Automatic Speaker Recognition System based on Optimised Machine Learning Algorithms

1st Tumisho Billson Mokgonyane
Department of Computer Science
University of Limpopo
Polokwane, South Africa
mokgonyanetb@gmail.com

2nd Tshephisho Joseph Sefara
Next Generation Enterprises and Institutions
Council for Scientific and Industrial Research
Pretoria, South Africa
tsefara@csir.co.za

3rd Thipe Isaiah Modipa
Department of Computer Science
University of Limpopo
Polokwane, South Africa
thipe.modipa@ul.ac.za

4th Madimetja Jonas Manamela
Department of Computer Science
University of Limpopo
Polokwane, South Africa
jonas.manamela@ul.ac.za

***Abstract*—Speaker recognition is a technique that automatically identifies a speaker from a recording of their voice. Speaker recognition technologies are taking a new trend due to the progress in artificial intelligence and machine learning and have been widely used in many domains. Continuing research in the field of speaker recognition has now spanned over 50 years. In over half a century, a great deal of progress has been made towards improving the accuracy of the system's decisions, through the use of more successful machine learning algorithms. This paper presents the development of automatic speaker recognition system based on optimised machine learning algorithms. The algorithms are optimised for better and improved performance. Four classifier models, namely, Support Vector Machines, K-Nearest Neighbors, Random Forest, Logistic Regression, and Artificial Neural Networks are trained and compared. The system resulted with Artificial Neural Networks obtaining the state-of-the-art accuracy of 96.03% outperforming KNN, SVM, RF and LR classifiers.**

***Index Terms*—Speaker recognition, support vector machine, k-nearest neighbors, artificial neural networks, multilayer percerptron, random forest, logistic regression**

I. Introduction

Automatic speaker recognition is a technique used to automatically recognise the identity of a speaker from a recording of their voice. Speaker recognition is an important topic in signal processing and has a variety of applications, especially in security systems [1]. Voice controlled systems and devices rely heavily on speaker recognition. Speaker recognition consists of two fundamental tasks, namely *speaker verification* and *speaker identification*. Speaker verification is the task of determining whether an unknown voice is from a particular enrolled speaker. The speaker in this case provides a voice sample with a claim to be one of the enrolled speakers and the system either rejects or accepts the claimed identity. Speaker identification is the task of associating an unknown voice with one from a set of enrolled speakers. The speaker provides a voice sample and the system determines to which of the known set of speakers the voice sample belongs. Speaker identification systems can be classified based on the system range of operation, the classification can either be closed-set or open-set [2]. In a closed-set identification, the speakers are all enrolled into the speaker database and the speaker with the closest match to the test signal is chosen to be the test speaker. In the case of open-set identification, speakers need not always be enrolled into the database, thus the system needs to perform an additional task of rejection in case the speaker is someone from outside the speaker database [2].

In addition, speaker recognition systems can be classified by the constraints placed on the input text corresponding to the speech used to train and test the system. The classification can either be text-dependent or text-independent. In the text-dependent case, the input sentence or phrase is fixed for each speaker, and in the text-independent case, there is no restriction on the sentence or phrase to be spoken [3]. Text-dependent speaker recognition system is suited for services such as telephone-based services and access control, where the users are considered cooperative [4]. Text-independent speaker recognition system is suited for application areas such as forensics and surveillance where speakers can be considered non-cooperative users, as they do not specifically wish to be recognised.

Research in the field of speaker recognition has now been in existence for over 50 years [5]. However, there is limited research conducted in the context of African indigenous languages, South African languages in particular. South African official languages are still classified as being highly under-resourced [6]–[9].This paper presents the development of an automatic speaker recognition system based on optimised machine learning algorithms. The system incorporates the classification of the Sepedi language speakers and can be used to automatically authenticate speaker identities using their voices to allow only the identified persons an access right to information systems or to facilities that need to be protected from the intrusion of unauthorised persons. We chose

the Sepedi language because it belongs to the 11 South African official languages and largely spoken in the Limpopo province of South Africa [10].

This paper is outlined as follows: Section II gives a brief discussion on machine learning algorithms. Section III details the implementation of the speaker recognition system.Section IV discusses the experimental results. The paper is concluded in Section V.

II. Machine Learning Algorithms

This section discusses the machine learning algorithms used for training speaker recognition systems.

