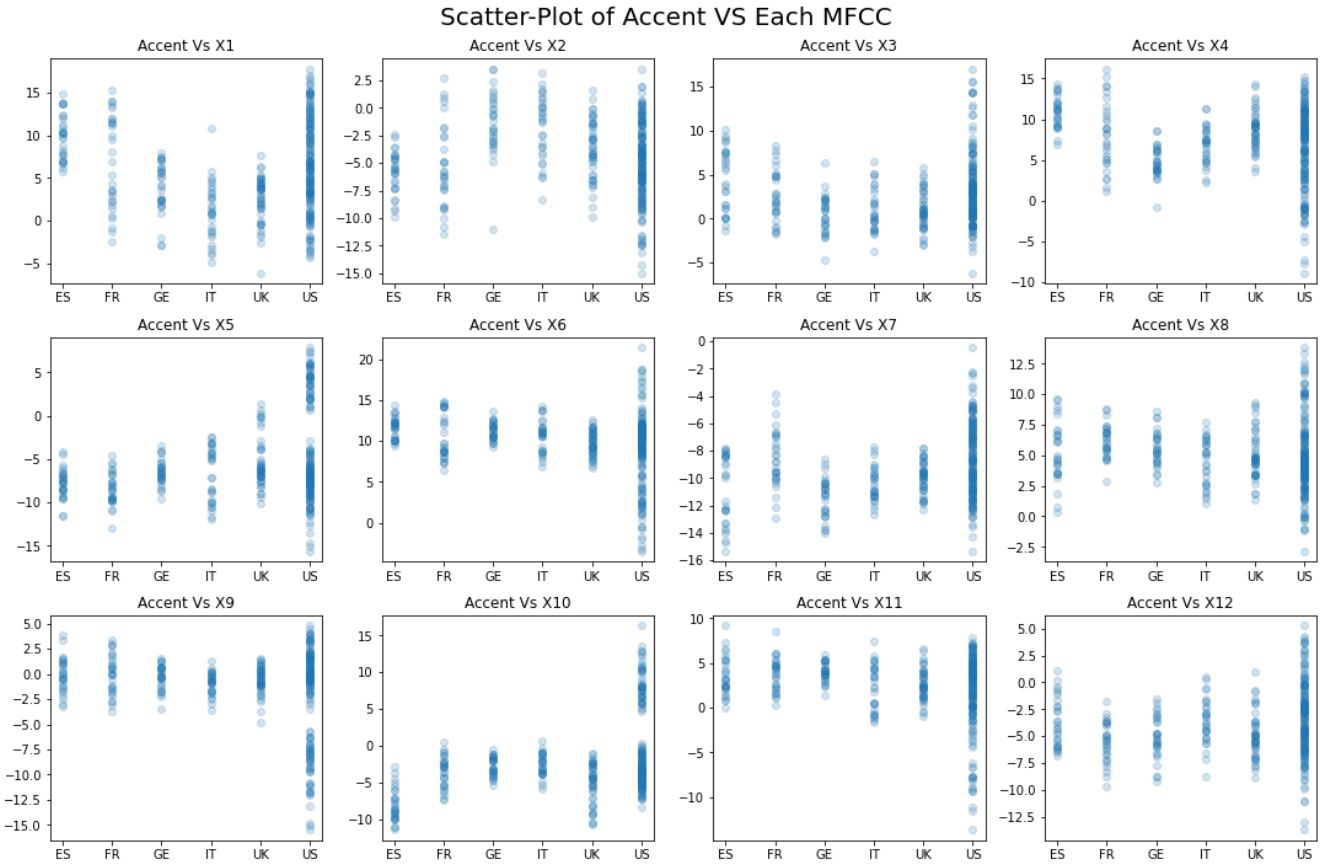


Fig 3. Scatter-plot of Accent vs Each MFCC

The scatter plot above for accent against each MFCC is used to check if any particular input variable is strongly associated with a language. For example, the US accent being dependent on the X11 variable.

However, I could not find any input variable which would distinctively predict a particular accent because the range of distribution of all the input variables X1 to X12 are overlapping for each accent. For example, X11 has similar distribution for each accent.

