EE 257 - Machine Learning for Electrical Engineers

ML Project

Instructor: Dr. Biren Sirkeci

Speaker Accent Recognition Data Set (2020)

Pushkar Deodhar (SJSU ID: 015266914)

Partnered With: Shravya Juluru

Data Set Description

Dataset Speaker Accent Recognition is taken from UCI Machine Learning Repository. In this project, we will categorize people based on their accents. Data was collected from around 300 audio recordings of different speakers. The dataset includes accents from speakers in Spain, France, Italy, Germany, the United Kingdom, and the United States. Half of the dataset is made up of data from accents of the people in the United States, while the other half is made up of data from accent data from the other nations indicated. A dataset is a collection of frequency coefficients derived from various audio recordings that have been preprocessed and translated into Mel frequency Cepstral coefficients (MFCC). The collection includes 12 MFCCs of this kind. MFCC is an audio signal representation technique for the short-term power spectrum that is widely utilized in speech recognition systems. The graphic below depicts a dataset with 12 MFCC in columns and data from several nations in rows.

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...
324	US	-0.525273	-3.868338	3.548304	1.496249	3.490753	5.849887	-7.747027	9.738836	-11.754543	7.129909	0.209947	-1.946914
325	US	-2.094001	-1.073113	1.217397	-0.550790	2.666547	7.449942	-6.418064	10.907098	-11.134323	6.728373	2.461446	-0.026113
326	US	2.116909	-4.441482	5.350392	3.675396	2.715876	3.682670	-4.500850	11.798565	-12.031005	7.566142	-0.606010	-2.245129
327	US	0.299616	0.324844	3.299919	2.044040	3.634828	6.693840	-5.676224	12.000518	-11.912901	4.664406	1.197789	-2.230275
328	US	3.214254	-3.135152	1.122691	4.712444	5.926518	6.915566	-5.799727	10.858532	-11.659845	10.605734	0.349482	-5.983281

Fig 1: Table output using head() command

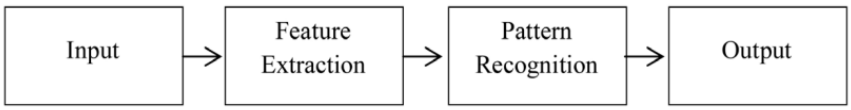

Fig 2: Block diagram of the process

Data Set Visualization

An algorithm using Mel-Frequency Cepstral Coefficients (MFCCs) is proposed to accomplish signal feature extraction for the job of speaker accent detection. The MFCC feature is then used to compare several classifiers. For each signal, the mean vector of the MFCC matrix is utilized as an input vector for pattern recognition. A total of 330 signals are investigated, including 165 from the United States and 165 from other countries. We would only use binary classification to determine whether speakers had a US accent or not. In this section, we shall visualize the data that has been provided to us. As previously indicated, the collection includes accent data from France, Italy, Germany, the United States, the United Kingdom, and Spain. A data collection of single English words read by speakers from six different nations was developed for accent detection and recognition. We look at the input data, which is X1, X2, ..., X12. There are no outliers in the box plot for X1, however, there are outliers in the following plots X2, ...X12. Because the number of data points is equal, the dataset is balanced. The histogram plot shows that data is restricted for X5, X10, X6, X9, and X11.

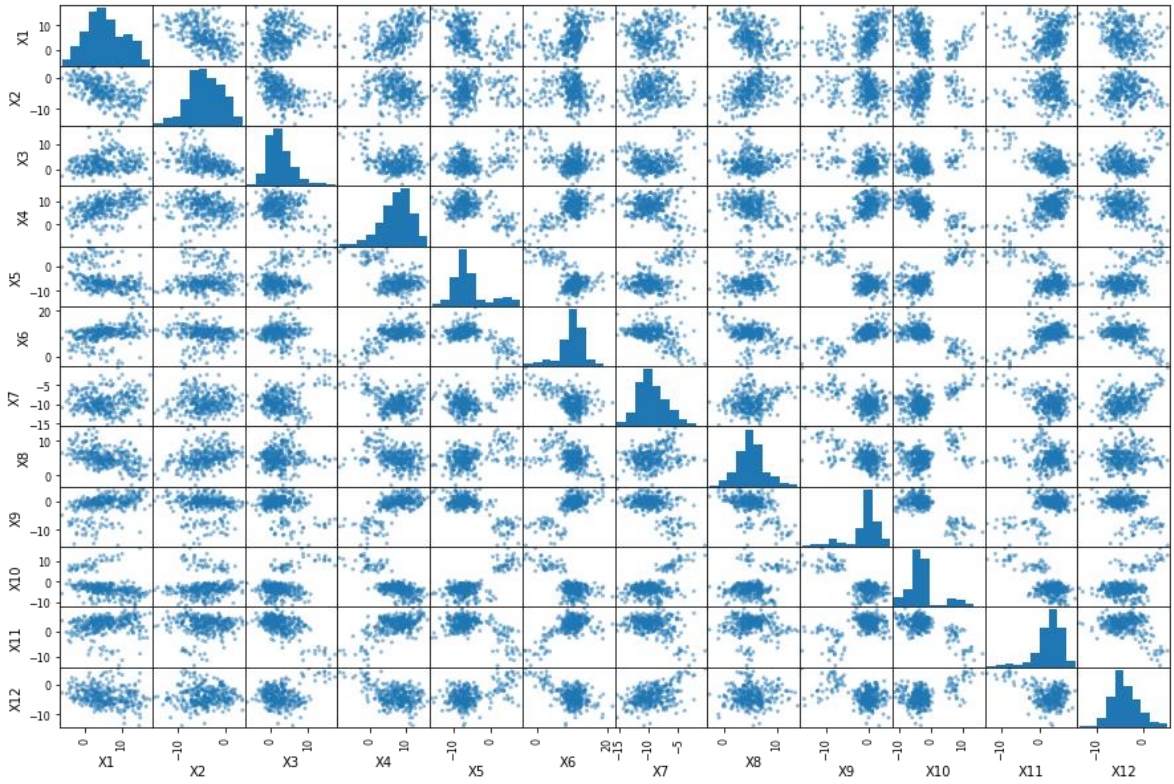

Fig 3: Scatter Plot

The scatter plot above shows the relation between many features. Some characteristics have a non-linear connection, whereas others have a linear relation.

