ICS3206 Assignment Report

Kian Parnis 0107601L

Contents

1 Introduction

In this project, five machine learning techniques were implemented to predict the sex of a speaker from their voice. In this report, the methodology will be outlined explaining the various design decisions taken for experimentation and evaluation of the methods implemented. This project was coded with python using Google Colab, and each method was implemented utilizing the Sklearn package which provides ready-made implementations for each algorithm. Sklearn also provides implementations for calculating a confusion matrix, a methods classification report, and its accuracy score. This report will be split as follows, in section two the approach taken for handling the data set will be explained. In section three each method's respective hyper-parameter will be discussed as well as the process to tweak them. Section four will cover the evaluation which will describe what metrics have been used and present results obtained. Finally, section five will provide a discussion of the results achieved, comparing the different results to determine which classifier is the right tool for the job.

2 The Dataset and Feature Selection

Before implementing any machine learning algorithm, it is important to go through the necessary steps of proper data collection, processing, exploration, and visualization to ensure that the data is more informative when conducting training. The first step is data collection. This step has been already handled[1] as a dataset was given in CSV format with the first twenty-one rows containing data of different acoustic properties (in real numbers) of each voice and the last column containing the labels to show if a particular propriety row belongs to a male or female. In total, this data set contains 3,168 recorded voice samples split 50% male and 50% female. This data set was uploaded to google colab and inserted as a data frame as follows:

```
df = pd.read_csv('voice.csv') #loading the csv file using panda
```

```
df.head() #checking the head of the csv file
```

kurt	sp.ent	sfm	...	centroid	meanfun	minfun	maxfun	meandom	mindom	maxdom	dfrange	modindx	label
.402906	0.893369	0.491918	...	0.059781	0.084279	0.015702	0.275862	0.007812	0.007812	0.007812	0.000000	0.000000	male
.613855	0.892193	0.513724	...	0.066009	0.107937	0.015826	0.250000	0.009014	0.007812	0.054688	0.046875	0.052632	male
.927705	0.846389	0.478905	...	0.077316	0.098706	0.015656	0.271186	0.007990	0.007812	0.015625	0.007812	0.046512	male
.177296	0.963322	0.727232	...	0.151228	0.088965	0.017798	0.250000	0.201497	0.007812	0.562500	0.554688	0.247119	male
.333713	0.971955	0.783568	...	0.135120	0.106398	0.016931	0.266667	0.712812	0.007812	5.484375	5.476562	0.208274	male

The next step is Data Preprocessing, the data set was checked to see whether it contained any missing or duplicated data as they may give an incorrect view of the statistics of the data. Pythons **.isnull()** and **.duplicated()** functions were used and no apparent missing or duplicated values were found. The next step was to check for outliers within the dataset, this step may provide small improvements when it comes to SVM, ANN, DTC, and LR but for k-means clustering dealing with outliers can have a great effect on the algorithmic performance making it sensitive to outliers. This is due to how k-means updates its cluster centers. It does this by taking an average of all the data points that are closer to each cluster center. Outliers can affect the average being calculated over the whole cluster, pushing the center to be closer to the outliers and giving incorrect results. To check for potential outliers, the distribution was plotted as well as a box plot to visualize the data. Figure 2.1 and 2.2 show the results of these visualizations.

