Gender Prediction by Voice

Max Matthew Camilleri (98302L)

1 Introduction

As humans, it is trivial for us to distinguish the gender of an individual based on there voice. However, for machines this task is not so simple. This project will explore the effectiveness of five machine learning techniques with regards to predicting gender based on twenty two characteristics found in the human voice. The goal of this project is finding the best algorithm for this task.

1.1 Background

Machine learning techniques are a class of algorithms that are able to learn and adapt without following explicit instructions. Depending of the method used to achieve this goal a machine learning algorithm will fall under one of a few main categories. One of these categories is supervised learning.

Supervised Learning algorithms require labeled date to be trained on, and are used for predicting unknowns or to classify objects based on a set of input parameters. To be trained, supervised learning approaches are provided a set of parameters and the correct answer they relate to. In the case of classification, these parameters are used to predict which class the object pertaining to this set of parameters falls in. Depending on the result and confidence of the prediction a set of weights and biases inside the algorithms are tweaked to achieve a better outcome on future examples. Given that this is a classification problem all of the algorithms what will be explored are supervised learning models. These algorithms are:

- Artificial Neural Networks
- Support Vector Machines (SVM)
- Decision Trees
- Logistic Regression
- K-Nearest Neighbor

2 Data Analysis

The data used for the testing of the aforementioned five algorithms was sourced from kaggle [1]. Kaggle is an online community platform which allows users to upload and download datasets. The dataset used for this implementation consists of 3,168 recorded voice samples each broken down into twenty two individual characteristics. For full list see Appendix A. Each recording is labeled as male or female depending on the gender of the speaker.

To prepare the data for training and testing the data was split into a train and a test set, each with 80 and 20 percent of the data respectively. This was done to retain a sizable chunk of data for training and also provide a section of unseen out of sample data to be tested against.

To avoid bias in any direction the dataset consists of equal parts male and female participants (See figure 1). Analysing the data further shows that no null data is found in the dataset.

When having this many characteristics it is important to determine which are most important to the classification model. One way of getting incite on this is by plotting each characteristics respective

[1] https://www.kaggle.com/datasets/primaryobjects/voicegender

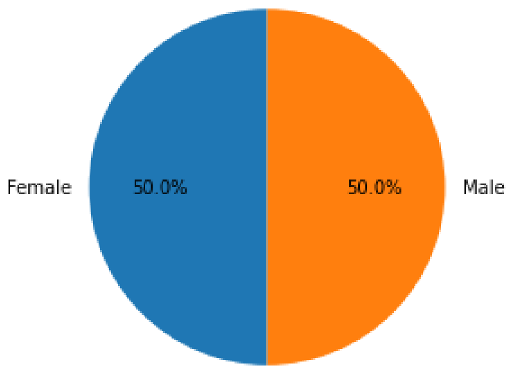


Figure 1: Data Label Distribution

box plot and checking the distribution of the data. Figure 2 depicts those box plots.