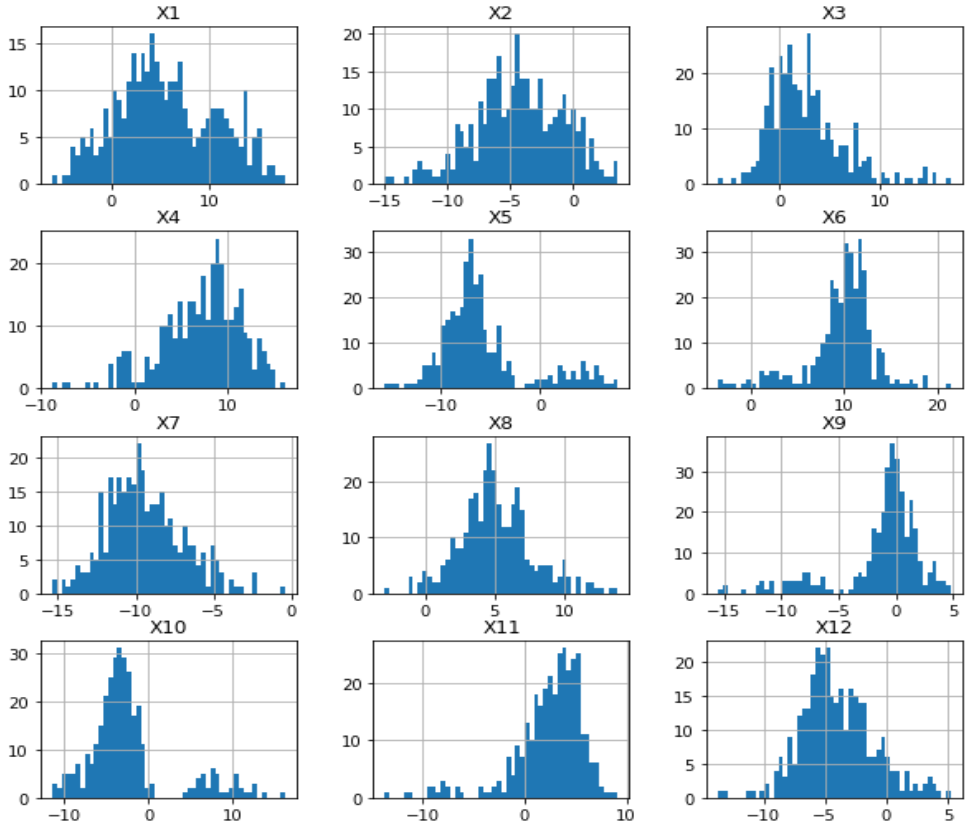

Fig 4: Collective Plot of Histogram of 12 MFCC

To check whether frequency coefficient features are balanced or not, a histogram plot is utilized. For each input MFCC, 12 histograms are shown. A histogram is a graphical representation of data that may be used to estimate probability distributions. We learned more about balanced and unbalanced features after analyzing the histogram plot.

There are several outliers for each box plot. An outlier is a reading that deviates from the norm owing to measurement or experimental mistakes. Outliers can have an impact on the model's performance. In our scenario, removing outliers from the dataset and training the model will result in fewer data points available for the model to train. As a result, the model will not be effectively trained. It will increase the error in the output. This point is explained in the following sections.

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12
X1	1.000000	-0.516210	0.111992	0.468499	-0.418736	0.368058	0.081495	-0.468839	0.392974	-0.358173	0.164058	-0.237836
X2	-0.516210	1.000000	-0.475168	-0.133245	0.091181	-0.110199	0.052197	0.354930	-0.105822	0.081283	-0.006415	0.021244
X3	0.111992	-0.475168	1.000000	-0.304516	0.325788	-0.321455	0.133902	0.037379	-0.304485	0.370584	-0.597373	0.174310
X4	0.468499	-0.133245	-0.304516	1.000000	-0.518275	0.587991	-0.276286	-0.317395	0.672918	-0.739205	0.560560	-0.435478
X5	-0.418736	0.091181	0.325788	-0.518275	1.000000	-0.602486	0.411764	0.442394	-0.765331	0.738060	-0.526548	0.443376
X6	0.368058	-0.110199	-0.321455	0.587991	-0.602486	1.000000	-0.505647	-0.395979	0.714804	-0.710877	0.630567	-0.535893
X7	0.081495	0.052197	0.133902	-0.276286	0.411764	-0.505647	1.000000	0.216448	-0.527454	0.532713	-0.408103	0.449124
X8	-0.468839	0.354930	0.037379	-0.317395	0.442394	-0.395979	0.216448	1.000000	-0.522148	0.297147	-0.033765	0.059785
X9	0.392974	-0.105822	-0.304485	0.672918	-0.765331	0.714804	-0.527454	-0.522148	1.000000	-0.759978	0.497700	-0.407227
X10	-0.358173	0.081283	0.370584	-0.739205	0.738060	-0.710877	0.532713	0.297147	-0.759978	1.000000	-0.673149	0.444235
X11	0.164058	-0.006415	-0.597373	0.560560	-0.526548	0.630567	-0.408103	-0.033765	0.497700	-0.673149	1.000000	-0.387403
X12	-0.237836	0.021244	0.174310	-0.435478	0.443376	-0.535893	0.449124	0.059785	-0.407227	0.444235	-0.387403	1.000000

Fig 5: correlation matrix of features

We can see from figure 5, the correlation matrix that there is a positive, negative, or nearly no relationship between various characteristics. Correlation ranges from -1 to 1. If the correlation is positive, then the variables tend to rise together. If the correlation is negative, then the variables have an inverse relation. Example X9 and X6 have a strong positive correlation, which indicates they are highly connected, which suggests they are likely to increase together. X10 and X9 have a substantial negative correlation, which suggests that if one variable increases, the other will drop. X11 and X2 exhibit practically negligible correlation, which indicates that both variables will raise or decrease regardless of the behavior of another variable.

Data Set Cleaning

The dataset has already been analyzed, and only the most significant 12 MFCC features are included. The boxplot dataset cleaning approach is utilized in this project to determine the number of extreme data points, also known as outliers. We estimated the outliers in each characteristic using the interquartile range (IQR). However, removing outliers from the dataset reduces the model's accuracy. There aren't enough data points to train and test our model as we eliminate outliers. Because the method is complex, we need more datasets to achieve acceptable accuracy, thus we opted not to eliminate outliers. Outliers also have an impact on the mean value of characteristics. We also ran the dropna() method to test whether there are any null values in the dataset, but as we can see from the output, the count value is the same for the number of datasets for each feature, indicating that there are no null values.

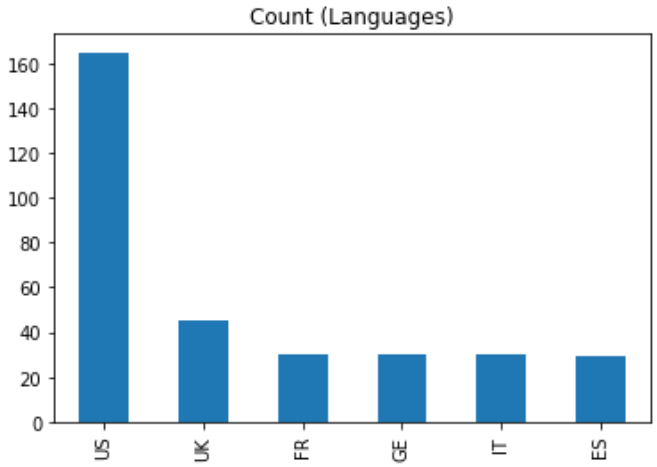

Fig 6: Dataset after cleaning

Related work

A comparison of classifiers in performing speaker accent identification using MFCCs was published in the publication: A Comparison of Classifiers in Performing Speaker Accent Recognition Using MFCCs [3]. One of the critical steps in the machine learning industry is model tuning and feature extraction. The performance and complexity of the model are determined by feature extraction and model tuning. Datasets are becoming increasingly dimensional as artificial intelligence and machine learning technology are progressing. When there is a huge dataset, machine learning models perform better; yet, the growth in the dataset poses difficulty in feature extraction. As a result, ML developers must devise straightforward techniques to perform feature extraction.