A. *K-Nearest Neighbors*

The K-Nearest Neighbors (KNN) algorithm belongs to a family of instance-based and lazy learning algorithms [11]. The KNN classifier works by taking a data point and looking at the k closest labeled data points. The data point is then assigned the label of the majority of the k closest points. An example of a KNN classification is depicted in Figure 1a. The test sample (dot) should be classified either to Class A of red squares or to Class B of blue triangles. If $k = 3$ (inner circle), the test sample is assigned to Class A because there are 2 squares and only 1 triangle inside the inner circle. If $k = 7$ (outer circle) the test sample is assigned to the Class B (4 triangles vs. 3 squares inside the outer circle). Kacur *et al.* [12] motivates the use of KNN classifier for the task of speaker recognition. The authors report that the KNN classifier trained with four nearest neighbours $(k = 4)$ yields the best accuracy results.

B. *Random Forest*

The Random Forest (RF) algorithm depicted in Fig. 1b, is a supervised machine learning algorithm for regression and classification tasks that work by building a large number of decision trees at training stage and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees [13]. RF is a highly accurate and robust method because of the number of decision trees participating in the process. In most cases, RF does not suffer from the overfitting problem because it takes the average of all the predictions, which cancels out the biases. The RF algorithm was used successfully for automatic speaker recognition tasks and reported to have achieved *state-of-the-art* performances [8].

C. *Logistic Regression*

Logistic Regression (LR) shown in Fig. 1c is a highly accurate and robust method that uses *multinomial logistic regression* method to generalise logistic regression to multiclass problems [14]. In most cases, LR does not suffer from the problem of overfitting because it has few parameters, and have a regulariser parameter to manage the overfitting. LR and its sparse version of kernel logistic regression method has shown to outperform the SVM and the Gaussian mixture models baseline system for text-independent speaker identification [15].

D. *Support Vector Machines*

The Support Vector Machines (SVM) is a supervised learning model with associated learning algorithms that analyse data and recognise patterns, used for classification and regression analysis [16]. SVM is a popular discriminative classifier which models the boundary between a speaker and a set of impostors and has proven to be a powerful technique for pattern classification [16], [17]. As shown in Figure 1d, the SVM is a classifier which models the decision boundary between two or more classes as a separating hyperplane.

E. *Multilayer Perceptrons*

A Multilayer Perceptron (MLP) is a feed-forward artificial neural network model that maps sets of input data onto a set of appropriate outputs. An MLP classifier consists of multiple layers where each layer is fully connected to the next layer. The nodes of the layers are neurons using non-linear activation functions, except for the nodes of the input layer. There can be one or more non-linear hidden layers between the input and the output layer [18], as shown in Figure 2. MLPs can handle extremely complex tasks, however they take a lot of time to train and are computationally expensive [16].

The current state-of-the-art methods of applying neural networks to speaker recognition, has been proven to be successful [19]. Wang *et al.* [19] apply neural networks to build a speaker recognition system using MFCCs features. The authors realised that when the number of speakers increases, the rate of recognition decreases. Hence, their solution was to increase the samples per speaker as the number of speakers increases.

III. Methodology

This section discusses the methods and procedures followed in this study, the section covers the dataset, feature extraction and normalisation, classifier model setup and evaluation.

A. *Dataset*

The data is obtained from the National Centre for Human Language Technology (NCHLT) [20], [21]. The Sepedi NCLHT speech data is used, the data contains speech audio files recorded by different speakers. We randomly sampled 100 speakers and 150 samples per speaker. Each sample is a recorded sentence of about 3-5 words saved as a mono waveform file. The data summarised in Table I is partitioned into train and test partitions of 80% train data and 20% test data set, that is 120 train samples and 30 test samples for each speaker.

TABLE I
Data Statistics

Unit	Value
No. of speakers	100
Instances per speakers	150
Total Duration (minutes)	821.45
Size (MB)	1638.40