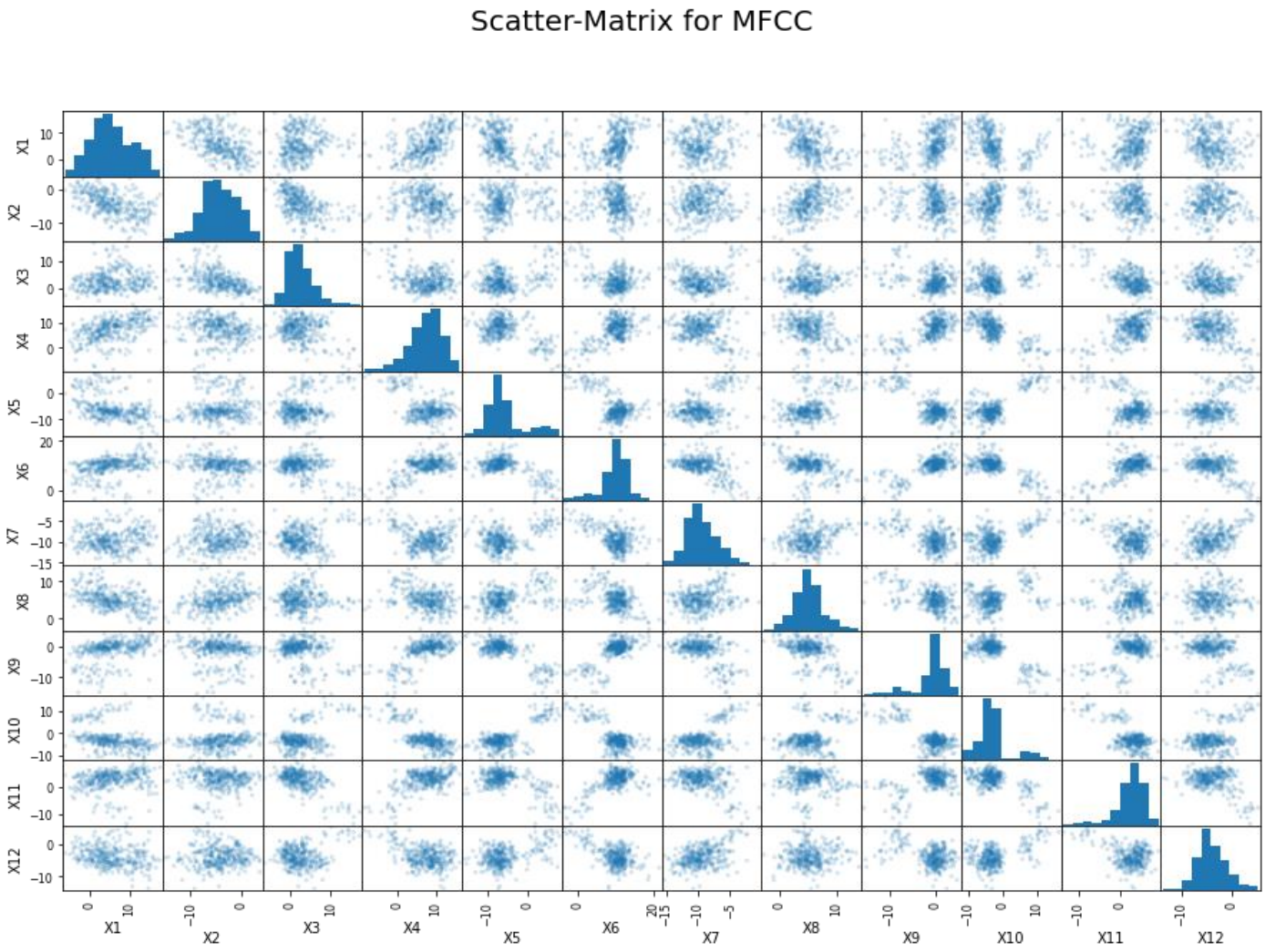


Fig. 4 Scatter Matrix

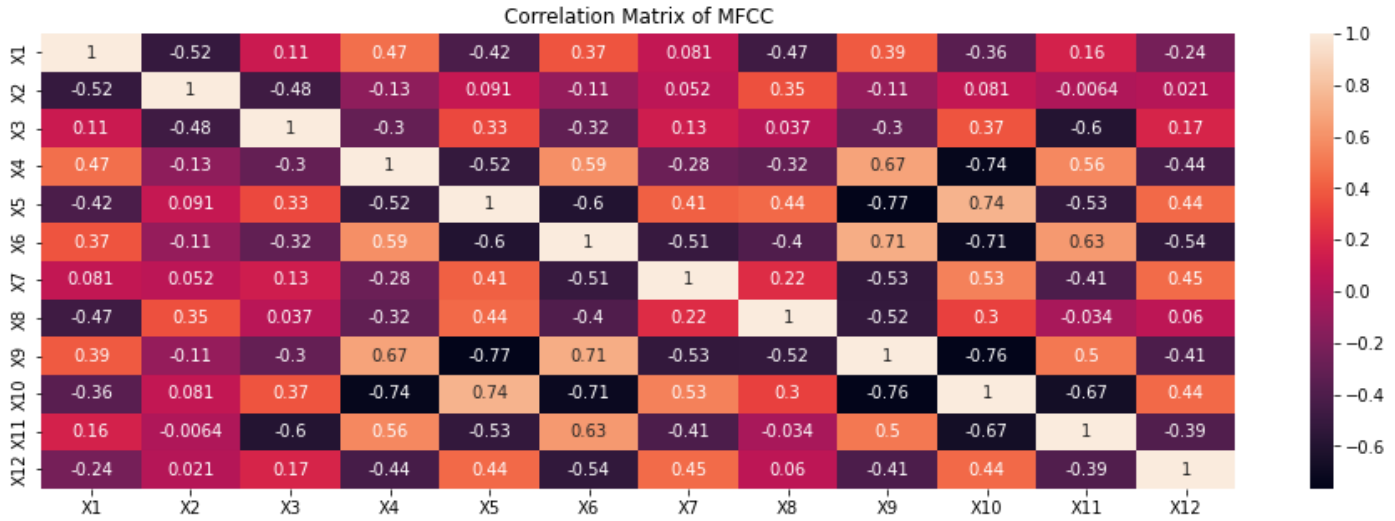


Fig. 5 Correlation Matrix

The previous two plots include a correlation matrix and a scatter matrix to check the correlation between input variables. The correlation matrix indicates that there is a moderate correlation between a few MFCC coefficients but there is no strong correlation. The scatter matrix also supports this as the data points are scattered in between each MFCC coefficient which makes it difficult to establish a relationship between any two coefficients.

Chapter 3: Data Set Cleaning

The dataset does not have any missing values or null values. Initially, the outliers were identified using the 1.5 x (IQR) method. This resulted in a total of 68 rows having outliers in the original dataset. Most of them were from the US label. However, CV results after removing the outliers indicated a decline in the accuracy of all models. The average drop in accuracy was around 5-7% for all models. It means that the outliers were falsely detected in this case. So I reran the outlier detection but with a more flexible factor of 3 x (IQR). This time the total number of rows with outliers was reduced to 30, and all of them were from the US label. The CV showed that the models trained after dropping these 30 samples had about the same or a little improved accuracy over the models trained without dropping outliers. The number of outliers detected for each factor is shown in the table below.

N	**Number Of Outliers**
1.5	63
2	50
2.5	41
3	30

*Table 2. Outlier Detection With {(Q1 - N * IQR), (Q3 - N * IQR)}*

Outlier detection becomes tricky when a dataset is very small. This applies to this dataset because it is unbalanced and 4 of the 6 labels only have 30 data points each. This is significantly less compared to the US label with 165 samples. So additional care needs to be taken to make sure that the number of outliers removed from these 4 labels does not make this dataset even more unbalanced. In the end, applying a 3 x (IQR) limit for outlier detection was safe to use as it resulted in no outliers from these 4 labels.

Chapter 4: Related Work

The Speech Accent Recognition dataset also appeared in the paper 'A Comparison of Classifiers in Performing Speaker Accent Recognition Using MFCCs' by authors and Scientific Research Publishing Inc[2]. The paper used this dataset and converted its multi-class classification task to binary classification. The 5 non-US accents were combined into a single group to make the dataset balanced. This paper aimed to classify MFCC as either US or non-US accent and compare the performance of models used on the dataset.

The paper describes the process of how audio signals are digitally processed to produce the MFCC features which can be used by machine learning models to make extrapolations. Then these MFCCs are used to train different models including LDA, QDA, SVM with linear and RBF kernel, and KNN. The paper also describes in brief how these classification techniques are performed.

Then the performance of these classification techniques was compared and it was observed that KNN provided the highest accuracy for this binary classification problem followed by SVM(RBF), QDA, SVM(PLY), and LDA. The models were also trained and evaluated with a different number of MFCC coefficients including 12, 19, 26, 33, and 39. Training models with higher number of MFCC coefficients improved the model accuracy. The models were also evaluated by their execution time. Here also KNN did better by having the lowest execution time followed by QDA, LDA, SVM(PLY), and SVM(RBF). Both SVM methods took much more time compared to LDA/QDA and KNN.

In the end, the author proposed using more number of MFCC than the first 12, but also suggested using some dimension reduction techniques to tackle the problems caused by high dimensionality. It was also suggested to use different summary statistics such as std. deviation, Gaussian mixtures to summarize a matrix of MFCC coefficients instead of using the mean.

Chapter 5: Feature Extraction

The feature extraction techniques are used to select the features that best represent the underlying correlation in the dataset. It helps the model eliminate irrelevant features and as a result, improves the overall accuracy. It also reduces model complexity which helps if the dataset is very large, however it may impact the model accuracy. In this case, the dataset was relatively smaller so reducing complexity was not the goal unless it improves the model accuracy.

A standard practice with using MFCC for speech application involves taking only the first 12 features from the total of 40 MFCC features as they are most relevant for making interpretations. So taking this into the account, this dataset probably has already applied one type of feature selection by taking only the first 12 MFCC features out of the total of 40. On top of that, I applied the below 2 feature selection methods to select the best of 12 features based on CV accuracy.

1) Feature Importance From Coefficients
2) Selecting features with Sequential Feature Selection

5.1 Feature Importance From Coefficients

The process involves applying RidgeClassifierCV to the input features to get an idea of their importance. The result is one normalized coefficient for each scaled feature of input data. The feature with the highest value is considered the most important and the feature with the lowest value is the least important. A good practice is to further normalize the coefficient by their standard deviation if the features have a similar range as in this case. Because the higher variation has the potential for more predictive ability. At last, We can define a threshold for importance and eliminate the features which fall below it. The feature importance for each language is plotted in the below figure.

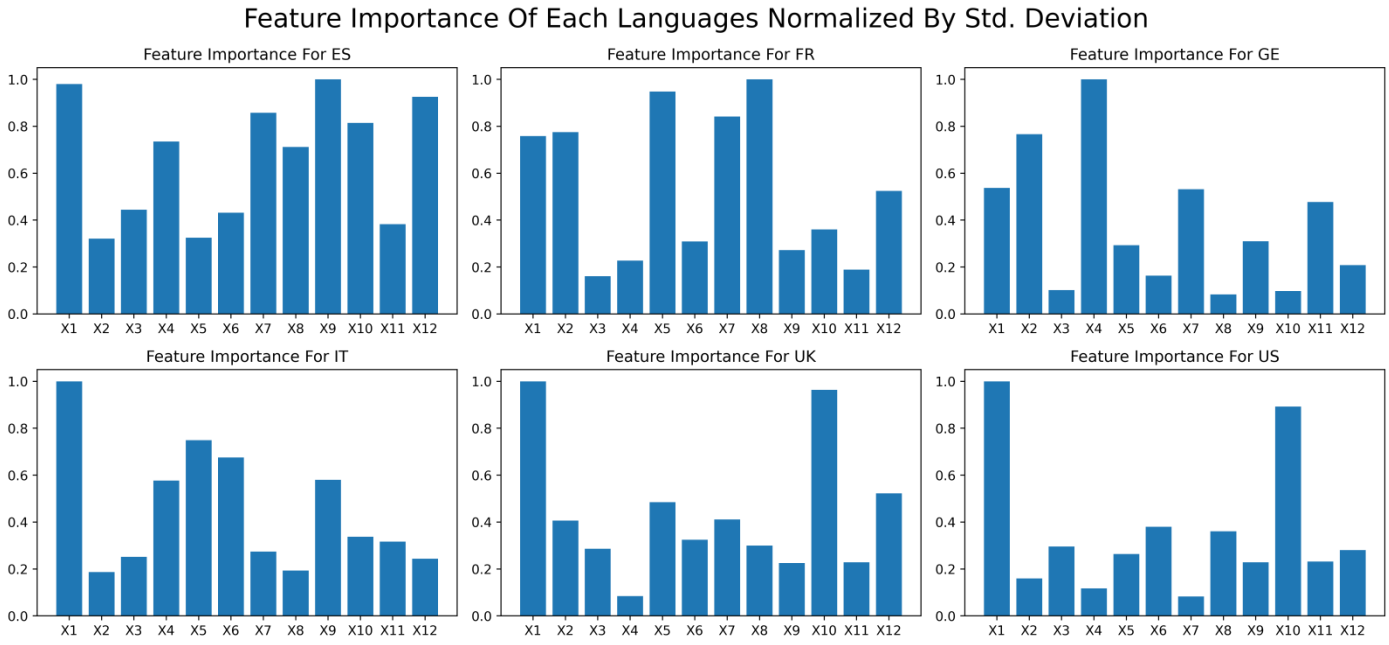


Fig. 6 Feature Importance Of Each Languages Normalized By Std. Deviation

The above plot indicates that some features are more important than others for given languages. For example, the US accent is much more dependent on features X1 and X10. So If we need to identify a particular language, we can eliminate the feature with importance below a threshold. But our final goal is

not limited to a binary prediction for a particular language. So I combined the feature importance of all languages and plotted them as shown below.