The paper describes the categorization of the accent of the person using Mel-frequency Cepstral Coefficients (MFCC), which is a prominent signal processing method. The study offers a method for classifying accents into non-USA and USA accents using binary classification. The study describes a classification strategy employing KNN, SVM, and discriminant analysis. The MFCC transforms signals from the time domain to the frequency domain. The author also used a high pass filter on the input signals in the study, where x[n] is the raw data and s[n] is the signal after the filtering.

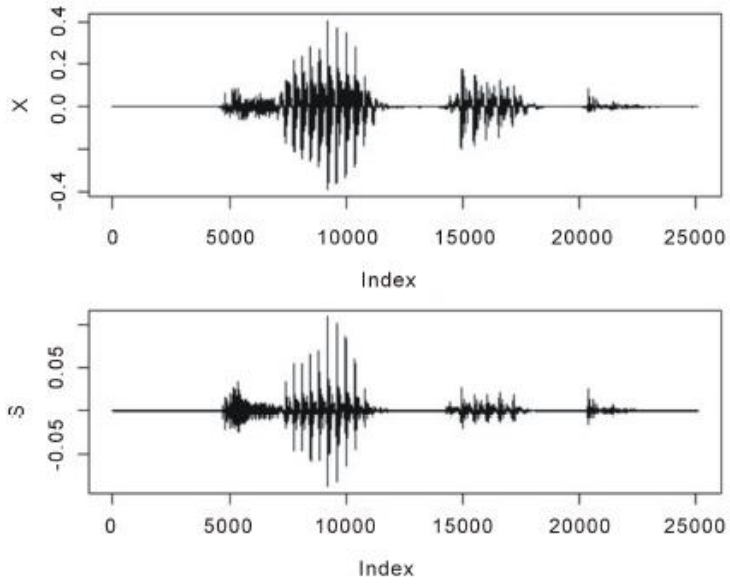

Fig 7: Filtering of the input signal [3]

Feature Extraction

There are 12 features in the dataset. Normally, an audio signal consists of 39 MFCC coefficients, however, in our dataset, 12 essential coefficients are provided, which play a vital role in accent classification. So, in conclusion, our dataset already includes extracted features, but to improve model performance, we attempted a few feature extraction strategies, which are as follows:

1) Principal component analysis (PCA): To reduce dimensionality and increase model performance, PCA was used to extract features from the supplied dataset. After performing, we discovered that X4, X5, and X6 may be deleted because they contribute less to the prediction. We obtain a variance ratio of roughly 79 when we use PCA for four components and an 84 percent variance when we use PCA for five components, therefore PCA equal to 5 is more dependable than PCA equal to 4.

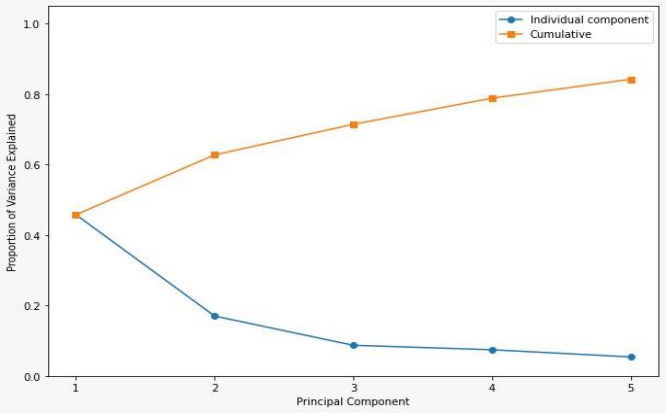

Fig 8: PCA analysis

X9 and X10 have a greater impact on PCA1 than X4, X5, and X6. PCA1 levels rise with X5 and X10 and fall with X4, X6, and X9. This implies that these five coefficients are related; if one grows, the others tend to increase/decrease. As a result, PCA1 may be regarded as a measure of the quality of X4, X5, X6, X9, and X10. The graph depicts the impact of the original predictors (variables) on each component. As we can see in figure 1.