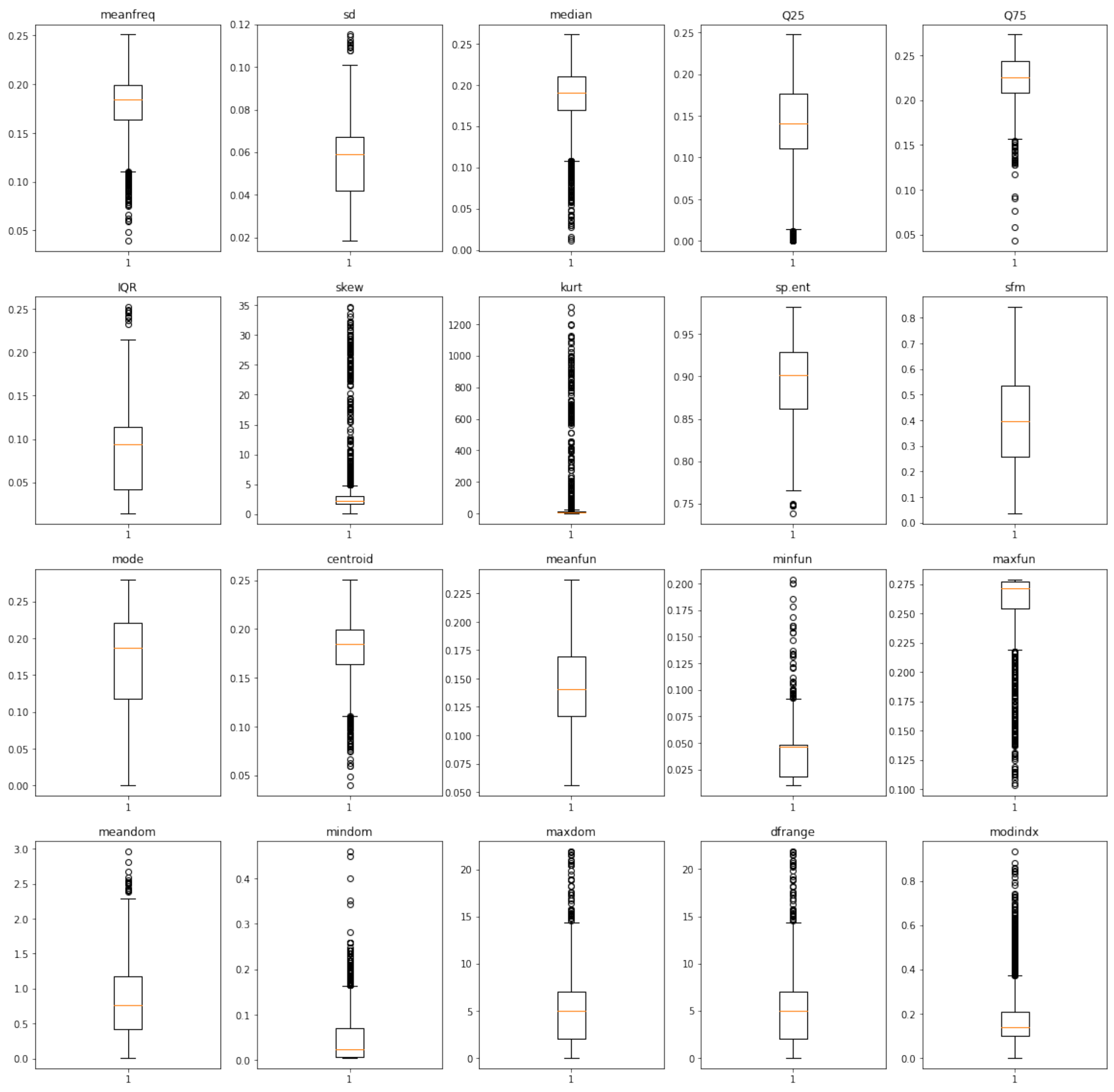


Figure 2: characteristic distributions

The data shows that the variables kurt, skew, maxfun and modindex all are heavily skewed and inconsistent. Based on this it was hypothesised that these characteristics would not be given a high importance by the algorithms or would negatively impact there performance.

3 Methodology

All the following algorithms were implemented in python3.

3.1 Artificial Neural Network

The implementation of the Artificial Neural Network (ANN) was done through the PyTorch library. PyTorch is a open source machine learning framework used for research.

For this binary classification problem, a network was designed consisting three linear layers each with an activation function applied to it. The first two layers used the relu activation function while the final layer had a sigmoid function applied to it. This was done to regulate the output between zero and one. Note that in the dataset a label value of zero referred to a male speaker while a value of one referred to a female speaker.

To combat over-fitting three approaches were applied, the first of which was momentum. This technique works by adding a fraction of the previous update to the next update. By doing this, updates in a different direction would have a smaller impact on the change in state of the algorithm. Similarly, updates in the same direction would be amplified and therefore the model would be trained faster. The second technique applied was an early stop system. The early stop system would stop the training of the network if an improvement is not made in more then the patience permitted by the algorithm. This patience is determined before the training of the network similar to the learning rate. Finally, a dropout layer was also used. Dropout layers make training more difficult with the intention of improving a models generalisation. They work by zeroing out a random set of hidden layer neural units, effectively resetting a layer. This technique acts to break apart co-adapted groups of neural units, i.e. brake apart a set of neurons contributing to over-fitting.

To improve the performance of the network Optuna was used to tune the hyperparameters. Optuna is an automatic hyperparameter optimization framework designed for machine learning. Five parameters were provided to the optuna algorithm to be tuned. These were: learning rate, epochs, momentum value, dropout rate and patience. Each of these parameters where given a range of values to be selected from. The learning rate, dropout rate, and momentum were bound to be a float between 0.0001 and 1, while the epochs and patience were bound between 20 and 100, and 10 and 50 respectively. The tuning resulted in the following results:

Parameter	Value
Learning Rate	0.8601
Dropout Rate	0.1801
Momentum	0.2793
Patience	30
Epochs	80

Table 1: Hyper-tuning Results for Neural Network

Optuna also provides a brake-down of the most important hyperparameters (figure 3).