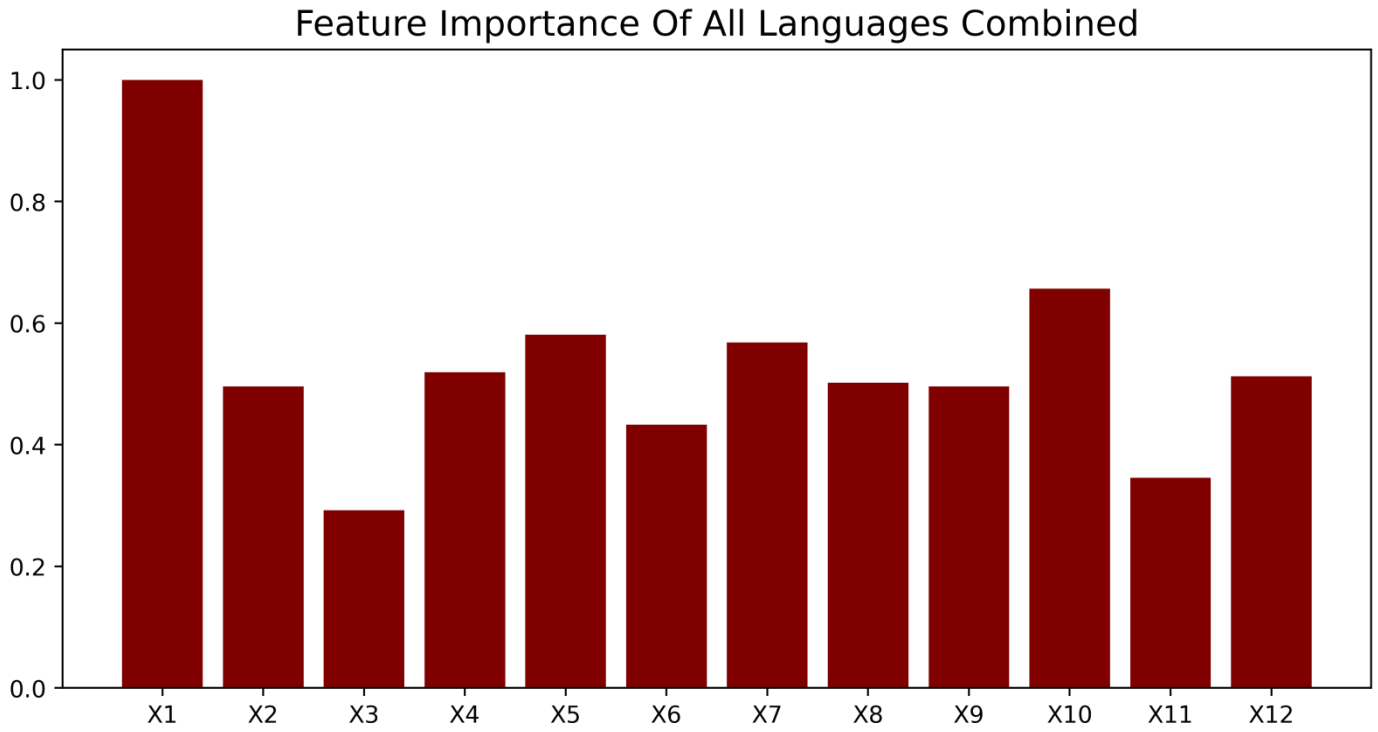


Fig. 7 Feature Importance Of All Languages Combined

From the above figure, I noticed that most of the features except X1 had nearly identical importance. So it was not possible to eliminate any feature without degrading model accuracy. Although one approach I could have taken with the results in Fig. 7 is to train 6 different models for each language with only the most important features of that language. Then use the one-vs-rest method on all 6 predictions and pick the prediction with the highest confidence value as the final prediction. In summary, no feature could be eliminated without reducing the model accuracy with this method.

5.2 Selecting Features With Sequential Feature Selection

In Sequential Feature Selection we start with a low number of features, and at each iteration, we choose the best new feature to add to our selected features based on the CV score. I implemented this method SequentialFeatureSelector[4] in the forward direction for number of features ranging from n=1 to n=12. The CV result of all models indicated that the accuracy decreased significantly with a reduction in the number of features. This is also consistent with the results of the feature importance method discussed in the previous section. So, it was best to continue training the models with all the features. The results of CV for each model are discussed further in the section Fine-tune your models & Feature Set.

Chapter 6: Model Development

6.1 Adding more training data

One of the main obstacles with this dataset was that it was too limited to create a multi-class classifier. Not only the dataset was small in size, but it was further worsened by the class imbalance between the US accent and the remaining accent. Out of all the 329 samples, about 165 belong to the US accent, 45 to the UK accent, and the remaining were divided equally amongst the other accents. This leaves only 30 samples each for ES, IT, GE, and FR accents. This is a very small number of samples to train a model and make reliable predictions on the test samples.

So to increase the number of samples and balance the dataset, I decided to derive additional MFCC coefficients and combine them with the given dataset. To calculate the MFCC coefficients, I used the Speech Accent Archive database hosted by George Mason University[5]. This dataset contains speech samples in the English language spoken by native speakers from different countries. I obtained 5 speech samples for every 6 languages used in our dataset and obtained MFCC for them.

To obtain the MFCC, each speech sample was split into words using split_on_silent from Pydub. A duration of 120 milliseconds and a silence threshold of -20 dB provided optimal splits. After that, MFCC was calculated on each word using python_speech_features. It provides one set of MFCC for every 25 milliseconds window in a word. So a half-second word would result in 20 sets of MFCC. To have just one set of MFCC for each word, all such sets produced for a word are summarized into just one set by simply taking a mean. The algorithm used for MFCC calculation and the obtained results is shown below.