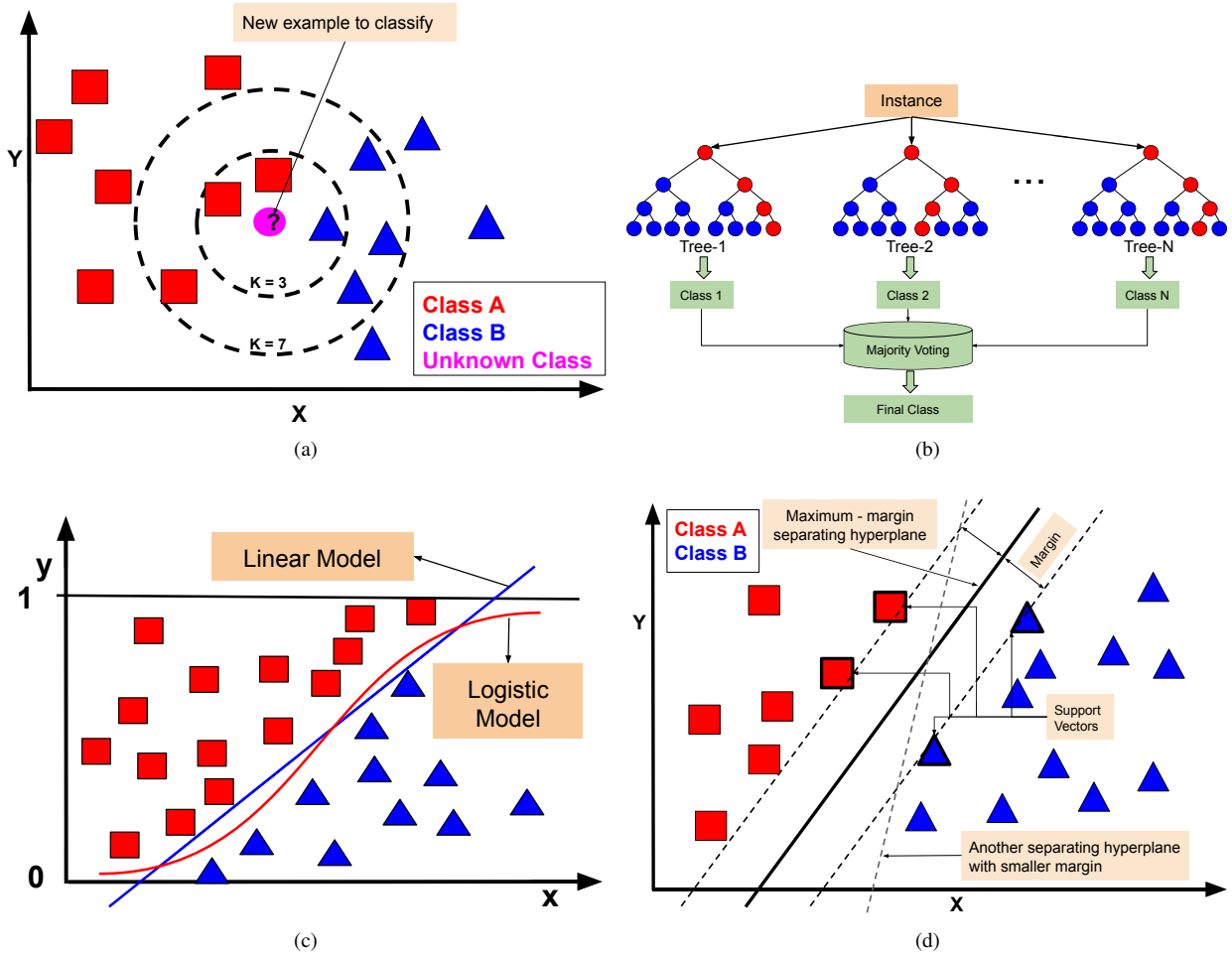


Fig. 1. (a) KNN classification example. (b) Random Forest. (c) Logistic regression lines. (d) A maximum-margin hyperplane that separates class A and class B training examples is found by an optimisation process.

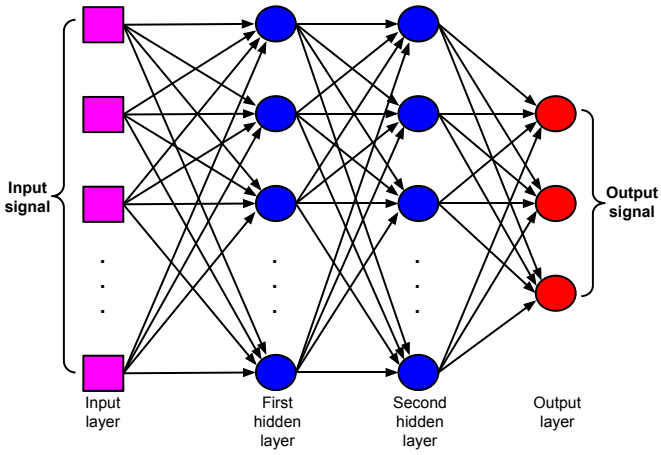


Fig. 2. MLP architecture with two hidden layers

B. Feature Extraction

The human voice contains numerous discriminative acoustic features of speech that can be used to identify speakers. Feature extraction is one of the most important aspect of speaker recognition. This step generates feature vectors that represent each speech signal. We extract acoustic features of speech using pyAudioAnalysis – an open-source comprehensive package developed in Python [22]. pyAudioAnalysis extracts a set of 34 short-term features discussed in [22]. The features include **Time-domain features** (Zero Crossing Rate, Energy, and Entropy of Energy), **Frequency-domain features** (Spectral Centroid, Spectral Spread, Spectral Entropy, Spectral Flux, Spectral Rolloff, Chroma Vector, and Chroma Deviation) and **Cepstral-domain features** (MFCCs). Fig. 3 shows the Time-domain (ZCR), Frequency-domain (Spectral Centroid) and Cepstral-domain (MFCC) features extracted from a single audio file of one speaker.