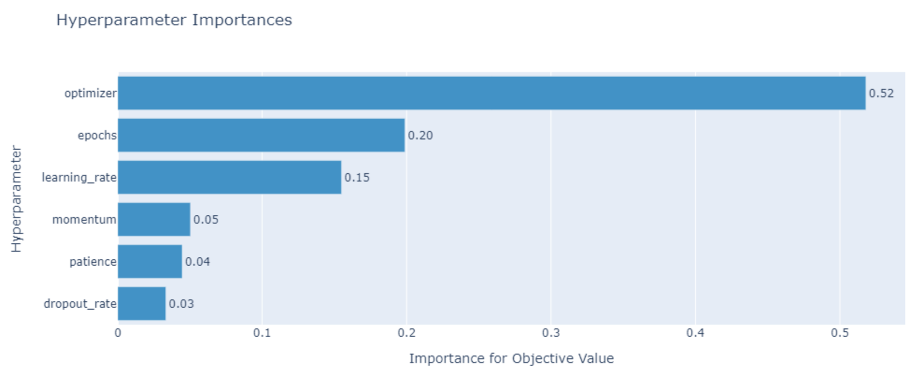
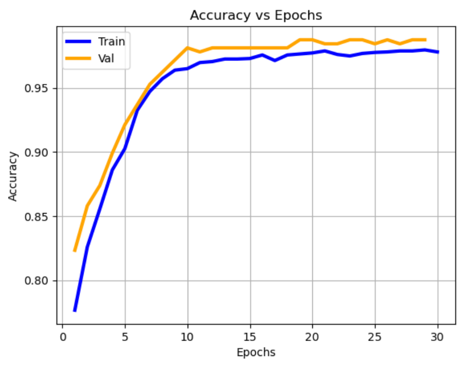
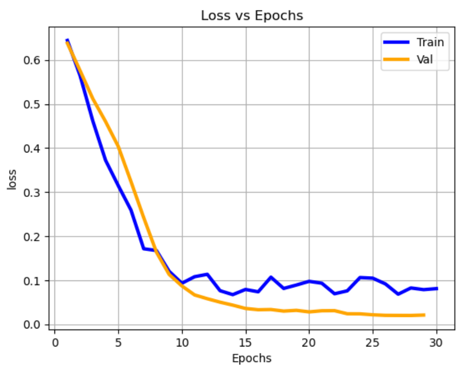


Figure 3: Hyperparameter Importance

Training the algorithm with the tuned parameters resulted in the graphs found in figure 4.



(a) Accuracy for ANN Throughout Training



(b) Loss for ANN Throughout Training

Figure 4: ANN Training Graphs

3.2 Logistic Regression

The implementation for Logistic Regression was done through the scikit-learn library, a simple and efficient tools for predictive data analysis. The library allowed for a quick and easy setup of the algorithm. To improve the algorithms ability to preform accurate predictions hyperparameter optimization was performed on the algorithm.

Two parameters where optimized in this process, the maximum number of iterations the algorithm can perform, and the inverse of regularization strength (C). Regularization is the application of a penalty to increasing the magnitude of parameter values. It is done to reduce over-fitting when training a model. The Smaller this value is the greater the regulation.

To best fit these hyperparameters they were fed into scikit-learns 'GridSearchCV' algorithm along with a number for possible values. The max iterations parameter was provided with four values ranging from 100 to 100,000. For the regularization strength 5 values were randomly chosen from the log-space ranging from minus four to four. The running of this function resulted in the following tuning.

Parameter	Value
c	10000
Max Iterations	10000

Table 2: Hyper-tuning Results for Logistic Regression

3.3 Decision Trees

Similarly to the previous approach, the implementation for Decision Trees was done using the scikit-learn library. Hyperparameter turning was also performed to improve the algorithms performance. Three main parameters were tuned, the first of which was the criterion, a function which measures the quality of feature splits in nodes. Criterion was chosen from two possible approaches, Gini and entropy. Gini refers to Gini impurity, a number between 0 and 0.5 indicating the likelihood of new, random data being misclassified if it were given a random class in accordance to the class distribution. A smaller Gini score implies a better model. The formula for this approach is denoted as:

$$Gini(D) = 1 - \sum_{i=1}^{k} {p_i}^2 \tag{1}$$

Considering a dataset D that contains samples from k classes. The probability of samples belonging to class i at a given node can be denoted as p_i.

Entropy on the other hand is a measure of disorder or uncertainty. For this reason in machine learning we aim to minimize entropy. Mathematically we can denote entropy as:

$$E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i \tag{2}$$

where p_i is the frequentest probability of a class i, and c is the set of all classes.

The second parameter to be tuned was the complexity parameter used for minimal Cost-Complexity pruning (ccp_alpha). To provide the optimization function with a reasonable number of parameters the following was conducted:

The scikit-learn function *'model.cost_complexity_pruning_path(x_train, y_train)'* was applied to a basic model. This function runs the model a number of times progressively increasing the value of alpha and recording the resultant total impurities. These values where extrapolated to produce the following results.From these results the top four performing alpha values where taken and fed into the hyper-tuning algorithm.

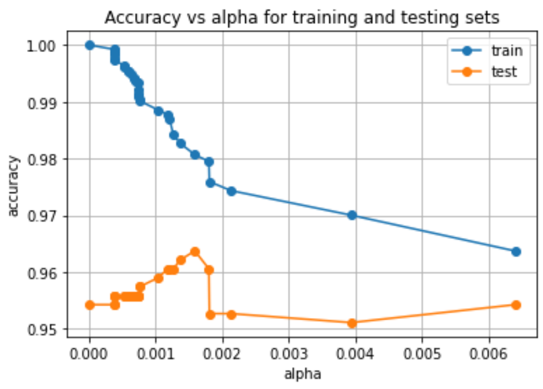


Figure 5: Accuracy vs Alpha

The final parameter to be tuned was the maximum depth of the decision tree. The algorithm was provided with 4 values to test ranging from 50 to 200. Following are the results after tuning.