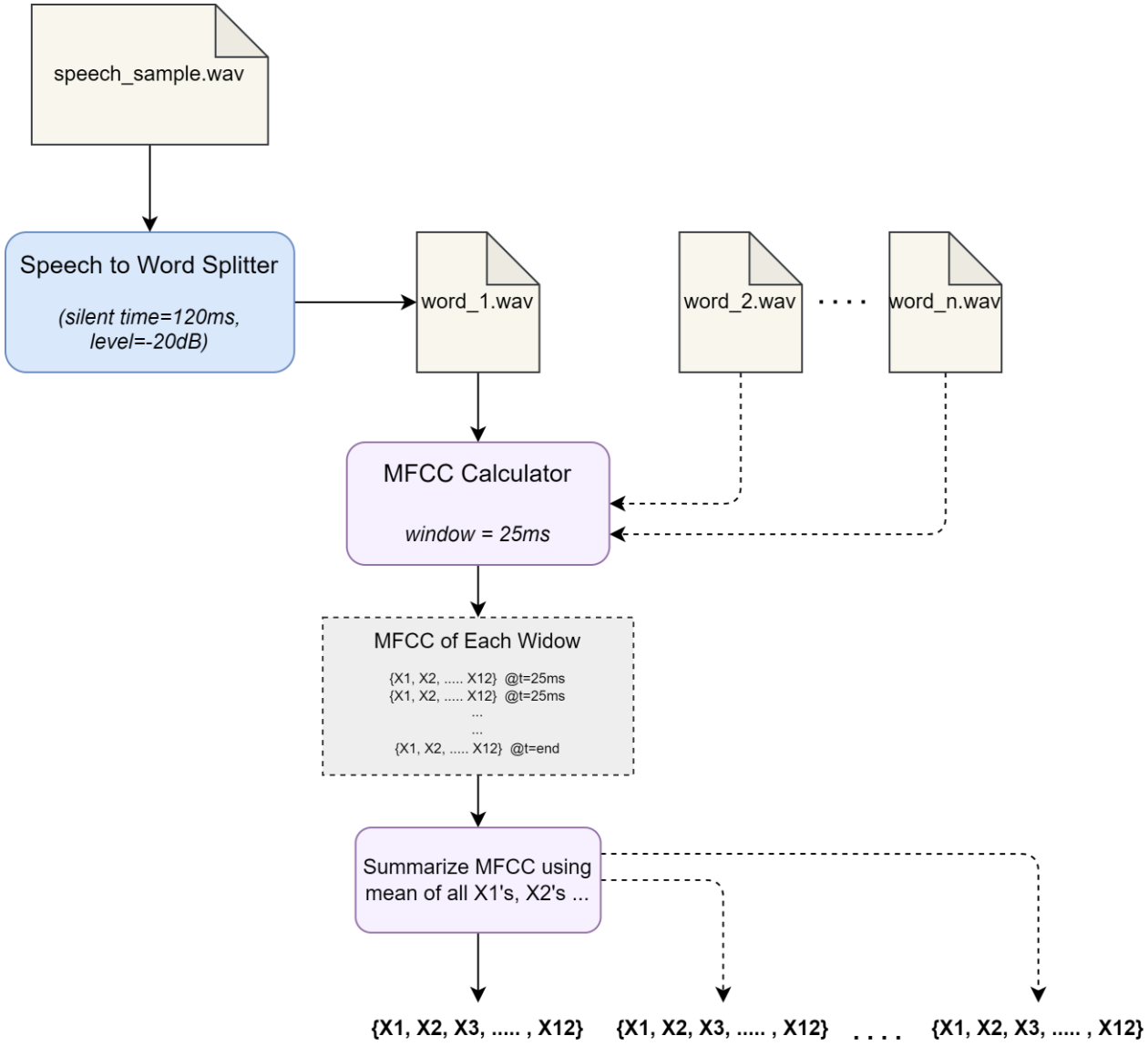


Fig. 8 MFCC Calculation Program Used on Speech Accent Archive Samples

language	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12
FR	-14.2673	-14.4879	38.16259	18.62273	11.90269	0.327577	-7.08501	-13.4937	11.08816	-17.8598	1.526808	-2.44517
FR	2.389558	2.507619	25.2594	2.924945	2.404405	-19.8418	-5.8585	-5.10716	19.94625	-12.499	-8.22522	-16.2975
FR	-1.66699	3.473245	28.67218	9.633882	13.32516	-5.58349	1.315247	-7.67748	11.01356	-22.0287	-2.3527	-10.2143
92 more rows												
GE	-2.50674	-16.6281	18.0225	8.570879	16.60086	-25.1058	-2.43225	-13.2839	-10.2089	-9.84835	-2.80762	-9.24867
GE	14.23794	-12.9896	4.918855	-13.1467	-1.79861	-23.5919	16.34908	-3.81945	-0.90544	-16.2034	-2.21726	-10.1873
GE	16.33078	12.6676	12.2601	-0.09608	9.079021	-18.0753	3.209196	-4.44365	0.516718	-15.1874	-7.75828	-10.2977
143 more rows												
IT	5.509652	-9.43521	34.63177	-0.05311	-19.4825	-30.1276	-13.7957	-14.0393	-13.3834	-17.773	1.506483	7.772739
IT	11.87571	7.053055	18.92694	-19.5418	-28.4551	-8.59966	-7.40479	-3.73764	-6.62799	-11.3113	-5.69065	4.557217
IT	-8.77398	-15.0009	2.771203	-16.8404	-9.71207	-20.9264	-12.6065	2.04339	11.04349	12.67039	9.774488	0.875725
126 more rows												
ES	0.444861	-4.38399	18.41019	8.592189	26.14567	-19.8385	7.977261	-10.6079	0.414503	-9.04174	-3.66851	-5.08311
ES	-0.41666	-2.09701	17.03558	-5.75433	15.05713	-27.0921	15.8511	-2.719	4.425694	-18.7313	-0.31931	2.743598
ES	11.5706	0.494651	4.134005	2.778257	30.11379	-6.73362	-7.80002	-29.5127	3.658062	-5.18086	2.252861	1.537174
146 more rows												
UK	17.04142	1.710063	8.685076	8.923113	9.641099	-24.8862	-5.24766	-1.33891	14.80012	-8.30551	-2.32236	-21.7989
UK	0.023779	-17.9368	29.0713	30.7793	14.86351	-25.9009	-2.58079	8.523504	4.711282	-11.7288	-0.2523	-18.2518
UK	8.785875	-25.4261	-1.32403	-15.51	-13.6613	-14.3241	39.12128	18.55669	1.455805	-15.8796	2.734224	-0.43827
60 more rows												
US	15.56134	-2.38933	-2.12648	-4.77691	-5.21903	-31.471	2.970871	-10.3861	-2.05636	-12.9394	6.155268	-10.9086
US	15.9932	-5.33528	0.382194	-5.85121	7.280806	-24.0341	6.937954	-10.1797	-0.29005	-5.4276	5.967549	-9.00393
US	11.02083	-6.23447	15.61453	16.1781	7.102891	-29.7583	-0.22125	-26.054	2.002052	7.591602	-3.04109	-20.1389
74 more rows												

Table 3. Resulting MFCC

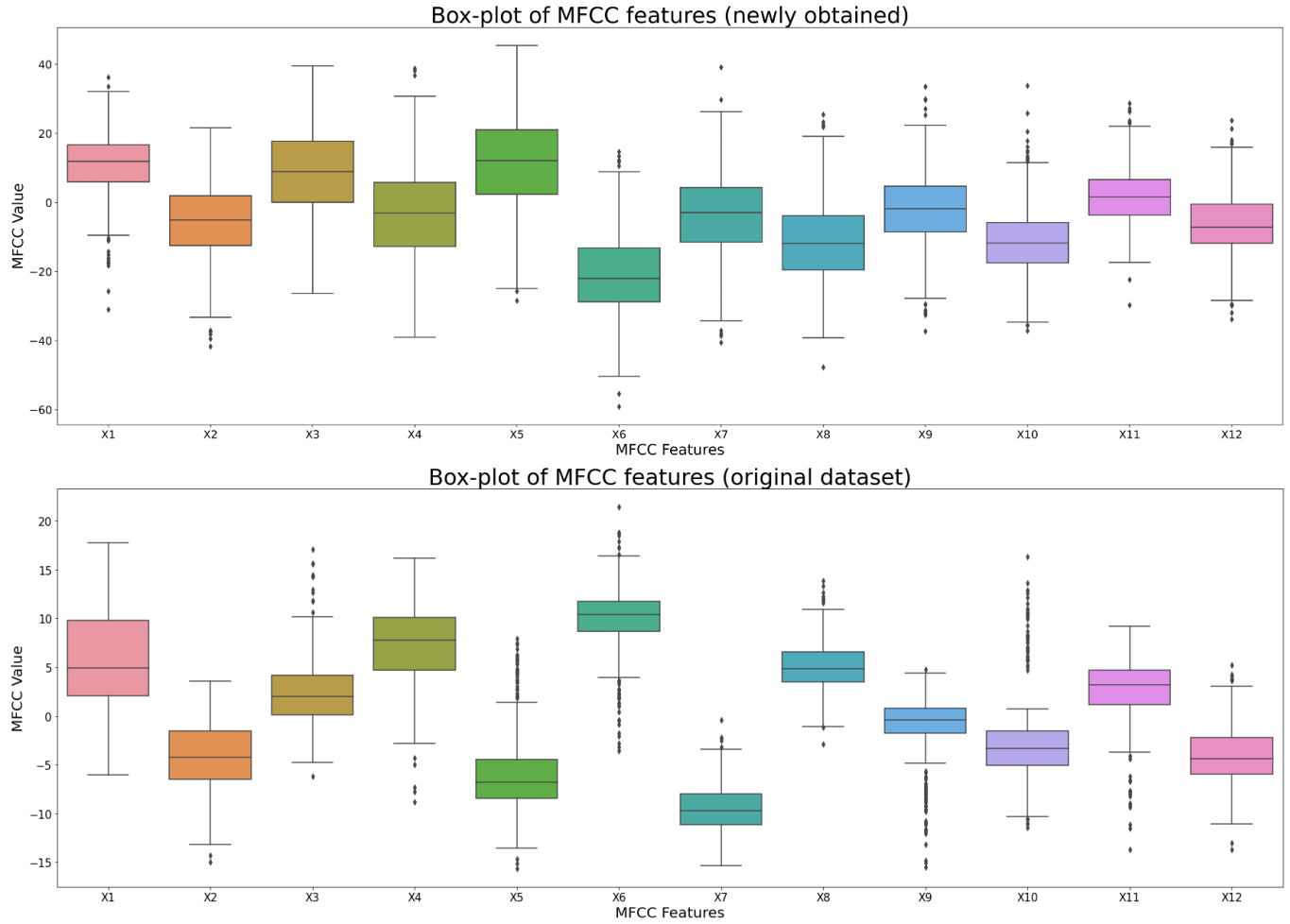


Fig. 9 Comparing New Obtained MFCC with Original Dataset

The newly obtained MFCC were reliable as I was able to make a reliable prediction with models developed on it. However, If you observe the distribution of new MFCC with the MFCC in the original dataset, you can see that the distribution of X1 to X12 is different. For example, X7 is centered around 0.0 in the original dataset but in the new dataset, it is centered around -8.0. Since the distributions were different, it was not possible to merge these two datasets even though the new MFCC samples were correct. A possible reason behind the difference in distribution could be due to the different parameters used in `python_speech_features.mfcc()` such as nftt, nfilt, preemph, and more. The original dataset

description on UCI Machine Learning Repository does not list what parameters were used to obtain MFCC. So I could not reproduce the same results even after trying several combinations. Although the newly obtain MFCC were still good to make predictions.

6.2 K-fold Cross-Validation

To split the dataset into training and testing samples I used the K-fold cross-validation method. A normal K-fold method is not suitable when a dataset is unbalanced because it does not split the samples of all classes in equal proportions. So I used a stratified version of the K-fold method using *StratifiedKFold* from scikit-learn. The dataset was split with K=5 which gives 5 different folds. Of all available folds, 4 folds were used for training and the remaining 1 fold was used for testing the model.

To obtain the overall accuracy, 5 different training and test sets were produced by selecting one unique fold as a test set each time. Each set was then used to train and test a model. In the end, the average of all 5 accuracies was considered the overall accuracy for that model.

6.3 Classification Techniques & Model Development Process

I applied the below classification techniques to this dataset.

1. K-Nearest Neighbor(KNN) Classifier
2. Logistic Regression (OVR)
3. Support Vector Machines
4. Decision Trees
5. Random Forest

The flow of model development is shown in the below diagram. After removing outliers and eliminating irrelevant features, a baseline model is run to serve as a reference for evaluating performance. After that, the model is cross-validated to tune the hyperparameters based on the accuracy score. Then the feature set is adjusted by running iterations of forwarding Subset Selection and PCA for all N features to obtain the best set of features for model development. In the end, the best hyperparameters and feature sets are used to develop the final model and its performance metrics are compared with the baseline model. Each step is discussed in further detail in the upcoming sections of this report.