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

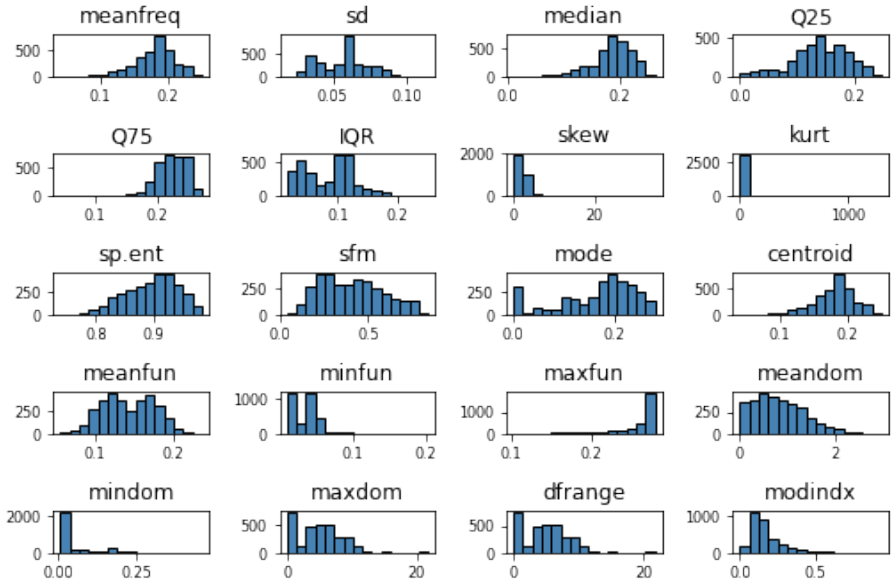


Figure 2.1: Distribution Plot

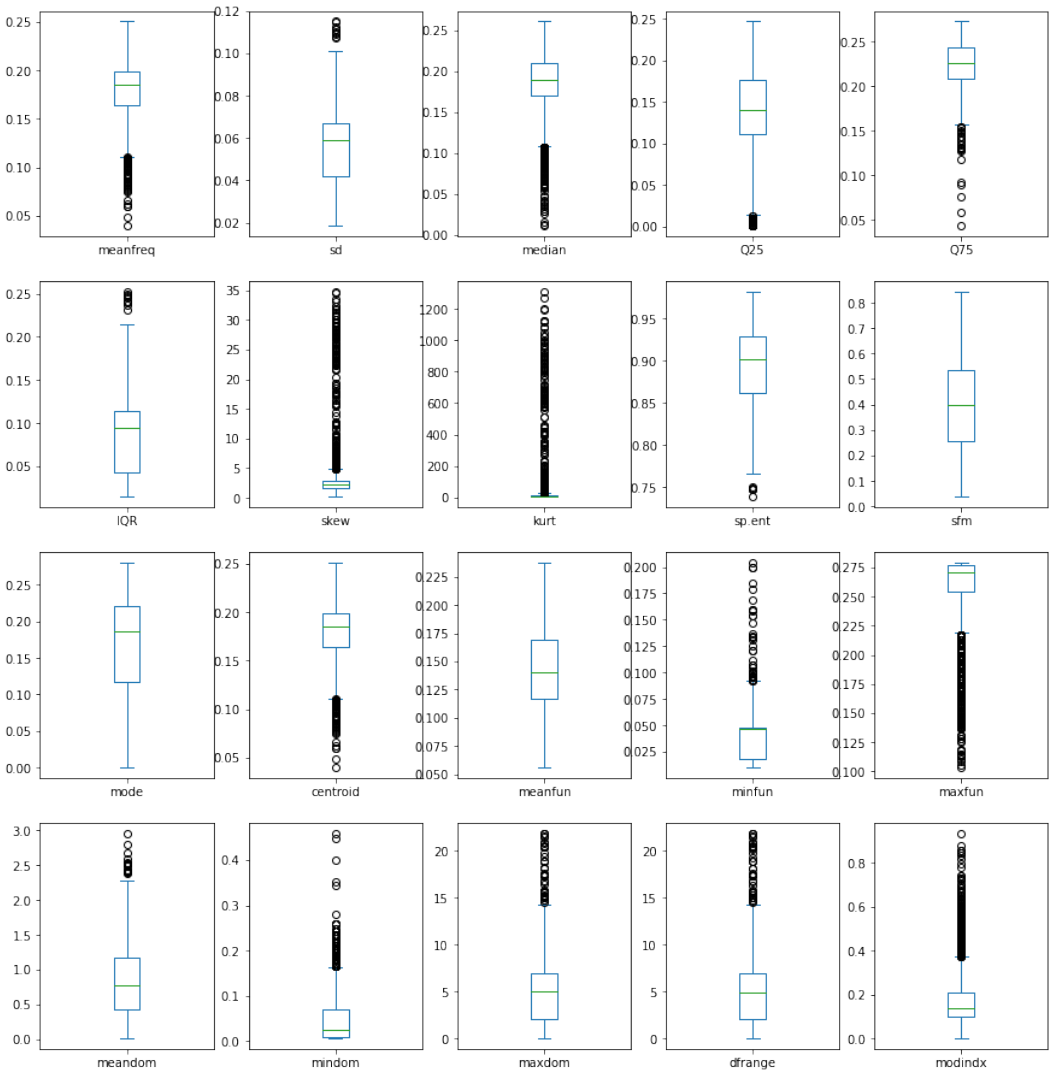


Figure 2.2: Box Plot

From these visualizations, it is shown that some of the features carry a large number of outliers, with the following features containing the most significant amounts: **meanfreq, median, Q75, skew, kurt, centroid, minfun, maxfun, mindom, maxdom, dfrange, modindx**. These features were thus selected to be dropped from the dataset for the number of outliers they contain. As already discussed this will most significantly improve k-means accuracy, but the dimensionality reduction also showed a decrease in training times for the other algorithms. The final step carried out is data normalization to set a common scale and ensure that the features don't have different ranges which may impact accuracy for algorithms that deal with distance. For normalization, the Z-Score was chosen such that the mean of all the values will equate to 0 and the standard deviation is 1. The following is the formula used to calculate a new value:

$$z = \frac{(x - \mu)}{\sigma}$$

- z: New value
- x: Original value
- μ: Mean of data
- σ: Standard deviation of data

3 Methods and Hyper-parameters

With the preprocessing step completed, each individual method can now be implemented and tested. However, training can overfit the training data, this is when the model performs better on the training set than on real-world data. This was tackled by splitting the dataset into a test set and a validation set. Since the dataset is rather large, a slightly bigger split was taken in favor of the test set which was a divide of 33% testing and 67% training. As discussed in the introduction the implementations are all ready-made. Thus, the main focus was on picking and tweaking the most relevant hyper-parameters for each algorithm. This is important because finding appropriate values for hyper-parameters can improve the performance of a model significantly. The reasoning behind which specific hyper-parameters were chosen will be discussed shortly, but when the hyper-parameters were picked, rather than manually tuning, sklearn's GridSearchCV [2] was used to determine the optimal values for a given model. This function takes a parameter grid which is specified as a dictionary in which hyper-parameters are mentioned along with the different values that they can take. This can be either an incremental range of values or in the case of ANN, for example, the different names of activation functions we would want to try out. GridSearchCV then tries all the combinations of values passed and evaluates each model for each combination using cross-validation. The following is a breakdown of which hyper-parameter was chosen for each model, it is important to note that some hyper-parameters have a minor effect on performance and including every parameter will lead to significant growth in searches that would need to be carried out. Thus, only the most prominent hyper-parameters will be picked.