Parameter	Value
ccp_alpha	0.0011873
Criterion	Gini
Max Depht	100

Table 3: Hyper-tuning Results for Decision Trees

Additionally figure 6 depicts the first three layers of the formed tree.

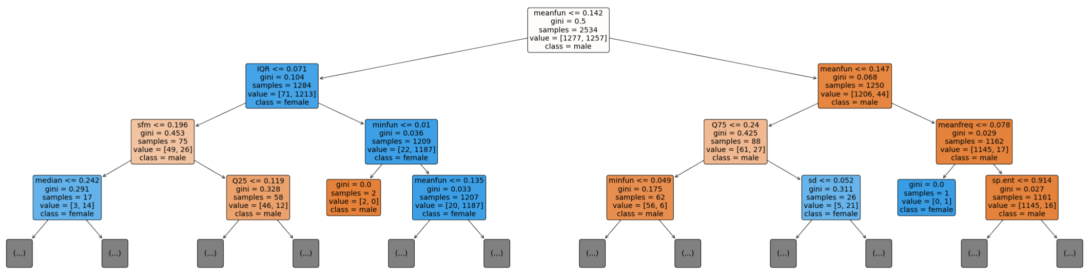


Figure 6: Accuracy vs Alpha

3.4 K-Nearest Neighbor

The implementation and subsequent hyperparameter tuning for K-Nearest Neighbor was also done using the scikit-learn library. Four main parameters where chosen to be tuned to improve the performance of this algorithm, the first of which were the leaf size and number of neighbors. The leaf size limits the number of leaves that can be constructed improving the speed of construction and querying, as well as reducing the memory requirements for the tree. The leaf size parameter was provided with a range spanning 5 to 50, while the number of neighbors was bound between 1 and 100.

The next parameter to be tuned was the metric. The metric is used to compute the distance between nodes. Scikit-learn supports a number of different distance metrics, with the most notable being euclidean distance and Manhattan distance. The euclidean distance is the measure of the true straight line distance between two points in the Euclidean space. Mathematical the Euclidean distance is denoted as:

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{3}$$

where p and q are two points in the Euclidean space, p_i and q_i are Euclidean vectors starting form the origin of the space, and n is the number of points.

The Manhattan distance on the other hand is the distance between two points based on the sum of the absolute differences of their Cartesian coordinates, i.e. the Manhattan distance is the shortest distance between two points following a grid pattern. The Manhattan distance is calculated using the following formula:

$$Dm(p,q) = \sum_{i=1}^{n} D|p_i - q_i| \tag{4}$$

where p and q are two points in the Euclidean space, p_i and q_i are Euclidean vectors starting form the origin of the space, and n is the number of points.

In the scikit-learn package these two parameters are combined into one parameter called 'minkowski'. To decide which algorithm will be used a separate variable p is provided. Setting the value of p to one would result in distance being calculated thought the Euclidean distance, while a value of two would use the Manhattan distance. P was provided to the tuning algorithm and given possible values of one and two. Following are the results of the tuning process.

Parameter	Value
Leaf Size	5
P	1
Metric	Minkowski (Euclidean)
Num. Neighbors	1

Table 4: Hyper-tuning Results for Decision Trees

3.5 Support Vector Machines

Finally, Support Vector Machines (SVM) were also implemented and tuned using scikit-learn. For tuning, three parameters were considered. The first parameter was the inverse of regularization strength (C). This is the same parameter as mentioned in section 3.2, Logistic Regression. This parameter was provided with five values withing the log space ranging from 10^{-4} to 10^{4}. In addition to this the algorithm was given a choice of two kernels, linear and RBF. A linear kernel is one which whose transformation changes the data separation linearly. This results in a series of straight lines which separate the space into different groups. A linear kernel can be defined as:

$$K = (\overrightarrow{X}_i \cdot \overrightarrow{X}_j + 1)^n \tag{5}$$

On the other hand, an RBF kernel (sort for Radial Bases Function) is a complicated combination of multiple polynomial kernels of different degrees. It uses this to project non-linearly separable data into a higher dimensional space, and therefore create an accurate bounding box. Mathematical this is depicted as:

$$K(X_1, X_2) = exp(\frac{||X_1 - X_2||^2}{2\sigma^2}) \tag{6}$$

In addiction to this a third parameter was passed for tuning, this was γ. Gamma refers to the scaling factor and is only applicable to the RBF kernel. For tuning this parameter was provided with four values ranging from 0.0001 to 1.

The results of tuning were as follows.

Parameter	Value
C	100
Kernel	Linear
Gamma	1

Table 5: Hyper-tuning Results for Support Vector Machine

Note that given the selected kernel was Linear the gamma value has no effect.