C. Feature Normalisation

Feature normalisation is an important aspect for a robust speaker recognition system. Its goal is to eliminate speaker and recording variability. We adopt the mean variance normalisation method [23], [24] where we normalise features so that they are centered around 0 with a standard deviation of

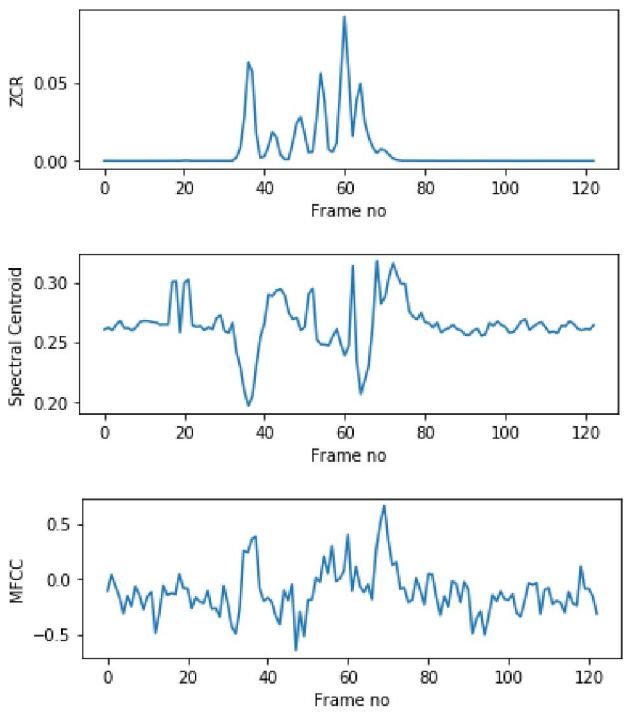


Fig. 3. Time, Frequency and Cepstral features extracted from a single audio file.

1. The normalised feature y_i is given as,

$$y_i = \frac{x_i - \mu}{\sigma} \tag{1}$$

where σ represents the variance and μ represent the mean for each feature x_i.

D. Classification Model Setup

Scikit-Learn [25] and Tensorflow [26] are used to train the machine learning algorithms discussed in Section II. The parameters for the machine learning algorithms are discussed below, we used *grid search* to find the best hyperparameters.

1) ***K-Nearest Neighbors (KNN):*** The KNN classifier is trained with different values of k which refers to the number of nearest neighbors and the *weight* parameter which refers to the weight function used in prediction. Possible values for the *weight* parameter are:

- **uniform**: uniform weights. All points in each neighborhood are weighted equally.
- **distance**: weight points by the inverse of their distance. In this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.

2) ***Random Forest (RF):*** The RF classifier is trained with the parameters max_depth and $n\_estimators$. The $n\_estimators$ parameter is the number of trees in the forest and the max_depth parameter is the maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than minimum number of samples required to split an internal node.

3) ***Logistic Regression (LR):*** We train LR classifier using Tensorflow. The classifier is compiled for 100 epochs with *stochastic gradient descent (SGD)*, adaptive learning rate method for gradient descent called *Adadelta* [27] and *Nadam* [28] which is essentially Adam [29]

4) ***Support Vector Machines (SVM):*** The SVM classifier is trained with the penalty parameter *C* of the error term and the kernel in use can be either *Linear, Polynomial, Radial Basis Function (rbf)* or *Sigmoid* kernel. The kernels are defined by the following equations:

$$Linear = \langle x, x' \rangle \tag{2}$$

$$Polynomial = (\gamma \langle x, x' \rangle + r)^d \tag{3}$$

$$RBF = exp(-\gamma \|x - x'\|^2) \tag{4}$$

$$Sigmoid = \tanh(\gamma \langle x, x' \rangle + r) \tag{5}$$

where γ is a positive parameter, d is the degree of the kernel, and r is the coefficient.

5) ***Multilayer perceptron (MLP):*** The MLP classifier is trained with two hidden layers each consisting 256 neurons, the activation function is set to *rectified linear unit* for each layer and the output layer is activated with *softmax* function. We use *softmax* because our data is categorical and softmax takes as input a vector of real numbers, and normalises it into a probability distribution consisting of probabilities. The standard (unit) *softmax* function $\sigma : \mathbb{R}^K \to \mathbb{R}^K$ is defined by the formula:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\Sigma_{j=1}^{K} e^{z_j}} \tag{6}$$

where $i = 1, \ldots, K$, $z = (z_1, \ldots, z_K) \in \mathbb{R}^K$, and z_i is an element of the input vector $\mathbf{z}$. The classifier is compiled with *SGD*, *Adadelta* and *Nadam* for 100 epochs.