3.1 Decision Tree Classifier

The hyper-parameters for Decision Tree Classifier specify how the tree should be structured along with the criterion of how the tree should be split. The first hyper-parameter specified was the depth of the tree. This is used to limit how deep the tree can be, which increases efficiency. This is set to a max depth of 3, 5, 7, and 9. The second parameter is the number of features to be considered when looking for the best split. This can be done by either taking the log or square root over the number of features. For measuring the quality of the split, this criterion can either be set as Gini or Entropy. Gini measures the frequency at which any element of the data set will be mislabelled when it is randomly labeled. This is denoted as: $GiniIndex = 1 - \sum_j p_j^2$, Where p_j is the probability of class j. Entropy measures the expected value of the information in a message. This is denoted as $Entropy = -\sum_j p_j \log_2 p_j$. Whereas before, p_j is the probability of class j. The final hyper-parameter chosen is for cost-complexity pruning. This will prune the tree down to a smaller subtree which is smaller than the value of CPP chosen. Pruning is done to decrease the size of the tree which also increases efficiency. CPP was set to 0.001, 0.01, and 0.1.

3.2 Logistic Regression

For Logistical Regression, our most notable hyper-parameter is what algorithm is used for our optimization problem. These are: newton-cg, lbfgs, and, liblinear. Newton-cg is a newton approach, it utilizes a Hessian matrix which makes it precise. However, due to second derivatives being computed, it is noted that it is also slow for large datasets. With gradient evaluations, lbfgs (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) approximates the second derivative matrix updates. It conserves memory by just storing the most recent updates. With large data sets, it isn't particularly quick. Lastly,

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`

liblinear uses a coordinate descent algorithm. The foundation of coordinate descent is the looped solving of univariate optimization problems to minimize a multivariate function. In other words, it advances in one direction at a time toward the minimum. The other hyper-parameters tweaked is the C parameter, which performs similarly to the C parameter in SVM with smaller values that specify stronger regularization. Different size extremes are to be tested therefore C is set to 100, 10, 1.0, 0.1, 0.01. Finally, the last parameter tweaked is iterations, this defines the maximum number of iterations taken for the solver to converge. Iterations were set to 100, 1000, and 10000.