4 Evaluation

After tuning each of the algorithms performance will be compared to determine the best performer.

4.1 F-Scores

One of the most prominent methods of evaluating the performance of an algorithm is F-score. F-score is a measure of a models accuracy on a dataset. It combines the precision and recall of the model and is defined as the harmonic mean between them. Following is The formula used to computer the F-score.

$$F = (1 + \beta^2)\frac{precision.recall}{precision + recall} \tag{7}$$

Where β is the degree of weighting applied to the model.

The most common form of F-score is the F1-score, where the value of β is 1. The value of F-scores is bound between 0 and 1, were 1 denotes a perfect score.

Precision is the ratio of correctly predicted true positive (TP) results and the total number of predicted positive results $(TP + FP)$. On the other hand, recall is the ratio of correctly predicted true positives (TP) and the total number of positive cases evaluated $(TP + FN)$

For each of the previously mentioned algorithms there respective recall and precision were used to calculate the F1-score based on the test data. These results are depicted in table 6.

Algorithm	Male Precision	Female Precision	Male Recall	Female Recall	F1-score
Artificial Neural Network	0.988	0.974	0.976	0.981	0.981
Logistic Regression	0.960	0.973	0.969	0.965	0.967
Decision Tree	0.954	0.967	0.969	0.952	0.961
K-Nearest Neighbor	0.791	0.774	0.766	0.799	0.782
Support Vector Machine	0.973	0.943	0.939	0.975	0.957

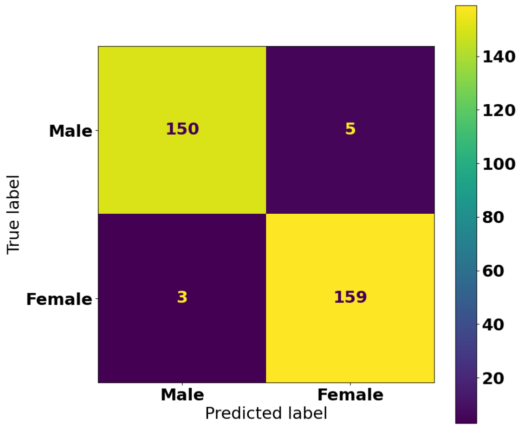
Table 6: Classification Report

Converting the F1-scores to a percentage accuracy yields the following results:

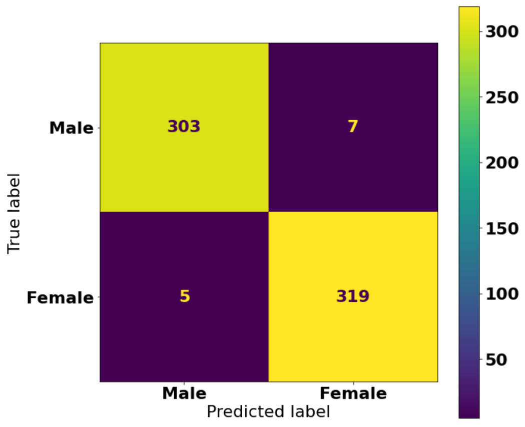
In addition to this each algorithm had a confusion matrix produced. A confusion matrix is an easy way of depicting the prediction accuracy of an algorithm. It consists of a two by two grid with each tile representing a correct or incorrect prediction and the corresponding labels. Below are the confusion matrices for each algorithm.

Algorithm	Accuracy (%)
Artificial Neural Network	98.1
Logistic Regression	96.0
Decision Tree	95.4
K-Nearest Neighbor	79.1
Support Vector Machine	97.3