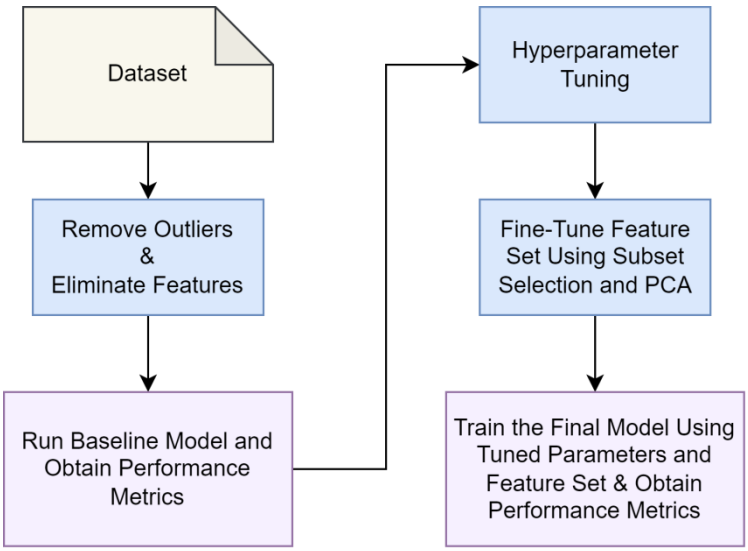


Fig 10. Model Development Process

Out of all 5 methods, Logistic Regression and SVM are binary classifiers. To make them work with the multi-class classification I used the One-Vs-Rest method. The training and testing error of baseline models is shown in the below table. From the training accuracy, it is clear that the decision tree and random forest are in the overfitting region.

Classification Method	Training Accuracy (average)	Training Accuracy (max)	Testing Accuracy (average)	Testing Accuracy (max)	Rank
KNN	0.89	0.89	0.67	0.87	2
Logistic Regression	0.81	0.81	0.63	0.74	3
SVM	0.70	0.74	0.61	0.66	4
Decision Tree	1.0	1.0	0.57	0.71	5
Random Forest	1.0	1.0	0.68	0.76	1

Table 4. Performance of Baseline Models

The approximate measurements of time required for training a single model of each classification method are shown in the below table. The Random Forest was the slowest as it requires training multiple models due to the ensembling technique. Logistic Regression also took more time to train. The remaining KNN, Decision tree, and SVM took the least and about the same amount of time for training.

Classification Method	Training Time (Approx.)	Rank
KNN	7 ms	2
Logistic Regression	60 ms	4
SVM	8 ms	3
Decision Tree	6 ms	1
Random Forest	200 ms	5

Table 5. Training Time For Classification Methods

Chapter 7. Fine-tune Model & Feature Set

After analyzing the performance of baseline models, I proceeded to tune the hyperparameters of each model using CV to find the best fit. After that, I applied some of the common optimization techniques to the dataset to select the best method.

7.1 Fine-tune Model

Checking all possible combinations of available hyperparameters can be computationally very expensive. As the number of hyperparameters increases, the number of different models required to train increases exponentially which takes a lot of time. Therefore we pick only the hyperparameters which are more important and have a significant impact on the model performance. The models were fined tuned by applying all combinations of hyperparameters selected for tuning as well as using GridSearchCV and cross-validation[3].

For KNN model, the hyperparameter K decides the number of nearest neighbors to include for the majority selection. A very low K can lead to overfitting whereas a high value of K would make the model underfitting to the dataset. So, to pick the best K, I cross-validated different models for K values ranging from 1 to 15. The final K was selected based on the misclassification rate. In most cases, K=1 provided the lowest misclassification rate. The accuracy of KNN model for each K is shown below.

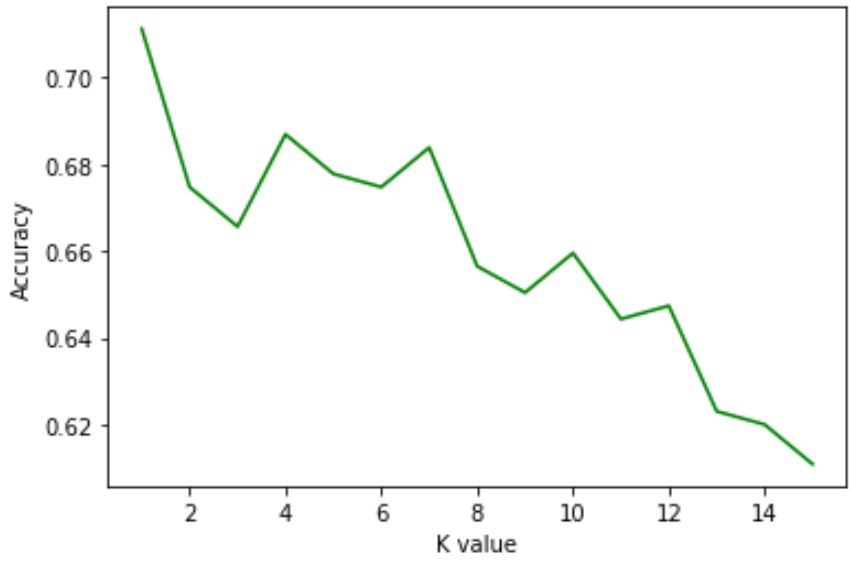


Fig. 11 Selecting The Best K for KNN

Logistic Regression does not have very critical hyperparameters. However, I selected solver, penalty, and C for tuning. The solvers ('newton-cg', 'lbfgs', 'liblinear'), penalty ('l1', 'l2'), and C(100, 10, 1.0, 0.1, 0.01) were selected as possible parameter values. To pick the best set, each combination were used to train a unique model and then cross-validated. The combination with the least misclassification rate was selected as the best hyperparameters. In this case, the 'newton-cg' solver with L2 penalty and C=1.0 was selected as the best set. The plot of C vs accuracy for the final model is shown below.

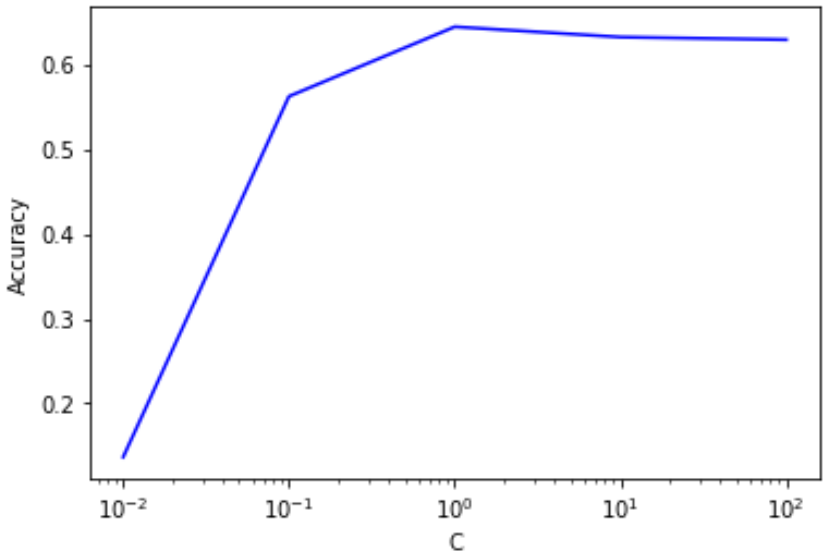


Fig. 12 Logistic Regression C(Inverse of regularization strength) vs Accuracy

SVM has a large number of parameters for tuning. The most important hyperparameter is the kernel selection as it is the main parameter in creating the decision boundary. I selected linear, poly, and RBF kernels as a set of available kernels as they are the most commonly used kernels. Another important parameter is the cost function 'C'. It decides the budget of slack variables which determines the level of flexibility provided to the decision boundary. For this dataset, RBF kernel with C=10 provided the best performance. The plot of accuracy against the cost function is shown below.

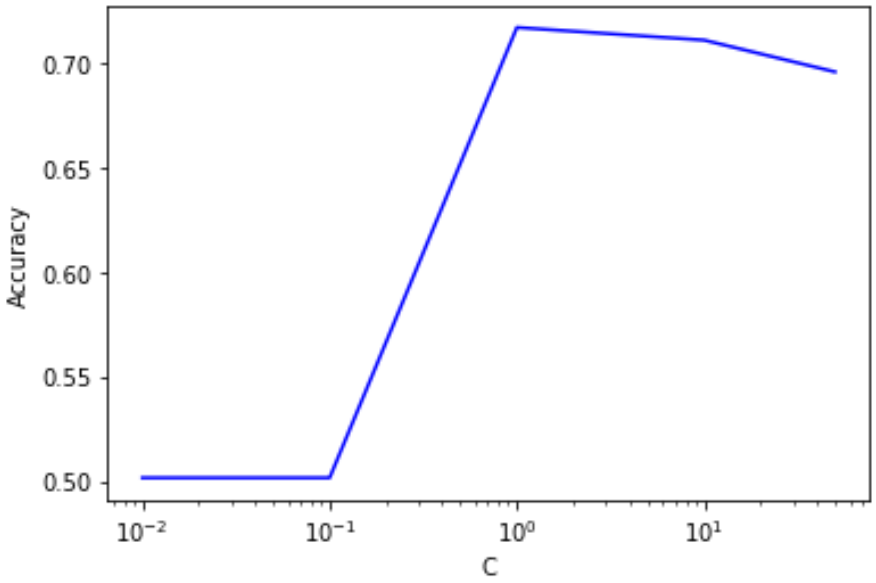


Fig. 13 SVM Cost Function vs Accuracy

In a Decision Tree, the max_depth parameter decides how deep the tree can be. A large depth allows the tree to have more splits and capture more information. However, it can also make the model overfitting to the training data. The min_samples_split parameter imposes the lower limit on the number of samples required to be present in each region after the split. A lower number of samples on split can also lead to overfitting. The cross-validation results with max_depth ranging from 1 to 30 and min_sample_split from 1 to 20 gave the depth 7 and split 5 as the best parameters. The below plot indicates how the accuracy of the decision tree changes with tree depth. You can observe that after a depth of 7, the additional increase in depth does not provide a significant increase in accuracy.