3.3 Support Vector Machine

Since we are dealing with two labels for Support Vector Machines, multi-class classification for one vs one or one vs many doesn't need to be passed as a hyper-parameter. Instead, the most prominent parameters are what Kernels will be used, the size of gamma, and our soft margin C. In SVM's a soft margin is used to prevent over-fitting due to the miss-classification of our data, this would be the width of the street of our hyperplane. Different size extremes are to be tested therefore C is set to 50, 10, 1.0, 0.1, 0.01. Kernels are some functions that help transform one value from one space to another. SVM's typically create a hyperplane that separates data points into classes, this may create high bias since the linear regression may not be able to model this data. Thus, different Kernels are used which may lead to better mappings. Four kernels were tested: Linear, RBF, Polynomial, and Sigmoid. These different kernels are shown in Figure 3.3.1

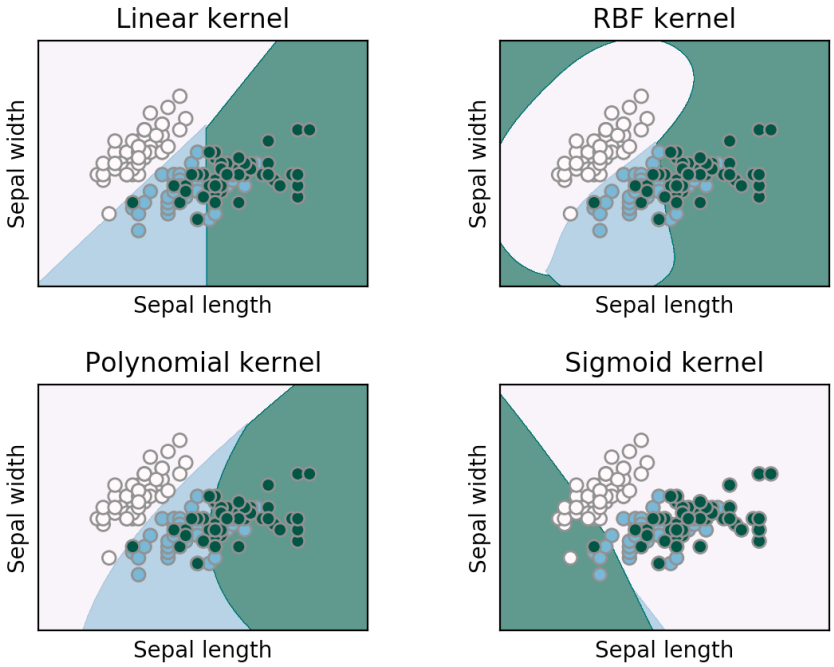


Figure 3.5.1: Kernel Functions

Finally, the parameter gamma is the kernel coefficient for rbf, poly, and sigmoid. This is a nob that decides if the boundary should be smooth or rough. This value was set to scale which decided the value using the following equation:

$$gamma = \frac{1}{Number\ of\ features * X.var()}$$

.

3.4 K-Means Clustering

In K-Means Clustering the most important hyper-parameter is K, which is the number of clusters we want to classify. In this case, the amount of groups we want to cluster is two, that is a cluster for males and a cluster for females. With K chosen, the two hyper-parameters available to us are which variation of the algorithm to use and the number of iterations. The two algorithms are the classical Lloyd algorithm and the Elkan variant which utilizes the triangle inequality to accelerate the process. The number of iterations is set to 300, 500, 600, 800, 1000, and 2000.