- Too dark = too positive effect
- Too light = too negative effect

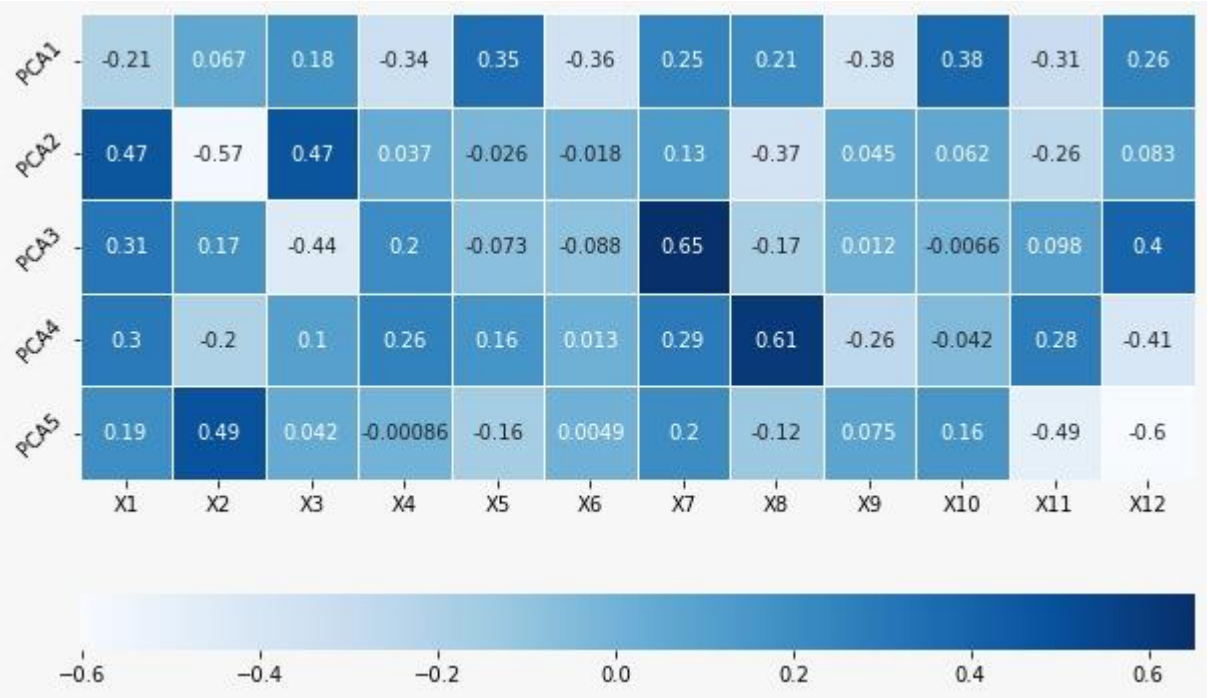

Fig 9: Relation between PCA and MFCC features

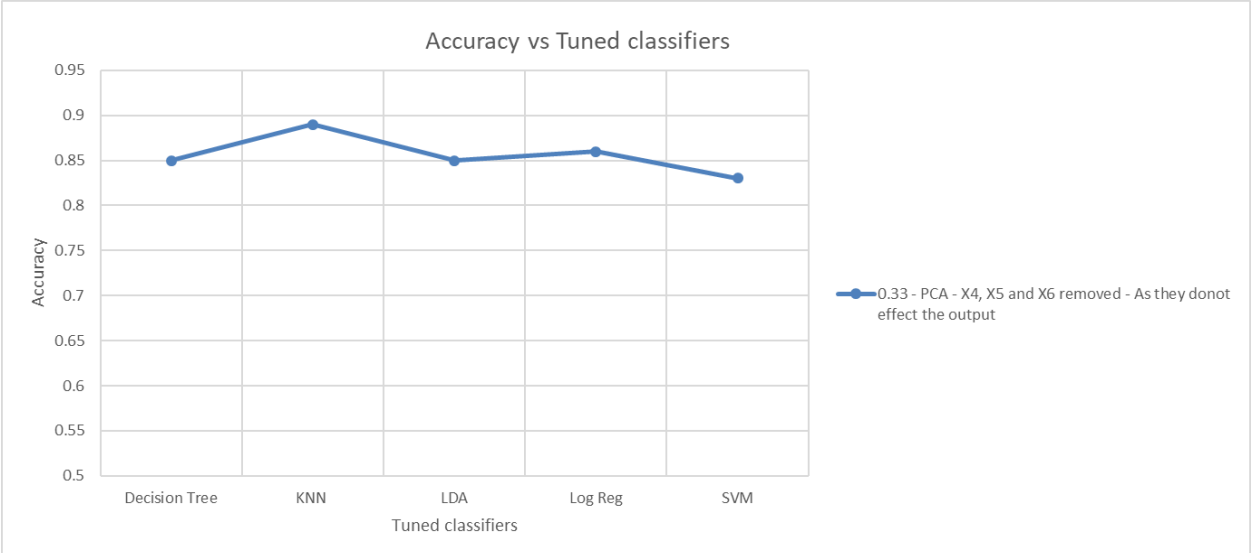

Fig 10: PCA for test split 0.33%

Figure 10, shows that the model's accuracy is lower (range between 0.8 and 0.9 only) when compared to other test split data (equal to 20% or 10%), but when PCA is done on a 33% dataset, its accuracy is higher when compared to the original dataset and test split equal to 33%.

2) Manual inspection: We examined our dataset and discovered that features X8 and X9 contain a high number of outliers, nearly equivalent to 50. IQR technique was used to calculate outliers. As a result, we chose to delete those two characteristics and test the model's accuracy. The findings were comparable to the performance obtained using PCA (except that the accuracy of the decision tree classifier was lower using this technique).

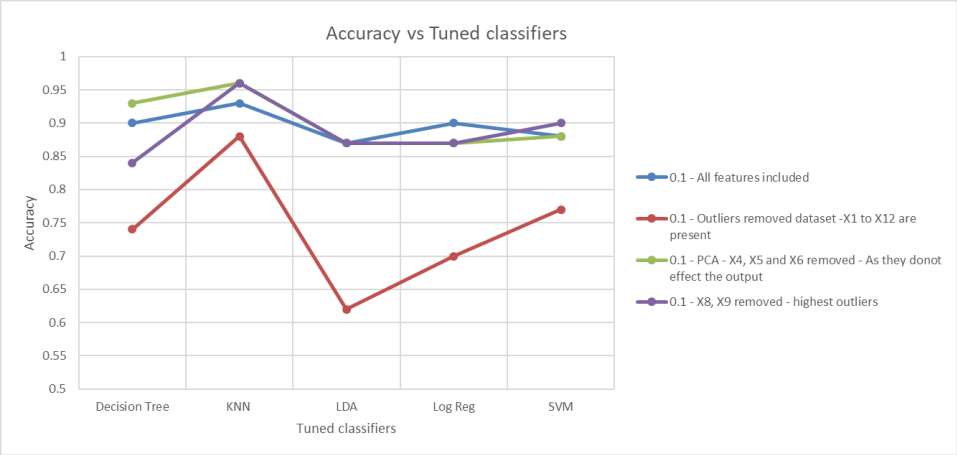

Fig 11: Accuracy after removing outliers and features

In Figure 11, when outliers were eliminated from all features, the available data points for training and testing were reduced, resulting in a decrease in classifier accuracy. Also, in the manual technique, we opted to exclude features with a high number of outliers; after eliminating outliers, we saw that the model's accuracy was growing. For PCA X4, X5, and X6, features were deleted, and we discovered that, unexpectedly, the accuracy of PCA is nearly comparable to the manual technique, with both providing nearly the best accuracy.

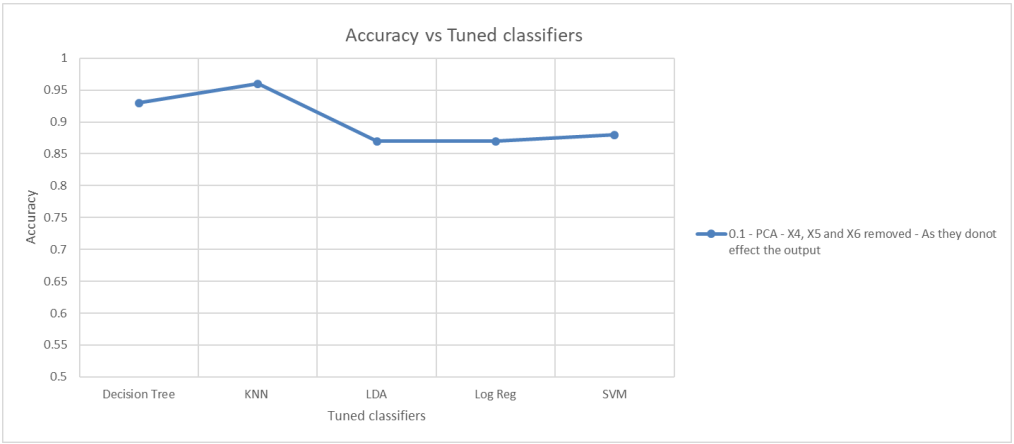

Fig 12: Accuracy after performing PCA

Figure 12, shows that after running PCA (Principal component analysis), the accuracy of the models is growing. We are deleting features X4, X5, and X6 since they have less of an impact on the output value.