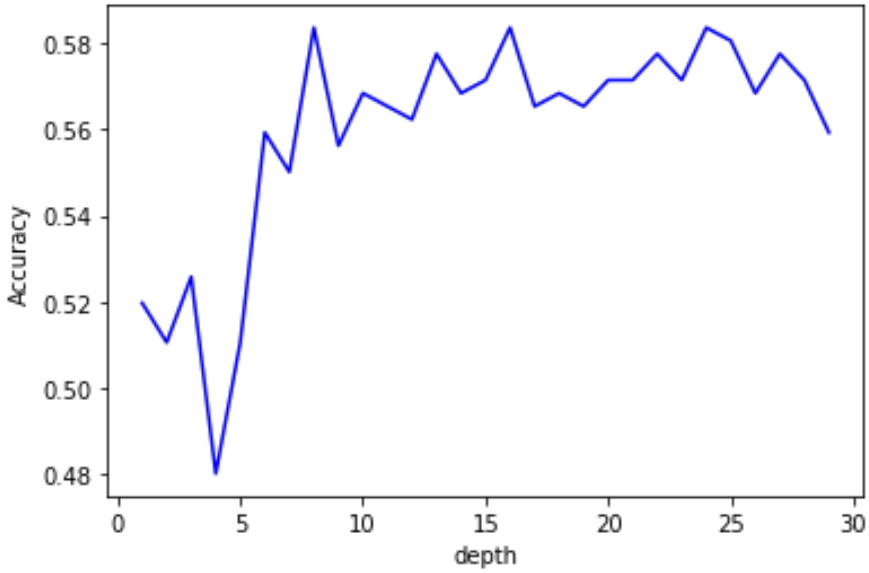


Fig. 14 Decision Tree Depth vs Accuracy

For Random Forest, the most important parameter is the number of random features to sample at each split point. Increasing max_features generally improves the performance as at each node now we have a higher number of options to be selected. The n_estimator parameter decides the number of trees to develop before taking the majority selection. However, a high number of n_estimator may be computationally expensive as it increases the number of different models required to be developed. I used log2 and sqrt for max_features and n_estimator between 10 to 1000. The best combination of max_features and n_estimator was found to be sqrt and 100 as shown in the below diagram.

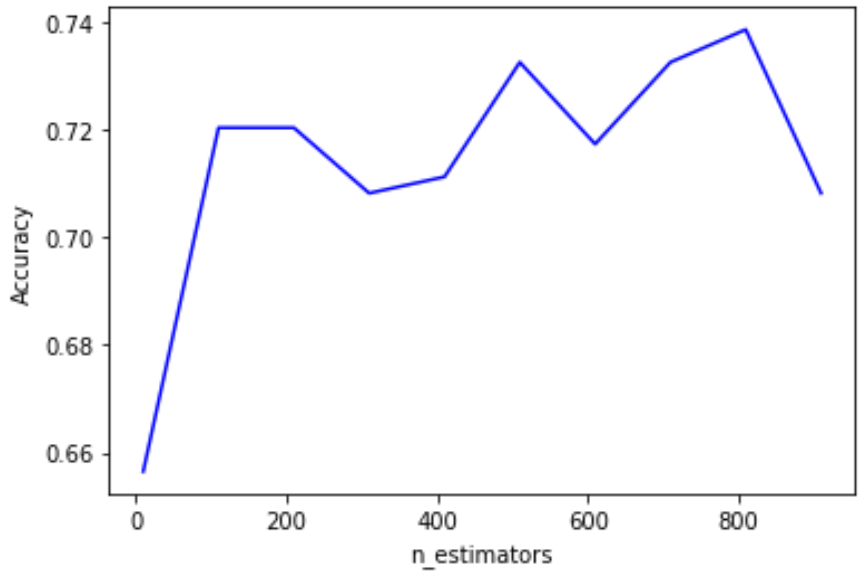


Fig. 15 Random Forest Number Of Trees vs Accuracy

7.2 Optimize Feature Set

Below two techniques were used to optimize the feature set. Both techniques aim to optimally reduce the number of features while maintaining or improving the overall accuracy of the model. These techniques help reduce the complexity of the model and also help in avoiding overfitting.

1) Forward Subset Selection
2) PCA

The forward subset selection method is an iterative process through which we select the best features using cross-validation. This helps eliminate the features that do not contribute much or are irrelevant for prediction. I applied this technique to select the best set of MFCC coefficients after the hyperparameter tuning of each model. In all cases, it was observed that the models trained with all the features performed the best as shown in the below plots.

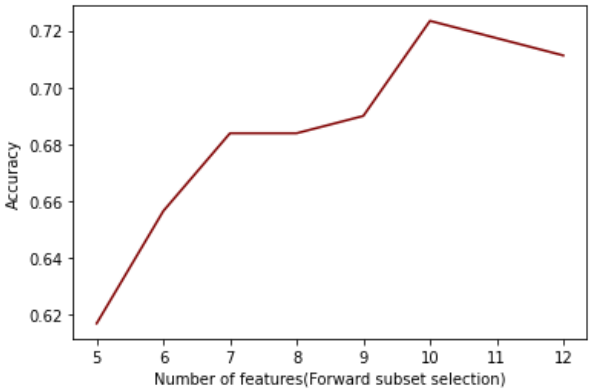


Fig. 16(a) KNN Fowrard Subset Selection

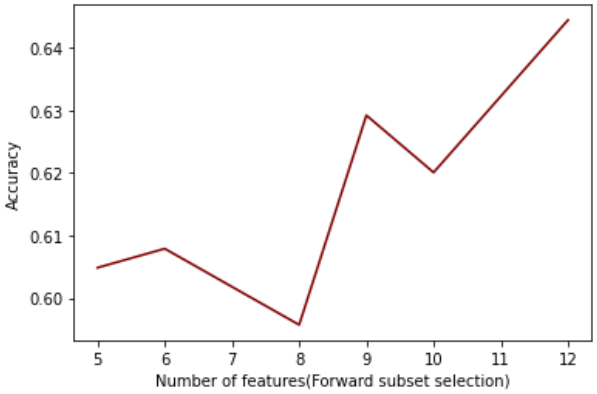


Fig. 16(b) Logistic-Regression Fowrard Subset

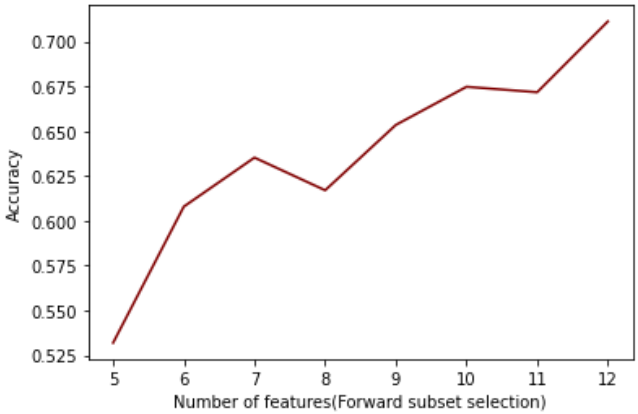


Fig. 16(c) SVM Forward Subset Selection

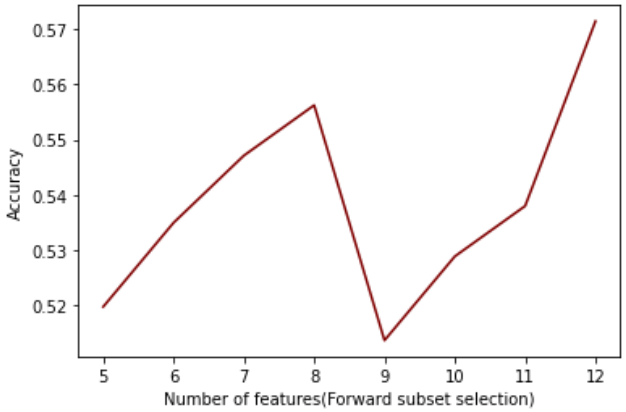


Fig. 16(d) Decision Tree Forward Subset

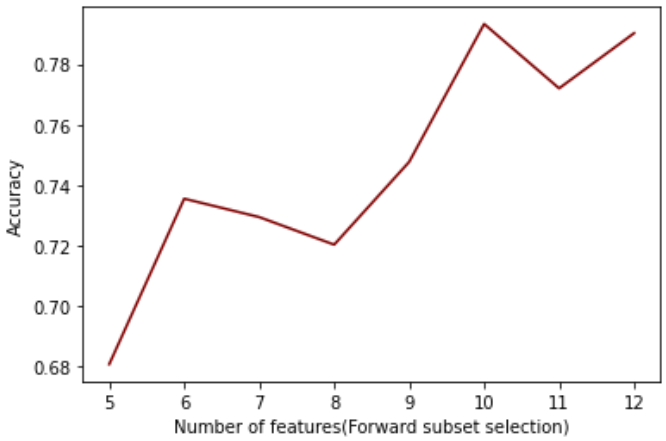


Fig. 16(e) Random Forest Forward Subset Selection

The Principal Component Analysis is used to reduce the dimensionality of the dataset by forming principal components that are linear combinations of all the features from the dataset. These principal components capture features and the regions where data vary the most. Each principal component is uncorrelated and

hence orthogonal to each other. The number of features in the dataset was not significantly high. So the PCA did not perform better on this dataset. The below plots show the effect of reducing features using PCA on the overall model accuracy.