3.5 Artificial Neural Networks

For artificial neural networks (ANN), the shape of the ANN was left at a default of two layers with a hundred neurons in each layer, along with this the default optimizer was left as the Adam Optimizer since it works pretty well on relatively large

datasets. The hyper-parameters that were tuned were the rate at which the learning rate declined, the activation function, and an early stopping parameter. The learning rate determines the step size at each iteration while moving towards a minimum loss function. The two parameters picked were constant, in which the learning rate remains the same throughout learning, and adaptive where the learning rate would decrease if the training loss fails to decrease. The two chosen activation functions were Relu and TanH. These activation functions can be seen in Figures 3.5.1 and 3.5.2 with ReLu specifying the output in a range of 0 to infinity, while TanH specifies the output from 1 to -1. These are the two most popular activation functions that are chosen to provide our non-linearity.

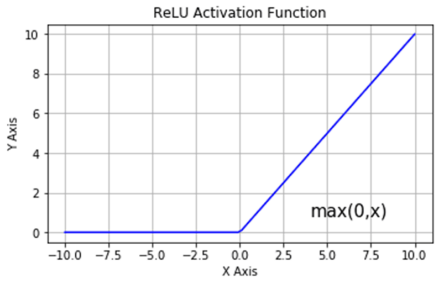


Figure 3.5.1: ReLU activation function

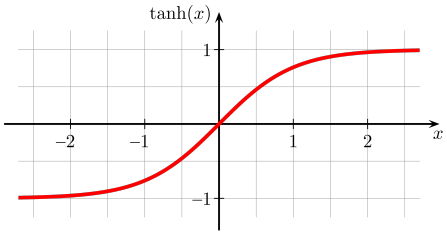


Figure 3.5.2: TanH activation function

Finally, the maximum iterations that the Adam iterates until convergence is set to 2000, this is set to a high value because it was also paired with early stopping. Early Stopping is used to terminate training when the validation score is not improving, which may lead to over-fitting. Early stopping requires a hyper-parameter for how many iterations are needed without the score improving before terminating. This is set to 50, 100, 150, and 200 iterations.

4 Evaluation

4.1 Metrics

The metrics used to compare the performance of each model were evaluated with the use of the model's accuracy score and confusion matrices. Accuracy is the most common evaluation metric used to measure the performance of a machine learning model. It represents the proportion of correct predictions made by the model. In other words, it is the number of times the model correctly predicted the outcome divided by the total number of predictions. For example, if a model correctly predicts 80 out of 100 instances, the accuracy of the model is 80%. The following is the formula for accuracy:

$$Accuracy = \frac{(Number\ of\ Correct\ Predictions)}{(Total\ Number\ of\ Predictions)}$$

A confusion matrix is a table that is used to define the performance of a classification algorithm. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa). The following is the table used for our binary classification problem:

		Predicted Labels		
		p	**n**	**total**
Actual Labels	**p′**	True Positive (TP)	False Negative (FN)	P′
	n′	False Positive (FP)	True Negative (TN)	N′
	total	P	N	

- *True Positive* (TP): These are the cases in which the model correctly predicted the positive class.
- *True Negative* (TN): These are the cases in which the model correctly predicted the negative class.
- *False Positive* (FP): These are the cases in which the model predicted the positive class, but the actual class was negative. This is also known as a "Type I Error".
- *False Negative* (FN): These are the cases in which the model predicted the negative class, but the actual class was positive. This is also known as a "Type II Error".

Using these values, we can calculate other important evaluation metrics such as precision, recall, specificity, and F1-score.

- *Precision*: The proportion of true positives among all positive predictions.

$$Precision = \frac{TP}{(TP+FP)}$$

- *Recall*: The proportion of true positives among all actual positive instances.

$$Recall = \frac{TP}{(TP+FN)}$$

- *Specificity*: The proportion of true negatives among all actual negatives instances.

$$Specificity = \frac{TN}{(TN+FP)}$$

- $F1$-*score*: The harmonic mean of precision and recall.

$$F1\text{-}score = 2 * \frac{Precision * Recall}{(Precision + Recall)}$$

4.2 Hyper-Parameters Results

After choosing which hyper-parameters to tweak (this is discussed in section 3), each feature selected is combined in a dictionary together and fed through the GridSeachCV Algorithm. The following are the top three results obtained, based on the accuracy metric (set to 3 s.f.):

Decision Tree Classifier				
Accuracy	ccp alpha	criterion	max depth	maxfeatures
0.968	0.001	gini	9	log2
0.966	0.001	entropy	7	log2
0.964	0.001	entropy	9	sqrt