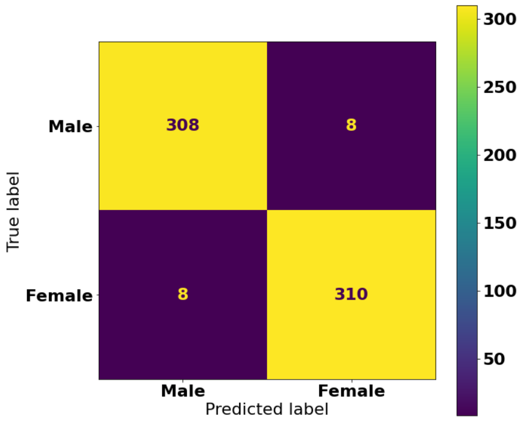
Table 7: Accuracy



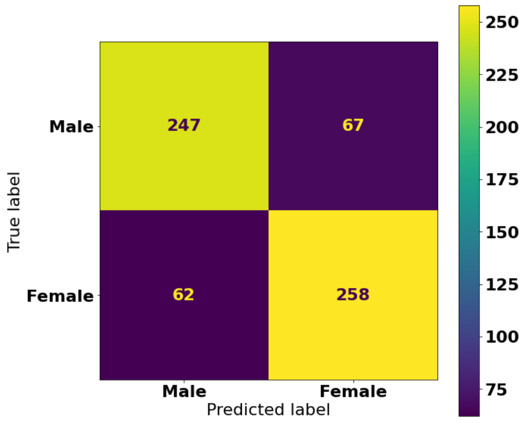
(a) Artificial Neural Network



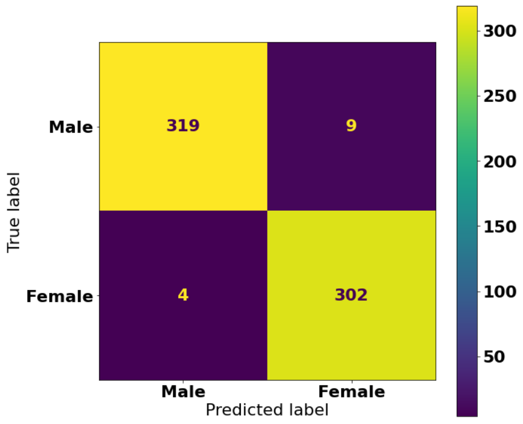
(b) Logistic Regression



(c) Decision Trees



(d) K-Nearest Neighbor



(e) Support Vector Machines

Figure 7: Confusion Matrices

Based on the results at this point one would conclude that the Artificial Neural Network approach is the best algorithm as it has the highest precision, recall and F1-scores across all categories.

4.2 Training Times

In addition to the performance of each algorithm, there training times were also recorded. The results of this can be seen in table 8.

Algorithm	Training Time (s)
Artificial Neural Network	10.549s
Logistic Regression	1.256s
Decision Tree	0.033s
K-Nearest Neighbor	0.006s
Support Vector Machine	17.232s

Table 8: Training Times

Decision Trees and KNN both trained in negligible time with logistic regression, ANNs and SVM following after. The longest training time belonged to SVMs. Note that given the scope these algorithms are being applied to, it is likely that they would only have to be trained once and deployed, making all of the algorithms training times feasible.

4.3 Feature Importance

When it comes to training machine learning models the quality of the input data has a big impact on the output of the model. This section will look into the importance of input parameters for the different algorithms. Support vector machines, neural networks, decision trees and logistic regression all have an internal mechanism of ranking there perceived parameter importance. For graphs depicting this check Appendix B.

As hypothesised in section 2, the variables kurt and skew were given a very low importance by all the tested algorithms. The modindex variable was also given a low importance in the decision tree and support vector machine algorithms. However, the relative importance of this variable in logistic regression was around the median value. Similarly, the maxfun variable was favoured more than expected by the logistic regression and SVM algorithms, with decision trees placing it at the bottom of the list.

The K-Nearest-Neighbor (KNN) algorithm considers all input parameters with equal importance, and therefore is highly infused by pattern-less data. To attempt to improve the subpar performance of KNN, the algorithm was rerun with only the top three variables from each of the tested algorithms. There variables were, meanfun, IQR, Q25 and minfun. Following is the resultant classification report.

Algorithm	Male Precision	Female Precision	Male Recall	Female Recall	F1-score
K-Nearest Neighbor	0.968	0.969	0.968	0.969	0.968

Table 9: Classification Report for KNN

As the report shows, the changes to the KNN increased the accuracy from 79.1% to 96.8%, putting it on par with the other algorithms. In light of this improvement the same test was ran for the other four algorithms although it is hypothesized that the effects will be minimal to none:

Algorithm	Male Precision	Female Precision	Male Recall	Female Recall	F1-score
Artificial Neural Network	0.956	0.987	0.987	0.957	0.972
Logistic Regression	0.978	0.962	0.963	0.977	0.972
Decision Tree	0.966	0.958	0.960	0.964	0.962
Support Vector Machine	0.985	0.963	0.968	0.983	0.975

Table 10: Updated Classification Reports

As previously mentioned, each of these algorithms have a method of ranking feature importance, and therefore there improvements were not as drastic. Of these four algorithms logistic regression found the greatest improvement with the reduced data. Going back to the hypothesised problematic variables in section 2, the maxfun parameter is given a high importance by logistic regression (Appendix B). The removal of this appears to be a reason for logistic regression having the largest

Algorithm	Acc. Before Feature Importance	Acc. After Feature Importance	% change
Artificial Neural Network	98.1	97.2	-0.9
Logistic Regression	96.0	97.2	1.2
Decision Tree	95.4	96.2	0.8
Support Vector Machine	97.3	97.5	0.2

Table 11: Change in Accuracy

increase in performance. Note that this reduction in data also reduce the run time of the algorithms, making them converge faster.