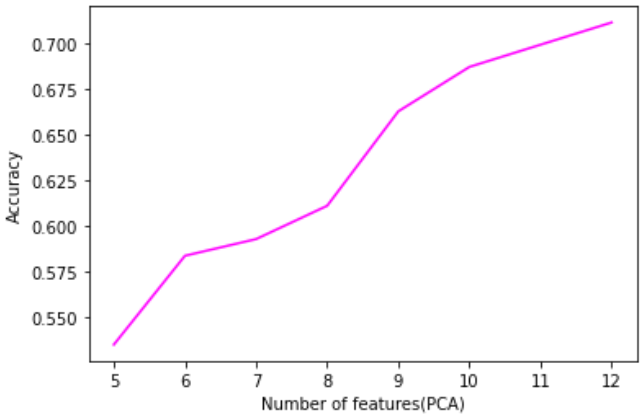


Fig. 17(a) KNN With PCA

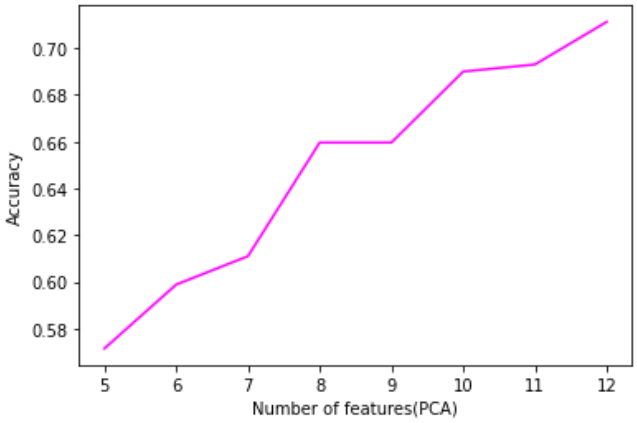


Fig. 17(b) Logistic Regression With PCA

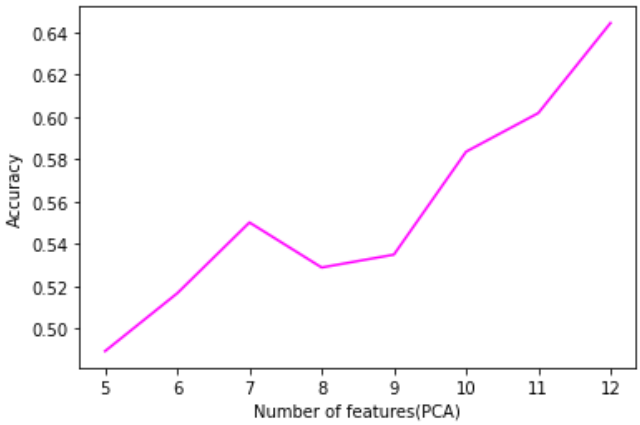


Fig. 17(c) SVM With PCA

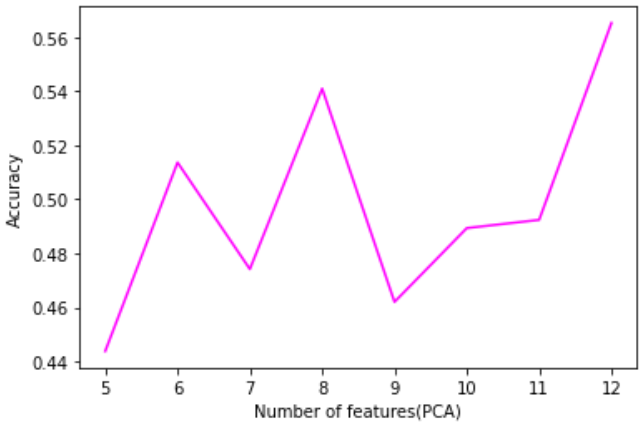


Fig. 17(d) Decision Tree With PCA

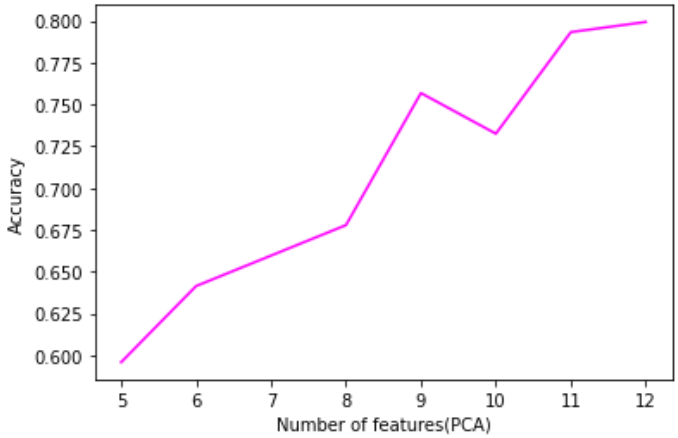


Fig. 17(e) Random Forest With PCA

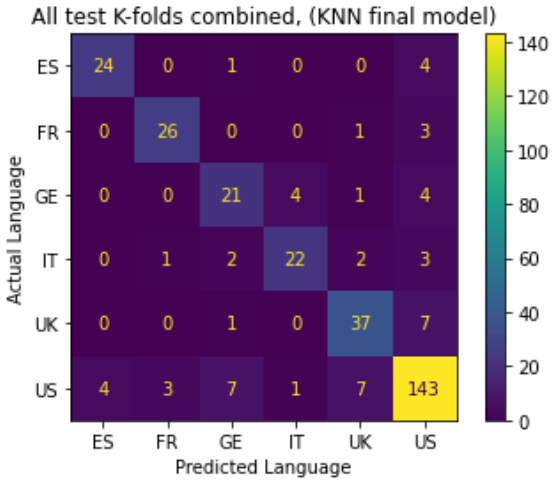
Chapter 8: Performance

The performance of each model was evaluated using K-fold cross-validation with K=5. I used accuracy, precision, and recall as the performance metrics to evaluate each model. Each metric was obtained by taking the average of all k-folds test results. The average and maximum accuracy of each model are shown in the below table. One important observation from the table is that the fine-tuning of the model and feature set provided around 10% to 20% of absolute improvement in model accuracy.

Classification Method	Training Accuracy (average)	Training Accuracy (max)	Testing Accuracy (average)	Testing Accuracy (max)	Improvement Over Baseline	Rank
KNN	1.0	1.0	0.83	0.87	16%	1
Logistic Regression	0.77	0.78	0.73	0.77	10%	4
SVM	0.97	0.98	0.83	0.93	21%	2
Decision Tree	0.85	0.90	0.67	0.74	10%	5
Random Forest	1.0	1.0	0.79	0.90	11%	3

Table 6. Final Model Performance

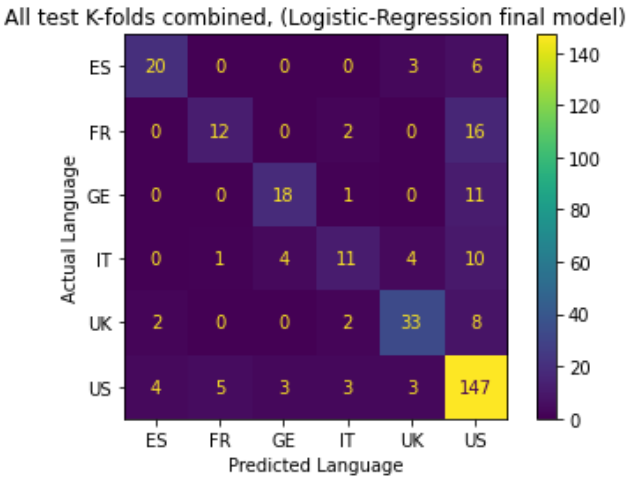
KNN provided the best average accuracy on the test set. However, the training accuracy(1.0) of KNN indicates that it was still overfitting to the training data even after taking the optimal K value. The confusion matrix and performance metrics are shown below. The model has fairly consistent precision and recall for all languages. So it can be said that the final KNN model was able to identify all the languages properly.