Logistic Regression			
Accuracy	C Value	Iterations	Solver
0.970	0.1	100	newton-cg
0.970	0.1	100	lbfgs
0.970	0.1	1000	newton-cg

Support Vector Machine (SVM)			
Accuracy	C Value	Gamma	Kernal
0.981	10	scale	rbf
0.980	50	scale	rbf
0.974	0.1	scale	linear

K-Means Clustering			
Accuracy	Algorithm	Iterations	Clusters
0.562	full	600	2
0.550	elkan	800	2
0.546	full	800	2

Artificial Neural Network (ANN)					
Accuracy	Activation	Early Stopping	Learning Rate	Iterations	Iterations no change
0.975	relu	True	constant	2000	200
0.975	relu	True	adaptive	2000	150
0.973	relu	True	constant	2000	150

4.3 Classification Report

With the results obtained from hyper-parameter tuning, the best hyper-parameters were selected and used to train our model. The following are the results obtained using the metrics illustrated in section 4.1:

Table 1: Decision Tree Classifier Results

	female	male	Total
female	506	19	525
male	10	511	521
Total	516	530	1046

	precision	recall	f1-score
female	0.98	0.96	0.97
male	0.96	0.98	0.97
weighted avg	0.97	0.97	0.97

Table 2: Logistic Regression Results

	female	male	Total
female	493	13	506
male	23	517	540
Total	516	530	1046

	precision	recall	f1-score
female	0.96	0.97	0.96
male	0.98	0.96	0.97
weighted avg	0.97	0.97	0.97

Table 3: Support Vector Machine Results

	female	male	Total
female	509	14	523
male	7	516	523
Total	516	530	1046

	precision	recall	f1-score
female	0.99	0.97	0.98
male	0.97	0.99	0.98
weighted avg	0.98	0.98	0.98

Table 4: K-Means Clustering Results

	female	male	Total
female	372	14	386
male	144	516	660
Total	516	530	1046

	precision	recall	f1-score
0	0.72	0.96	0.82
1	0.97	0.78	0.87
weighted avg	0.88	0.85	0.85

Table 5: Artificial Neural Network Results

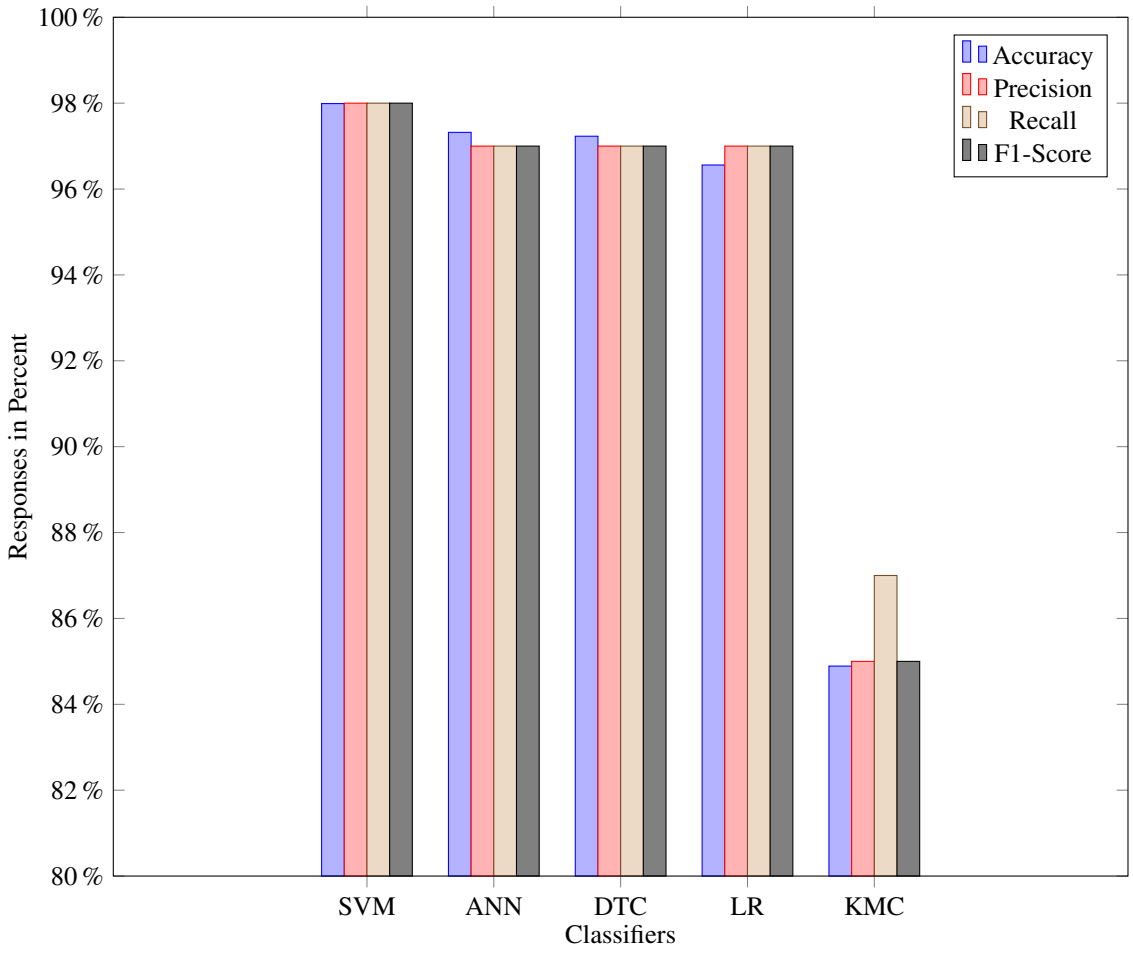
	female	male	Total
female	501	13	514
male	15	517	532
Total	516	530	1046