Model development

Using the original dataset, we created a baseline model. We also divided the data into several training and test ratios, such as 10% test, 90% train, 20%-80%, and 33-67%, to evaluate the model's performance. We categorized data using five distinct classifiers, which are as follows:

1. K-nearest neighbors (KNN)
2. Linear discriminant analysis (LDA)
3. Logistic regression
4. Decision tree
5. Support vector machine (SVM)

For this dataset, the KNN classifier is utilized to classify data based on its nearest neighbors. To evaluate the model's performance, different numbers of neighbors are examined. When the number of neighbors is equal to 5 or 7, the model's accuracy is higher; when the number of neighbors is equal to 2, the model's accuracy is lowest. KNN classifier performs well when the number of features or data points is big, and it is also beneficial when data is not linearly separable. The accuracy of the KNN classifier is higher than that of the other models employed in this study. For this dataset, KNN had an accuracy of about 90%.

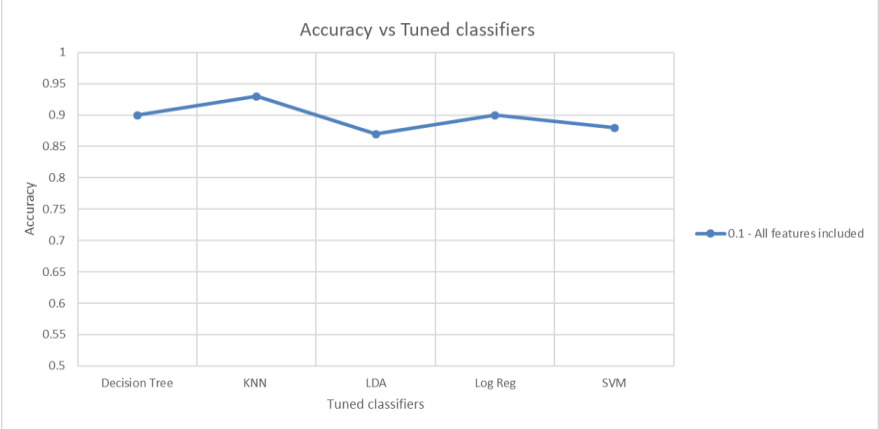

Fig 13: Performance of model when test split equally to 0.10%

Figure 13, shows that when the test to train ratio is 1:9, the decision tree and KNN classifier both performed well over 90 percent, and all other classifiers perform well when compared to other test cases when the test to train ratio is 2:8 and the test split is 33 percent.

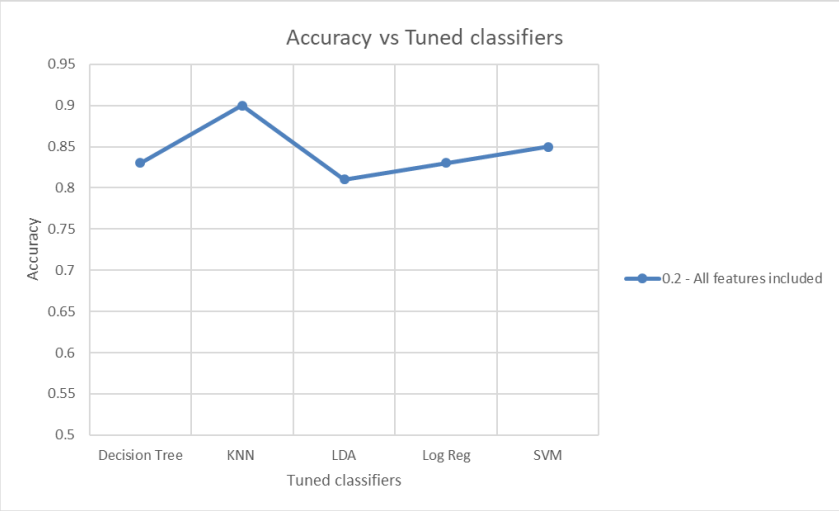

Fig 14: Performance of model when test split equally to 0.20%

Figure 14, shows that the accuracy of the models is worse when the testing to training data ratio is equal to 2:8 than when the test split is equal to 10 %.

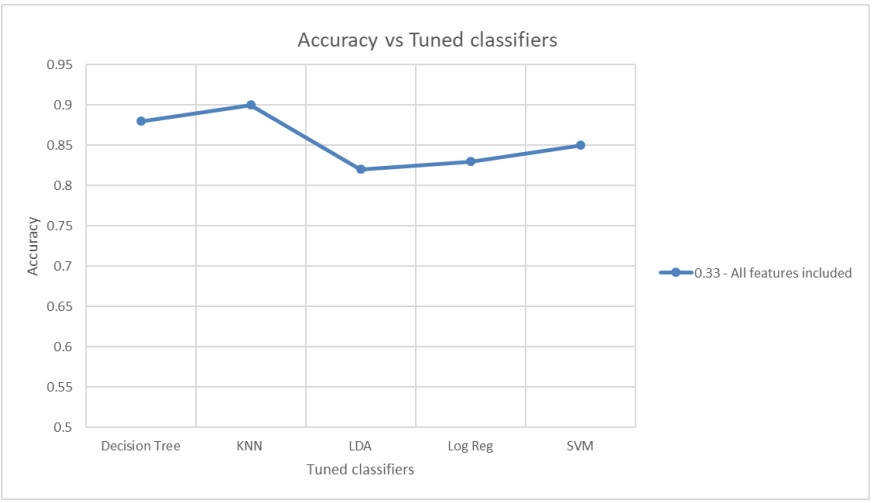

Fig 15: Performance of model when test split equally to 33%

Result: After referring above plots, the KNN classifier produced the best results in all conditions. When the test split was set to 33 percent, the model's performance declined for a few classifiers, such as LDA accuracy, indicating that the data did not have a Gaussian distribution. When the test split was 20 percent, KNN performed virtually identically to when the test split was 33 percent, which was unexpected, and logistic regression accuracy was the same. The decision tree classifier's performance also decreased in this situation, even though it was supposed to improve, compared to the test split of 33 percent. SVM performed substantially identically to other situations when the test split was 10 percent. The KNN classifier performed best in this scenario, and the decision

tree training accuracy was equal to one, indicating that the model was overfitting for the training dataset. So, after evaluating the aforementioned factors, we chose to tune the model to improve its performance for a few classifiers.

Fine-tune your models & Feature Set

Approach 1: Hyperparameters: The accuracy of the model and dataset classification is determined by the different parameters, which are essential elements in determining the model's performance. The parameters are hyperparameters that should be tuned with the help of training data. Few classifiers, such as KNN and decision trees, may be tuned using the number of neighbors or leaves. Furthermore, for other classifiers, changes in the solver or kernel can be used to adapt the model like for logistic regression and LDA. As with the decision tree classifier, we specified the number of leaves and the depth of the tree. We tested the classifier's performance by increasing the number of leaves from 1 to 10. We were obtaining the optimum performance for a specific number of leaves.