	Language	Accuracy	Precision	Recall	F-score
0	ES	0.827586	0.857143	0.827586	0.842105
1	FR	0.866667	0.866667	0.866667	0.866667
2	GE	0.700000	0.656250	0.700000	0.677419
3	IT	0.733333	0.814815	0.733333	0.771930
4	UK	0.822222	0.770833	0.822222	0.795699
5	US	0.866667	0.871951	0.866667	0.869301
6	Average(weighted)	0.829787	0.831455	0.829787	0.830221

Fig. 18 Performance Metrics For KNN

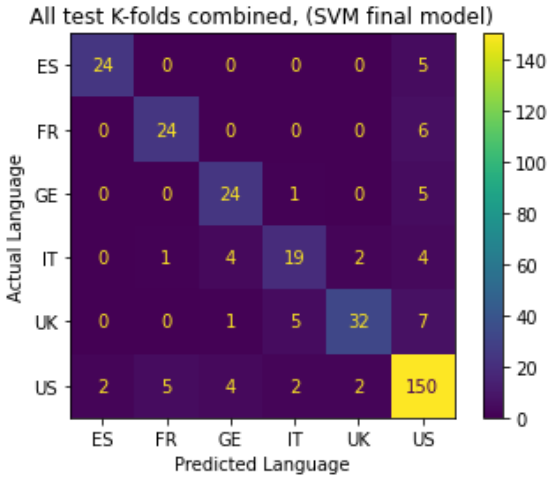
The logistic regression(OVR) has relatively lower accuracy. It also took more time to train. So, overall we can say that it did not perform well. The correlation matrix and performance metrics are shown below. The model did not perform well while identifying FR, GE, and IT accents as indicated by lower accuracy, precision, and recall. Many of the predictions are falsely identified as the US accent probably due to the unbalanced nature of the dataset.



	Language	Accuracy	Precision	Recall	F-score
0	ES	0.689655	0.769231	0.689655	0.727273
1	FR	0.400000	0.666667	0.400000	0.500000
2	GE	0.600000	0.720000	0.600000	0.654545
3	IT	0.366667	0.578947	0.366667	0.448980
4	UK	0.733333	0.767442	0.733333	0.750000
5	US	0.890909	0.742424	0.890909	0.809917
6	Average(weighted)	0.732523	0.724350	0.732523	0.719097

Fig. 19 Performance Metric For Logistic Regression

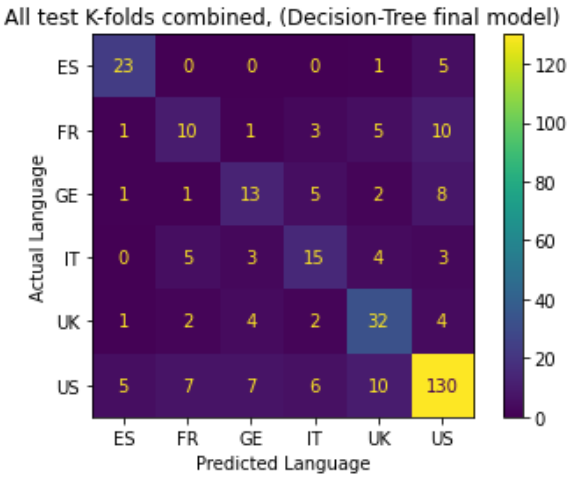
SVM performance was very much similar to KNN. Both models have the same accuracy as well as similar precision and recall. SVM was also able to identify all the languages confidently. It also took the least time to train the model.



	Language	Accuracy	Precision	Recall	F-score
0	ES	0.827586	0.923077	0.827586	0.872727
1	FR	0.800000	0.800000	0.800000	0.800000
2	GE	0.800000	0.727273	0.800000	0.761905
3	IT	0.633333	0.703704	0.633333	0.666667
4	UK	0.711111	0.888889	0.711111	0.790123
5	US	0.909091	0.847458	0.909091	0.877193
6	Average(weighted)	0.829787	0.831395	0.829787	0.828142

Fig. 20 Performance Metric For SVM

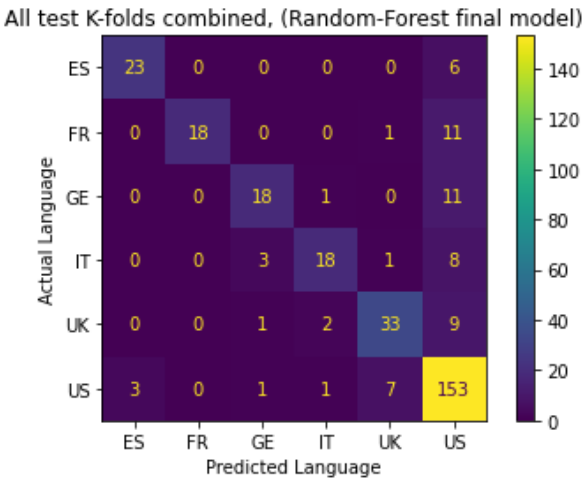
The decision tree performed the worst out of all models. It has very low precision, recall, and accuracy. It could not predict FR, GE, and IT accents similar to the logistic regression. However, the decision tree did not have any bias towards the US accent like logistic regression.



	Language	Accuracy	Precision	Recall	F-score
0	ES	0.793103	0.741935	0.793103	0.766667
1	FR	0.333333	0.400000	0.333333	0.363636
2	GE	0.433333	0.464286	0.433333	0.448276
3	IT	0.500000	0.483871	0.500000	0.491803
4	UK	0.711111	0.592593	0.711111	0.646465
5	US	0.787879	0.812500	0.787879	0.800000
6	Average(weighted)	0.677812	0.676869	0.677812	0.676096

Fig. 20 Performance Metric For Decision Tree

The random forest model performed average compared to SVM and KNN. It also has overfitting as indicated by training accuracy of 1.0. It was able to predict the US accent with high recall. But it also has a bias towards the US accent similar to the logistic regression and indicated by low precision. The model was not able to accurately predict FR, GE, and IT accents.The random forest also took the highest time to train due to ensembling. It is almost 10 times higher than logistic regression.



	Language	Accuracy	Precision	Recall	F-score
0	ES	0.793103	0.884615	0.793103	0.836364
1	FR	0.600000	1.000000	0.600000	0.750000
2	GE	0.600000	0.782609	0.600000	0.679245
3	IT	0.600000	0.818182	0.600000	0.692308
4	UK	0.733333	0.785714	0.733333	0.758621
5	US	0.927273	0.772727	0.927273	0.842975
6	Average(weighted)	0.799392	0.810136	0.799392	0.793708

Fig. 20 Performance Metric For Random Forest

Chapter 9: Overall Discussion Of Results & Conclusions

In this project, I performed the multi-class of detecting the accent from speech. I trained different models and optimized them with cross-validation to achieve better test performance. One major problem with the dataset was the lack of sample points compounded by its unbalanced nature. Due to this, some of the models were not able to handle the test data well. I also tried deriving additional data samples by calculating the MFCC of the audio samples from the Speech Accent Archive[5]. Although new MFCCs were calculated properly, it was not possible to use them with this dataset due to different distributions.

Out of all the methods, KNN followed by SVM provided the highest accuracy. They also took the least time to train. Tree-based methods did not perform well and resulted in overfitting, and they also took relatively more time to train. One important thing to note is that the hyperparameter tuning was critical to these two models. Selecting the kernal for SVM and using different values of K had a significant impact on the model performance. Logistic regression and decision were affected by the unbalanced nature of the dataset. Hyperparameter tuning alone provided around 10% to 20% improvement for all models.

Feature selection and dimension reduction techniques including Forward Subset Selection and PCA did not help in improving the model accuracy. Regularization using ridge also could not eliminate any feature. Reducing the number of features negatively affected the model's performance. So It can be said that all the features were important in predicting the output. The K-fold cross-validation also provided consistent training and testing performance compared to the validation set approach.

In conclusion, some of the trained models were able to predict the accent from MFCC coefficients. Hyperparameter tuning was found to be a critical step in the model development process. The consistency of the K-fold method in cross-validation was also important in tuning the model and obtaining performance metrics. In most cases, the lack of samples in the dataset was the bottleneck in developing better models.

References

[1] Fokoue, E. (2020). UCI Machine Learning Repository [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Speaker+Accent+Recognition Irvine, CA: University of California, School of Information and Computer Science. [Accessed: 28-March-2022]

[2] Zichen Ma, Ernest Fokoué, 'A Comparison of Classifiers in Performing Speaker Accent Recognition Using MFCCs', Scientific Research Publishing Inc, 2014. [Accessed: 12-May-2022].

[3] 'Tune Hyperparameters for Classification Machine Learning Algorithms', Analytics Vidhya. [Online]. Available: https://machinelearningmastery.com/hyperparameters-for-classification-machine-learning-algorithms/ [Accessed: 8-May-2022].

[4] 'Model-based and sequential feature selection', scikit-learn. [Online]. Available: https://scikit-learn.org/stable/auto_examples/feature_selection/plot_select_from_model_diabetes.html#sphx-glr-auto-examples-feature-selection-plot-select-from-model-diabetes-py [Accessed: 10-May-2022]

[5] Steven H. Weinberger, 'The Speech Accent Archive', [Online]. Available: https://accent.gmu.edu/ [Accessed: 2-May-2022]