E. Evaluation

The behaviour of each model is evaluated based on certain criteria to assess its performance. The following evaluation measurements are used to evaluate the performance of the models.

1) ***Accuracy:*** The percentage of the samples which are correctly classified from all the samples given. We report on validation accuracy to avoid overfitting and testing accuracy to avoid underfitting.

2) ***Precision:*** The ability of a classifier to return only relevant instances, that is the proportion of the examples which truly have class x among all those which were classified as class x.

3) ***Recall:*** The ability of a classifier to identify all relevant instances, that is the proportion of examples which were classified as class x, among all examples which truly have class x.

4) F_1 ***score:*** A single metric that combines recall and precision using the harmonic mean.

*5) **Root Mean squared error (RMSE)**:* The RMSE is a quadratic scoring rule which measures the average magnitude of the error.

*6) **Categorical Cross Entropy**:* Measures the performance of a classifier model whose output is a probability value between 0 and 1.

IV. Experimental Results

This section discusses the results of the classifier models. We provide results for parameter optimisation, results for overfitting and how the classifiers perform with their best hypeparameter values.

A. Results on Parameter Optimisation

The KNN parameter optimisation results are shown in Figure 4a. It is observed that the best accuracy is achieved when the KNN classifier is trained with 14 nearest neighbors and the *weight* parameter set to **distance**. This means that closer neighbors of a query point have a greater influence than neighbors which are further away. The KNN classifier does not perform well when all points in each neighborhood are weighted equally (setting the *weight* parameter to **uniform**). It is also observed that the accuracy decreases as the number of nearest neighbors (k) increase for both *weight* parameters, meaning that when k is set to a larger value, the KNN classifier misclassifies most of the data. From this results, grid search selected $k = 14$ as the best hyperparameter and set the *weight* parameter to **distance**.

For RF, we optimise two parameters which are **depth** and **estimators (trees)**. When using a smaller (limiting) depth of the tree, the accuracy is lower. This means we have reduced the variance (good) of the decision tree but at the cost of increasing the bias (bad). However, when the depth of the tree increases, the accuracy also increases as shown in Figure 4b. When increasing the number of decision trees from 0 to 40 in Figure 4b the accuracy increases, hence, depth and number of trees affect the performance of the RF yielding good results when increased to a certain limit. From this results, grid search selected 147 trees with the depth of 41 as the best hyperparameters.

Figure 4c shows the results for the SVM's *Linear, RBF, Polynomial* and *Sigmoid* kernels trained with different values of the parameter *C*. The results show that the best accuracy is achieved when the *RBF* kernel is in use for lower values of the parameter *C*. As the value of the parameter *C* increases, the accuracy remains constant for the *Linear* kernel and the accuracy for the *RBF* kernel slightly increases. The *RBF* accuracy does not go below the *Linear* kernel's accuracy after $C = 3$. The accuracy is poor for the *Polynomial* and *Sigmoid* kernels. We observe *Sigmoid* decreasing performance as *C* increases. From this results, grid search selected the RBF kernel and $C = 3$ as the best hyperparameters.

The learning curves for LR and MLP are given in Fig. 4d for accuracy and Fig. 5 for cross entropy loss function. From Fig. 4d, we observe that SGD is slow to converge even after 100 epochs. Followed by Adadelta for LR and Nadam for LR. MLP is quicker to converge on Adadelta and Nadam. Although, Adadelta for MLP did not converge quicker than Nadam we observe from the curve that Adadelta is better than Nadam.

B. Results on Overfitting

To precisely measure the accuracy of the models, overfitting is a crucial task that needs to be investigated when building machine learning models. We use cross entropy loss function to measure how the model react to new observations. If the cross entropy loss increases during testing then the model is no longer learning correctly, hence, there is a need to optimise the model. We investigate overfitting in Fig. 5, we observe the learning curves for most the optimisers did not increase but kept decaying. Adadelta and Nadam started at a lower cross entropy loss for MLP but Nadam seems likely to overfit. Hence, we selected Adadelta for MLP and Nadam for LR as optimisers.

To measure overfitting for RF, when a single decision tree is used is very sensitive to data variation and can overfit easily to noise in the data. When more trees are added the tendency to overfit decreases.