For SVM, the kernel plays a key role in deciding model correctness [2]. In certain conditions, kernel poly provided decent performance, and for the poly kernel, we altered the degree to 2,5,8 to produce better performance. LDA and logistic regression solvers were significant in determining classifier performance. The KNN classifier performed better for certain neighbors' models.

Approach 2: Removing outliers: When all outliers were eliminated from all features, only the KNN classifier performs well, whereas the LDA classifier performs poorly.

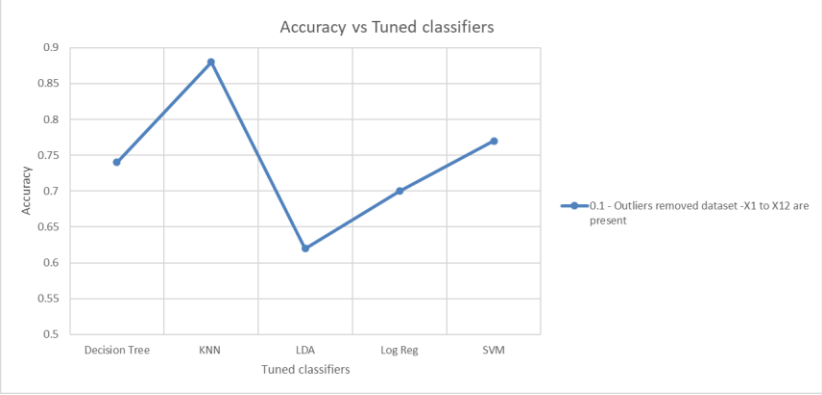

Fig 16: Accuracy after removing all outliers from the dataset

When all outliers are eliminated from all features, accuracy as shown in Figure 16, only the KNN classifier provided acceptable accuracy, whereas the LDA classifier provided the least accuracy. Overall, deleting the outliers was not a wise idea in this case because it affected the performance of all the models.

Approach 3: Eliminating features:

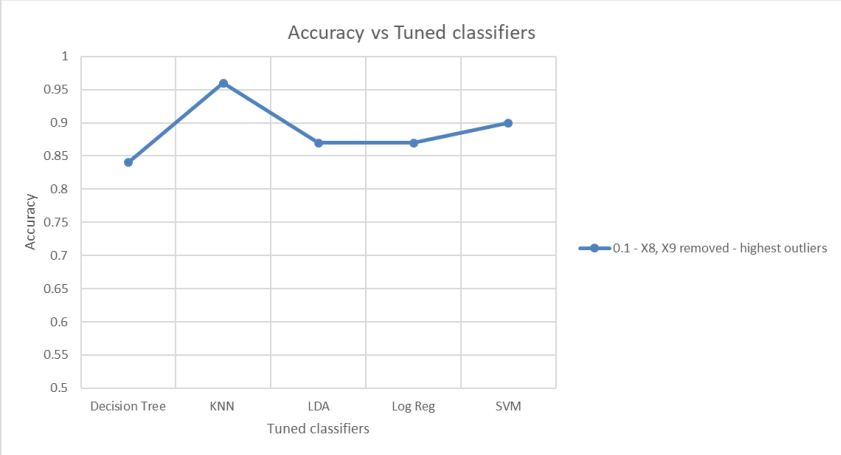

Fig 17: Performance after eliminating features X8, X9

Figure 17, shows that after conducting the manual feature removal strategy, the accuracy of the decision tree model decreased when X8 and X9 features were deleted, while the accuracy of KNN increased, and KNN provided the greatest accuracy. The accuracy of this technique was equal to the accuracy of PCA on the dataset.

Model Performance

Area under the curve (AUC): We did categorization based on US and NON-US accents since the model was less complicated and had higher accuracy. If we had created a model that does data categorization between all of the languages accessible in the dataset, our model have become more complex. When we removed a few characteristics from the dataset, the model's accuracy decreased. Following are graphs of the area under the curve (AUC) parameters to determine the performance of the model.

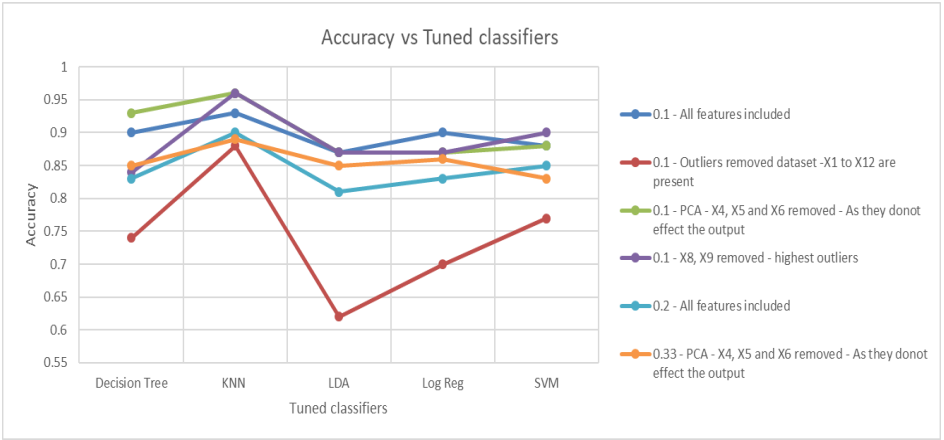

Fig 18: Performance of all models

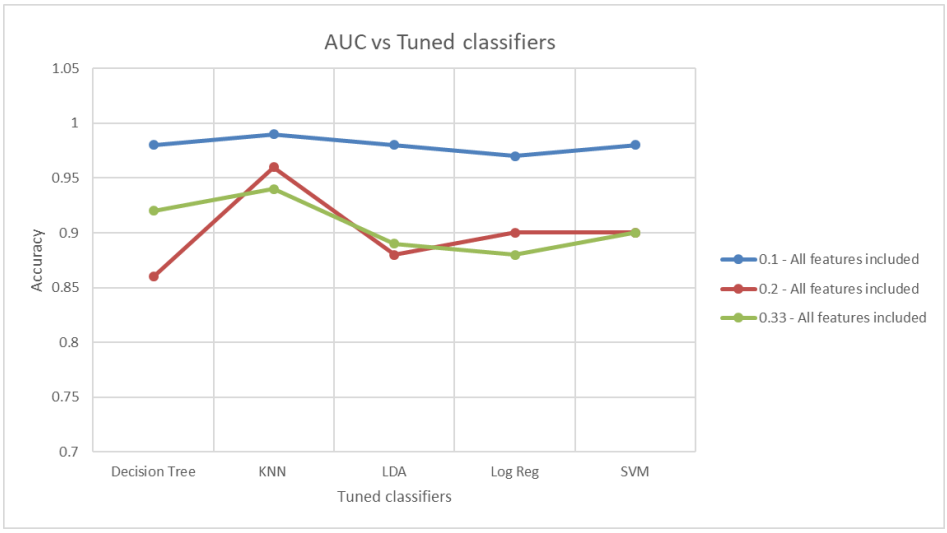

Fig 19: AUC for different test split of data

As seen in figure 19, the AUC indicates the model's performance for a given dataset; if the AUC is near 1, the model is doing well; if it is close to 0, the model is performing poorly. For a test split equal to 10%, all models performed well; for a test split equal to 33%, all models performed poorly.