	precision	recall	f1-score
female	0.97	0.97	0.97
male	0.98	0.97	0.97
weighted avg	0.97	0.97	0.97

4.4 Score Comparison

The weighted averages for Accuracy, Precision, Recall, and F1-Score can then be taken and plotted using a bar chart for each of our different classification algorithms to better visualize our results.

Table 6: Classification Algorithm Results



5 Conclusion

In Table 6 it is shown that the best-performing classifier over our data set is Support Vector Machines closely followed by Artificial Neural Networks, Decision Tree Classifier, and Logistic Regression. K-Means Clustering was the worst-performing classifier achieving $\approx 14\%$ less accuracy than SVM. When comparing training times, once the proper hyper-parameters are found training for all classifiers is carried out in a short amount of time. However, when performing hyper-parameter tuning, training times become much more noticeable with Support Vector Machines (2718s) and Artificial Neural Networks (2356s) taking significantly longer than the other three classifiers: Logistic Regression (120s), Decision Tree Classifier (26.4s), and K-means Clustering (7.14s). It is also noted that without feature selection, k-means clustering was the only classifier that had a significant drop in accuracy down to around $\approx 60\%$.

With this, keeping in mind results in terms of training time vs accuracy, Decision Tree Classifier ends up being the best tool for the job, it achieves $\approx 0.76\%$ less accuracy than the top model (SVM), however, tuning is performed quickly (26s) rather SVM's (2718s). Logistic Regression ends up being a very close second performing with $\approx 1.43\%$ less accuracy than the top model with tuning also performed quickly (120s). As a final remark, in other real-world scenarios, data may not be presented as cleanly as this data set (No missing values found), and with the amount of features columns vs feature attribute rows, dealing with outliers was reasonably simple with the large amount of data collected. Thus, the right tool for the job may vary with some classifiers performing better on other datasets than others. K-Means clustering, however, may only prove fruitful if the dataset is simple and doesn't contain much noise since it's noise sensitive.

6 Statement of completion

Item	Completed (Yes/No/Partial)
Implemented artificial neural network	Yes
Implemented support vector machine	Yes
Implemented k-means clustering	Yes
Implemented decision tree learning	Yes
Implemented logistic regression	Yes
Evaluated artificial neural network	Yes
Evaluated support vector machine	Yes
Evaluated k-means clustering	Yes
Evaluated decision tree learning	Yes
Evaluated logistic regression	Yes
Overall comparison of methods and discussion	Yes

Plagiarism Declaration Form

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 report 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.

Kian Parnis	ICS3206	1/19/2023
Student's full name	**Study-unit code**	**Date of submission**

Title of submitted work: ICS3205-Submission

Student's Signature