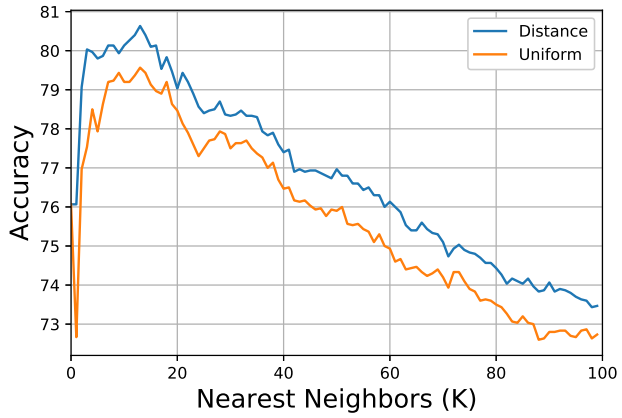
To measure overfitting for SVM. The *C* parameter is used by the SVM optimiser to avoid misclassifying each training sample. The optimiser selects a smaller-margin separating hyperplane for higher value of *C*. Conversely, a very smaller value of *C* causes the optimiser to search for a larger-margin separating hyperplane, which may results with the hyperplane misclassifying more samples. Hence, we selected $C = 3$ as the best hyperparameter when *RBF* is the kernel.

To avoid overfitting for KNN, we need to select K high enough to avoid overfitting, but small enough to avoid underfitting the distribution. The grid search selected $k = 14$ as the best hyperparameter.

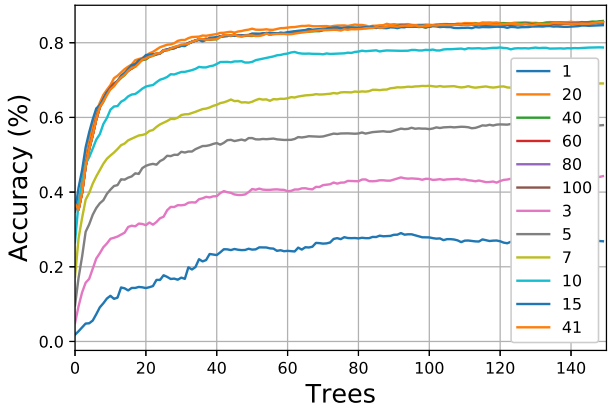
C. Results on Performance of Optimised Parameters

Figure 6 shows the accuracy of the classifier models trained with default parameters against hyperparameters. It is observed that the classification accuracy is poor when the classifiers are trained with default parameters and improves when the models are trained with hyperparameters. KNN's default parameters give an accuracy of 78.80% whereas the hyperparameters improve the accuracy by 2.13%. Random forest hyperparameters improve the accuracy by 19.80%. The accuracy of Logistic regression is not really affected by the hyperparameters as it improves by only 0.27%. SVM and MLP accuracy's increase by 1.57% and 2.20% respectively.

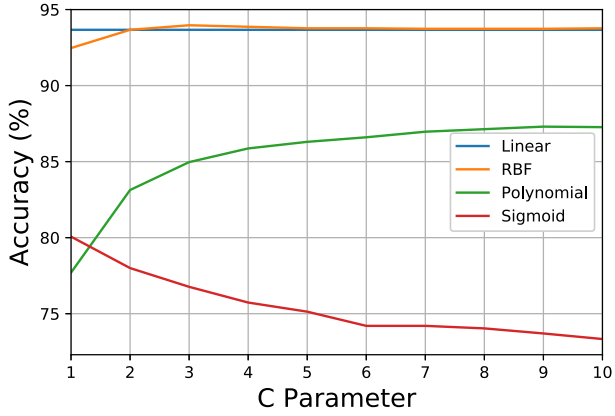
The results obtained when the classifier are trained with hyperparameters are shown in Table II. MLP outperforms all the classifiers and KNN and RF have the lowest accuracy. A difference of 0.30% is observed between LR and SVM classifiers. Table II also shows that both precision and recall are similar to the classification accuracy results for RF, LR, SVM, and MLP models. Hence, it is enough to use accuracy. On the other hand, precision and recall of KNN has a difference of 1.97% which is high and infer that accuracy is not enough



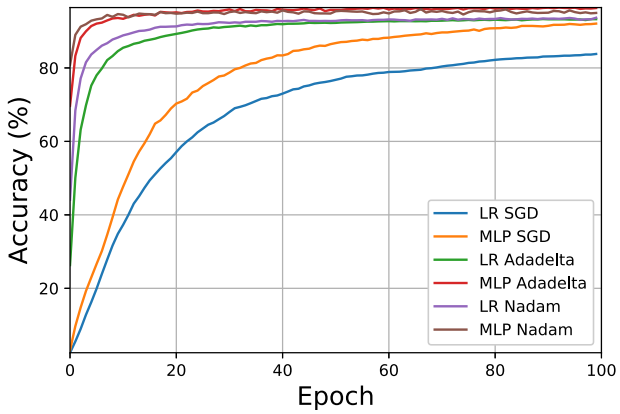
(a)



(b)



(c)



(d)

Fig. 4. (a) Accuracy for KNN against number of nearest neighbors. (b) Accuracy for RF against number of trees for various depth. (c) Accuracy for SVM against C for various kernels. (d) Accuracy for LR and MLP using SGD, Adadelta, and Nadam optimisers.