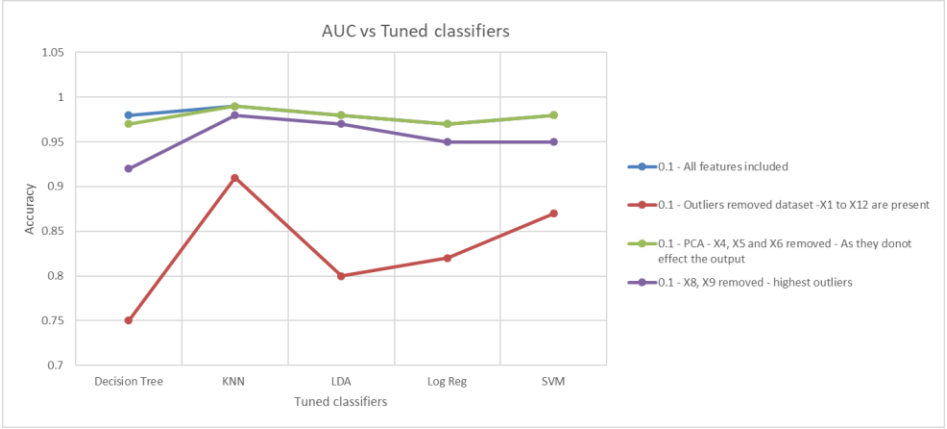

Fig 20: AUC after removing outliers and features

According to Figure 20, when we attempted to extract features from the supplied dataset, the accuracy of the model decreased, when we excluded outliers from all features since there was less data for training and testing (compared to all features included model). However, model accuracy was improved for PCA and manual feature extraction.

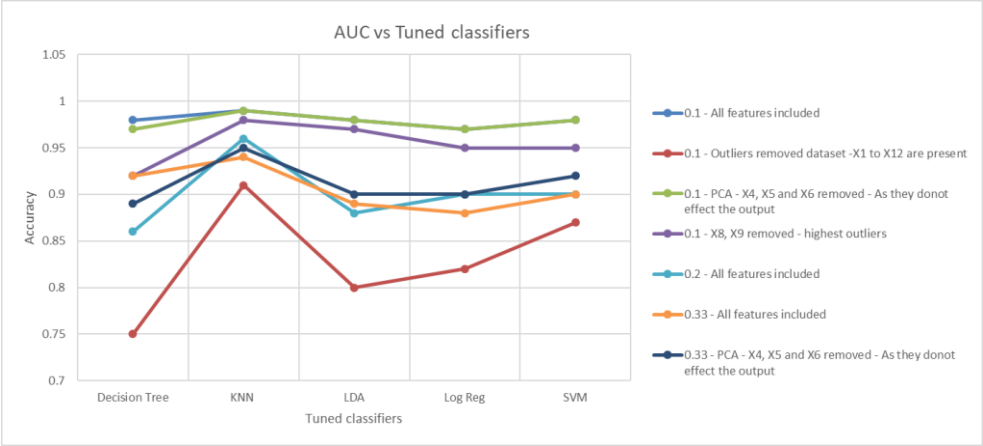

Fig 21: AUC for all the implemented cases

In terms of SVM, if the hyperplane classifies the dataset linearly, the algorithm is known as SVC, and if the dataset is separated using a non-linear technique, the algorithm is known as SVM.

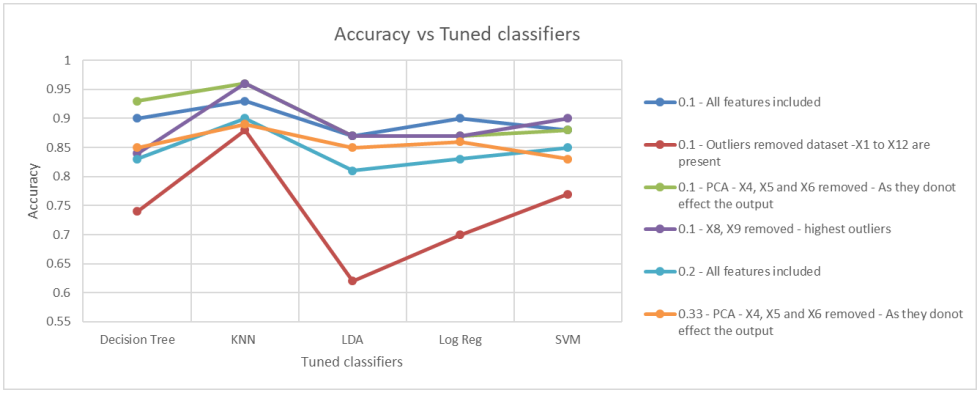

Fig 22: Accuracy of the classifiers for all the implemented cases

Each of the legends indicates:

1. **In blue:** test-to-train split ratio (TTR) of the data of 0.1, and original unchanged datasets used.
2. **In Red:** TTR = 0.1, and dataset with outliers eliminated.
3. **In green:** TTR = 0.1, and dataset after PCA (i.e. features X4, X5, X6 are removed)
4. **In Violet:** TTR = 0.1, and dataset with eliminated features X8, X9
5. **In sky blue:** TTR = 0.2, and the original dataset
6. **In orange:** TTR = 0.33, and dataset after performing PCA (i.e. features X4, X5, X6 are removed)

The preceding graph shows that the LDA performed the poorest across all datasets while the KNN performed the best across all datasets. After doing PCA and manual feature extraction, the model's accuracy significantly improved. The decision tree classifier has the second-best performance.

Cross-validation: We validated the stability of our model using cross-validation, and we also detected model overfitting using cross-validation [4]. We can see the balance between accuracy and recall value by looking at the F1 CV score. F1- score is the harmonic mean of precision and recall. The table below shows that the KNN has a strong cross-validation F1 score, indicating that its accuracy and stability on our dataset are satisfactory. Because LDA has the lowest F1, it is unsuitable for our dataset. Also, the PCA for different test splits, and F1 score is reducing and changing continuously.