5 Conclusion

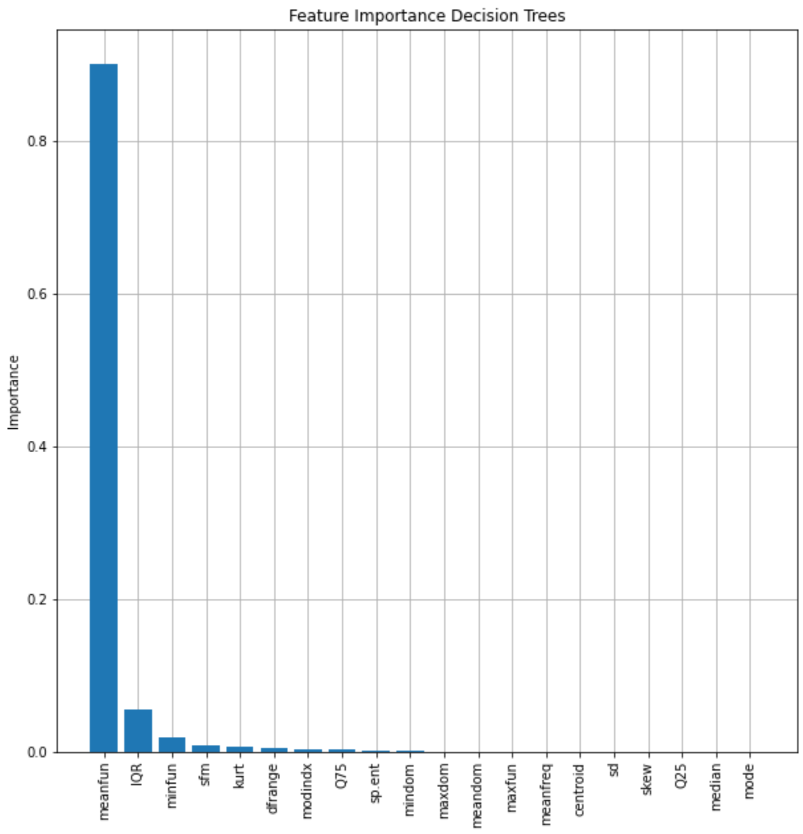
Considering all the results discussed in section 4, one can determine that the best performing algorithm is artificial neural networks. The algorithm scored the highest f1-score (0.981) when presented with all of the data, and the second highest when trained on only the top four variables, differing only by an accuracy of 0.3% from support vector machines. Its drop in performance when removing less prevalent variables shows that those variables where providing some additional information to the ANN that other algorithms were not interpreting. Moreover, given that the prediction time for an ANN is a simply sum of products it is more than fast enough to implement in a real world situation, given the model would only have to be trained once. In a situation where a the model had to be constantly trained and updated in real time the KNN algorithm with reduced features would be the best algorithm, as it still boasts a high accuracy of 96.8% while training in under 0.006 seconds.

6 Appendix A

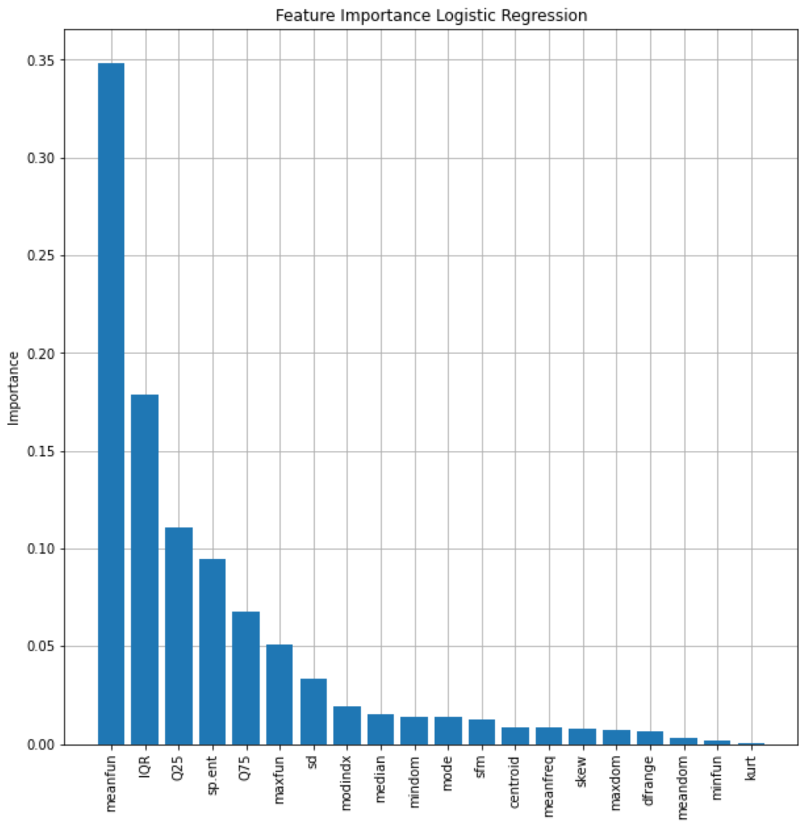
- **meanfreq:** mean frequency (in kHz)
- **sd:** standard deviation of frequency
- **median:** median frequency (in kHz)
- **Q25:** first quantile (in kHz)
- **Q75:** third quantile (in kHz)
- **IQR:** interquantile range (in kHz)
- **skew:** skewness (see note in specprop description[2])
- **kurt:** kurtosis (see note in specprop description[2])
- **sp.ent:** spectral entropy
- **sfm:** spectral flatness
- **mode:** mode frequency
- **centroid:** frequency centroid (see specprop[2])
- **peakf:** peak frequency (frequency with highest energy)
- **meanfun:** average of fundamental frequency measured across acoustic signal
- **minfun:** minimum fundamental frequency measured across acoustic signal
- **maxfun:** maximum fundamental frequency measured across acoustic signal
- **meandom:** average of dominant frequency measured across acoustic signal
- **mindom:** minimum of dominant frequency measured across acoustic signal
- **maxdom:** maximum of dominant frequency measured across acoustic signal
- **dfrange:** range of dominant frequency measured across acoustic signal
- **modeindx:** modulation index. Calculated as the accumulated absolute difference between adjacent measurements of fundamental frequencies divided by the frequency range
- **label:** male or female