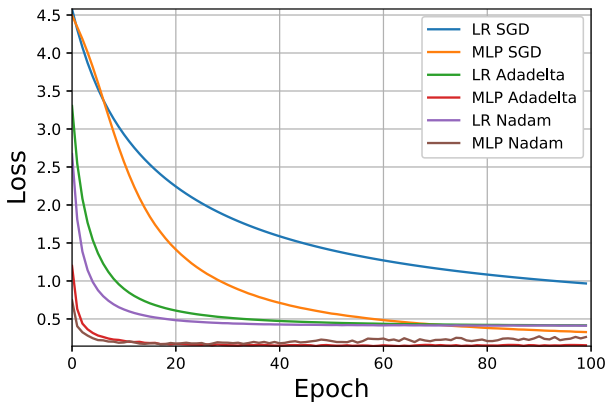
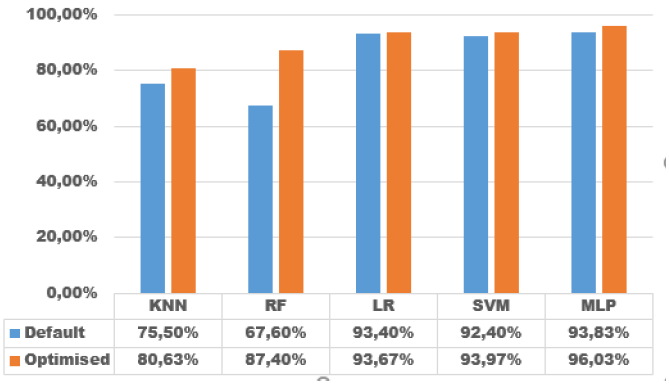


Fig. 5. Loss function for LR and MLP



	KNN	RF	LR	SVM	MLP
Default	75,50%	67,60%	93,40%	92,40%	93,83%
Optimised	80,63%	87,40%	93,67%	93,97%	96,03%

Fig. 6. Accuracy using default and optimised parameters

to measure the performance. This high difference is mostly caused by unbalanced labels. In this case we use F_1 score as the final measure. Hence, KNN resulted with F_1 score of 79.98%.

TABLE II
PERFORMANCE OF THE CLASSIFIER MODELS.

Performance Measure	Classification Models				
	KNN	***RF***	***LR***	***SVM***	***MLP***
Accuracy	80.63%	87.40%	93.67%	93.97%	96.03%
Precision	82.60%	87.59%	93.82%	94.15%	96.16%
Recall	80.63%	87.40%	93.67%	93.97%	96.03%
F1 Score	79.98%	87.19%	93.65%	93.94%	96.02%
RMSE	19.51	14.84	10.47	11.03	7.67

V. CONCLUSION AND FUTURE WORK

This paper reported on the development of an automatic speaker recognition system based on optimised machine learning algorithms. The paper briefly described all the stages (training and testing) which covered feature extraction and normalisation, classification model training and evaluation. The dataset of Sepedi speech data was obtained from the NCHLT project. Features were extracted using pyAudioAnalysis library and we used Scikit-Learn and Tensorflow to train the classification models. The *KNN, Random Forest, Logistic Regression, SVM* and *MLP* classifiers are first trained with their default parameters and trained secondly with their hyperparameters. The results are compared and it is observed that the performance is poor for default parameters and improves when the hyperparameters are used. As an extension to this study we shall investigate how spoken languages impact the accuracy of automatic speaker recognition systems. We shall also study the effects of acoustic features of speech towards a text-independent automatic speakers recognition system focusing on under-resourced languages.

ACKNOWLEDGMENT

This study was facilitated at the University of Limpopo, Department of Computer Science and is based on research supported by the University of Limpopo, Telkom Centre of Excellence for Speech Technology and Council for Scientific and Industrial Research.

REFERENCES

[1] N. Singh, R. Khan, and R. Shree, "Applications of speaker recognition," *Procedia engineering*, vol. 38, pp. 3122–3126, 2012.

[2] H. B. Kekre and V. Kulkarni, "Closed set and open set speaker identification using amplitude distribution of different transforms," in *2013 International Conference on Advances in Technology and Engineering (ICATE)*, 2013, pp. 1–8.

[3] Y. Liu, Y. Qian, N. Chen, T. Fu, Y. Zhang, and K. Yu, "Deep feature for text-dependent speaker verification," *Speech Communication*, vol. 73, pp. 1–13, 2015.