	Dataset split	model	F1 CV score	F1 score on test set	Precision score on test set	Recall score on test set	best parameters
0	Org_10	KNeighborsClassifier	0.87	0.7273	0.7286	0.7273	{'n_neighbors': 3}
1	Org_10	LogisticRegression	0.77	0.8178	0.8236	0.8182	{'C': 0.1, 'penalty': 'l1', 'solver': 'libline...
2	Org_10	SVC	0.88	0.7549	0.7753	0.7576	{'C': 100.0, 'gamma': 0.01, 'kernel': 'rbf'}
3	Org_10	LinearDiscriminantAnalysis	0.71	0.7879	0.7893	0.7879	{'solver': 'lsqr'}
4	Org_20	KNeighborsClassifier	0.86	0.7576	0.7576	0.7576	{'n_neighbors': 5}
5	Org_20	LogisticRegression	0.76	0.7395	0.7538	0.7424	{'C': 10, 'penalty': 'l2', 'solver': 'liblinear'}
6	Org_20	SVC	0.86	0.7567	0.7614	0.7576	{'C': 100.0, 'gamma': 0.01, 'kernel': 'rbf'}
7	Org_20	LinearDiscriminantAnalysis	0.73	0.7714	0.7791	0.7727	{'solver': 'lsqr'}
8	Org_33	KNeighborsClassifier	0.87	0.8257	0.8258	0.8257	{'n_neighbors': 3}
9	Org_33	LogisticRegression	0.73	0.7966	0.8089	0.7982	{'C': 0.1, 'penalty': 'l1', 'solver': 'libline...
10	Org_33	SVC	0.85	0.7798	0.7803	0.7798	{'C': 100.0, 'gamma': 0.01, 'kernel': 'rbf'}
11	Org_33	LinearDiscriminantAnalysis	0.70	0.7773	0.7945	0.7798	{'solver': 'lsqr'}
12	PCA data set - 0.1	KNeighborsClassifier	0.85	0.7268	0.7276	0.7273	{'n_neighbors': 7}
13	PCA data set - 0.1	LogisticRegression	0.78	0.8178	0.8236	0.8182	{'C': 0.1, 'penalty': 'l1', 'solver': 'saga'}
14	PCA data set - 0.1	SVC	0.87	0.6667	0.6679	0.6667	{'C': 100.0, 'gamma': 0.01, 'kernel': 'rbf'}
15	PCA data set - 0.1	LinearDiscriminantAnalysis	0.74	0.7571	0.7623	0.7576	{'solver': 'lsqr'}
16	PCA data set - 0.2	KNeighborsClassifier	0.84	0.6800	0.6861	0.6818	{'n_neighbors': 5}
17	PCA data set - 0.2	LogisticRegression	0.77	0.7861	0.7977	0.7879	{'C': 1, 'penalty': 'l2', 'solver': 'saga'}
18	PCA data set - 0.2	SVC	0.85	0.8008	0.8173	0.8030	{'C': 10.0, 'gamma': 0.01, 'kernel': 'rbf'}
19	PCA data set - 0.2	LinearDiscriminantAnalysis	0.71	0.7714	0.7791	0.7727	{'solver': 'lsqr'}
20	PCA data set - 0.33	KNeighborsClassifier	0.86	0.7337	0.7345	0.7339	{'n_neighbors': 5}
21	PCA data set - 0.33	LogisticRegression	0.75	0.8062	0.8162	0.8073	{'C': 0.1, 'penalty': 'l1', 'solver': 'saga'}
22	PCA data set - 0.33	SVC	0.84	0.7504	0.7590	0.7523	{'C': 10.0, 'gamma': 0.1, 'kernel': 'rbf'}
23	PCA data set - 0.33	LinearDiscriminantAnalysis	0.70	0.7972	0.8050	0.7982	{'solver': 'lsqr'}

Fig 23: cross-validation scores

F1 score formula = 2 * (Precision * recall) / (Precision + recall)

Conclusion

1) LDA models on the linear dataset: The LDA classifier performs poorly on the dataset. It was unable to distinguish between data points precisely. We attempted to improve LDA accuracy by using Principle component analysis (PCA), however, the model's performance remained unchanged. The solver parameter is important in determining the model's accuracy. We discovered in between LDA and logistic regression when one classifier outperforms, the other performs well for the dataset.

2) KNN showed the best performance: For all scenarios, the KNN classifier performed better. It had an accuracy between 0.85 and 0.95. When we ran PCA on KNN, its performance improved as well. The model performed well with non-linear data. We also attempted to test the accuracy of KNN for various numbers of neighbors, and we found that the best accuracy was obtained for specific numbers that we examined.

3) Decision tree showed 2nd best performance: For this dataset, the decision tree classifier performed second best. We examined its performance under various scenarios. However, when we attempted to do manual feature extraction, i.e. removing X8, and X9 features, its performance did not improve as predicted. We also examined decision tree performance by varying the number of leaves, selecting the leaf with the greatest performance, and then training the model.

4) SVM model: Kernels were also important in deciding the performance of the SVM classifier. We attempted to assess SVM performance using various kernels such as poly, linear, and so on, and the accuracy varied depending on the kernel. We also experimented with SVM using the SVM() and SVM.SVC() routines. However, when compared to the KNN classifier, SVM does not get the greatest results.

5) PCA analysis: After doing the Principle component analysis on our dataset and under various scenarios. We discovered that it improves the accuracy of all of our models except the logistic regression, which did not perform as expected.

6) Manual feature extraction: We analyzed the dataset and discovered that features X8 and X9 had a high number of outliers, so we removed those features and tested them; surprisingly, it worked and provided good performance for all models except the decision tree classifier, whose accuracy was nearly equal to the dataset after performing PCA.

7) Fine-tuning: We attempted to tweak the models after eliminating the outliers from the dataset, however, because the data points were restricted after removing the outliers, there was less data for training and testing, resulting in lower model accuracy.

Future suggestions: We did binary classification on the dataset because we classed it as having a USA accent or a non-USA accent. In the future, we will be able to train the models to categorize data into its appropriate accent form/output class.

Issues: Because the dataset is small, removing outliers reduces the accuracy of our model. As a result, we were unable to eliminate the outliers from our data. Because the dataset contained fewer features, there was less room for fine-tuning the models.

Codes referred from other sources

Cross-Validation: We have referred to the code and implementation of cross-validation on the online website[1].
PCA: We learned about PCA implementation from an online website. We followed the code and executed it in our model [5].

Reference:

[1] Cross-validation: evaluating estimator performance [online]

Available: https://scikit-learn.org/stable/modules/cross_validation.html

[2] Implementing SVM and Kernel SVM with Python's Scikit-Learn [online]

Available:https://stackabuse.com/implementing-svm-and-kernel-svm-with-pythons-scikit-learn/ [Accessed: 09-May-2019]

[3] A Comparison of Classifiers in Performing Speaker Accent Recognition Using MFCCs [online]

Available:https://www.researchgate.net/publication/271710350_A_Comparison_of_Classifiers_in_Performing_Speaker_Accent_Recognition_Using_MFCCs [Accessed: Jan-2015]

[4] 4 Ways to Evaluate your Machine Learning Model: Cross-Validation Techniques [online]

Available: https://www.analyticsvidhya.com/blog/ [Accessed: 21-May-2021]

[5] MazinOnsa / SPEAKER-ACCENT-RECOGNITION [online]

Available:https://github.com/MazinOnsa/SPEAKER-ACCENT-RECOGNITION/blob/main/SAR-MFCC.ipynb [Accessed: 21-March-2021]