[2]https://www.kaggle.com/datasets/primaryobjects/voicegender

7 Appendix B



(a) Feature Importance for Decision Tree algorithm



(b) Feature Importance for Logistic Regression algorithm

Figure 8: Feature Importance

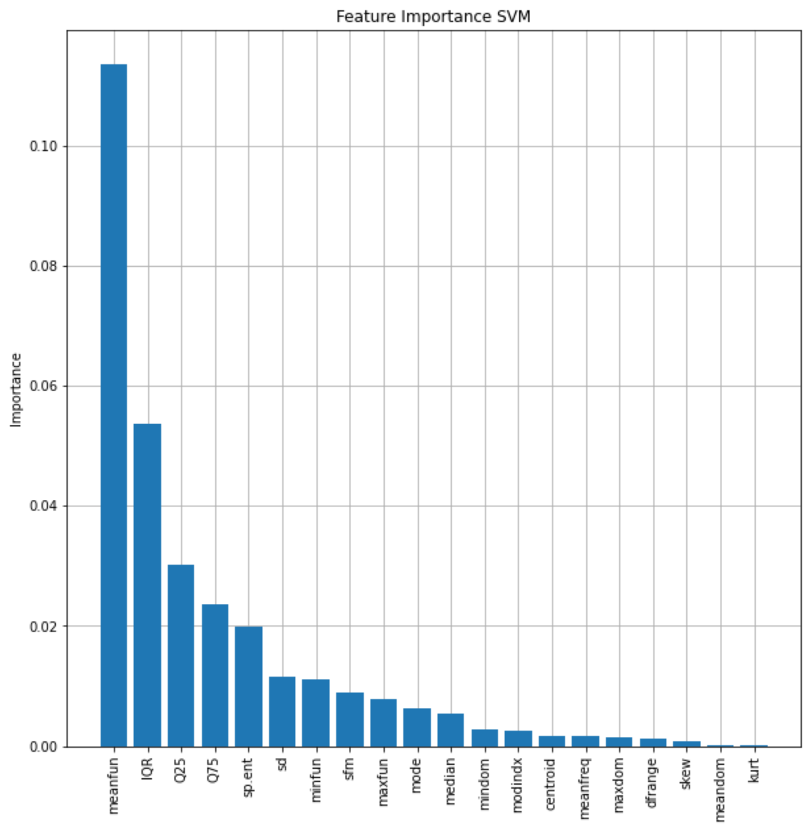


Figure 9: Feature Importance

Item	**Completed (Yes/No/Partial)**
Implemented artificial neural network	Yes
Implemented support vector machine	Yes
Implemented K-nearest neighbour	Yes
Implement decision tree learning	Yes
Implemented logistic regression	Yes
Evaluated artificial neural network	Yes
Evaluated support vector machine	Yes
Evaluated K-nearest neighbour	Yes
Evaluated decision tree learning	Yes
Evaluated logistic regression	Yes
Overall comparison of methods and discussion	Yes

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Declaration

Plagiarism is defined as "the unacknowledged use, as one's own, of work of another person, whether or not such work has been published, and as may be further elaborated in Faculty or University guidelines" (University Assessment Regulations, 2009, Regulation 39 (b)(i), University of Malta).

I the undersigned, declare that the assignment submitted is my work, except where acknowledged and referenced.

I understand that the penalties for committing a breach of the regulations include loss of marks; cancellation of examination results; enforced suspension of studies; or expulsion from the degree programme.

Work submitted without this signed declaration will not be corrected and will be given zero marks.

(N. B. If the assignment is meant to be submitted anonymously, please sign this form and submit it to the Departmental Officer separately from the assignment).

Max Camilleri	[illegible]
Student Name	Signature
______________________	______________________
Student Name	Signature
______________________	______________________
Student Name	Signature
______________________	______________________
Student Name	Signature

ICS 3206	Gender Prediction by Voice
Course Code	Title of work submitted

18/01/2023
Date