[4] T. Kinnunen and H. Li, "An overview of text-independent speaker recognition: From features to supervectors," *Speech communication*, vol. 52, no. 1, pp. 12–40, 2010.

[5] S. Furui, "50 years of progress in speech and speaker recognition research," *ECTI Transactions on Computer and Information Technology (ECTI-CIT)*, vol. 1, no. 2, pp. 64–74, 2005.

[6] F. de Wet, J. Badenhorst, and T. Modipa, "Developing speech resources from parliamentary data for South African English," *Procedia Computer Science*, vol. 81, pp. 45–52, 2016.

[7] P. J. Manamela, M. J. D. Manamela, T. Modipa, T. J. Sefara, and T. B. Mokgonyane, "The automatic recognition of Sepedi speech emotions based on machine learning algorithms," *2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)*, pp. 1–7, 2018.

[8] T. B. Mokgonyane, T. J. Sefara, T. I. Modipa, M. M. Mogale, M. J. Manamela, and P. J. Manamela, "Automatic speaker recognition system based on machine learning algorithms," in *2019 Southern African Universities Power Engineering Conference/Robotics and Mechatronics/Pattern Recognition Association of South Africa (SAUPEC/RobMech/PRASA)*, 2019, pp. 141–146.

[9] T. B. Mokgonyane, T. J. Sefara, M. J. Manamela, and T. I. Modipa, "The effects of data size on text-independent automatic speaker identification system," *2019 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)*, 2019.

[10] Statistics South Africa (STATS SA), Census 2011, [Online]. Available: http://www.statssa.gov.za/publications/Report-03-01-78/Report-03-01-782011.pdf. [Accessed 25 August 2018].

[11] D. Aha and D. Kibler, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37–66, 1991.

[12] J. Kacur, R. Vargic, and P. Mulinka, "Speaker identification by k-nearest neighbors: Application of pca and lda prior to knn," in *2011 18th International Conference on Systems, Signals and Image Processing*, 2011, pp. 1–4.

[13] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[14] F. E. Harrell, "Ordinal logistic regression," in *Regression modeling strategies*. Springer, 2015, pp. 311–325.

[15] M. Katz, M. Schaffoner, E. Andelic, S. E. Kruger, and A. Wendemuth, "Sparse kernel logistic regression using incremental feature selection for text-independent speaker identification," in *2006 IEEE Odyssey - The Speaker and Language Recognition Workshop*, June 2006, pp. 1–6.

[16] J. K. Sahoo and D. Rishi, "Speaker recognition using support vector machines," *International Journal of Electrical, Electronics and Data Communication*, vol. 2, no. 2, pp. 01–04, 2014.

[17] T. B. Mokgonyane, T. J. Sefara, M. J. Manamela, and T. I. Modipa, "Development of a text-independent speaker recognition system for biometric access control," in *Southern Africa Telecommunication Networks and Applications Conference (SATNAC)*, 2018, pp. 128–133.

[18] F. Richardson, D. Reynolds, and N. Dehak, "Deep neural network approaches to speaker and language recognition," *IEEE Signal Processing Letters*, vol. 22, no. 10, pp. 1671–1675, 2015.

[19] Y. Wang and B. Lawlor, "Speaker recognition based on MFCC and BP neural networks," in *2017 28th Irish Signals and Systems Conference (ISSC)*, June 2017, pp. 1–4.

[20] E. Barnard, M. H. Davel, C. v. Heerden, F. d. Wet, and J. Badenhorst, "The NCHLT speech corpus of the South African languages," in *Spoken Language Technologies for Under-Resourced Languages*, 2014.

[21] N. J. De Vries, M. H. Davel, J. Badenhorst, W. D. Basson, F. De Wet, E. Barnard, and A. De Waal, "A smartphone-based ASR data collection tool for under-resourced languages," *Speech communication*, vol. 56, pp. 119–131, 2014.

[22] T. Giannakopoulos, "pyaudioanalysis: An open-source Python library for audio signal analysis," *PloS one*, vol. 10, no. 12, 2015.

[23] T. Mazibuko and D. Mashao, "Feature normalization in SVM speaker verification using telephone speech," in *Southern Africa Telecommunication Networks and Applications Conference (SATNAC)*, Mauritius, September 2007.

[24] F. Pyrtuh, S. Jelil, G. Kachari, and L. Joyprakash Singh, "Comparative evaluation of feature normalization techniques for voice password based speaker verification," in *2013 Fourth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)*, Dec 2013, pp. 1–4.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in Python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[26] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[27] M. D. Zeiler, "Adadelta: An adaptive learning rate method," 2012.

[28] T. Dozat, "Incorporating Nesterov momentum into Adam," in *ICLR*, San Juan, Puerto Rico, May 2016.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.