# Comparison of the Performance of Different Machine Learning Classification Methods on Voice Gender Recognition based on Voice Sound Wave Components

Arian Tashakkor, *40023494*

**Abstract**

In this article, we will compare the performance of eight different mainstream Machine Learning classification methods on a dataset obtained from extracting the components of the sound waves from the voices recorded from male and female participants with the task of recognizing the gender of the participants solely from the components of their voice sound waves. We will first define the dataset and its components, we will then discuss and give a brief review of the methods used for comparison in this study followed by the results of the comparison and a conclusion.

## I. Introduction

IN this study we aim to compare the performance of several powerful different machine learning algorithms on the same dataset. We will make the same exact comparison before and after feature extraction using PCA[1] to observe how much the classification scores will differ.

It is also important to note that since the dataset was balanced between the two available classes, we have opted to use Mean Accuracy as our measure of comparison.

The dataset itself was obtained from Kaggle [1] and as is also explained there, it contains 20 components extracted from the sound waves of 3168 males and females combined (50% each) accompanied by their respective labels. The classification task at hand is to determine the gender to which each voice sample belongs.

After ensuring that the dataset does not contain missing values, an analysis of the correlation between different pairs of features showed that several pairs were extremely highly correlated. More specifically, 3 pairs of unique pairs of features were more than 95% correlated and therefore, were removed as the subsequent step of preprocessing, and we are left with 17 features with relatively low correlation. Then, dataset was normalized over all of its features to improve the stability and convergence properties of the classifiers later applied to it. Finally, the dataset is partitioned into a 80-20 split for training and testing purposes respectively.

## II. Review of Methods and Experiment Setup

In total, eight different Machine Learning methods were compared on the same dataset before and after feature extraction to see which of them performs best. The methods will be briefly reviewed along with the hyperparameters with which they were used for this study.

### A. Review of Methods

The following is a brief review of the different methods of classification used in this study.

*1) Logistic Regression:* Logistic Regression, simply put, estimates the probability of an event occurring, based on a given dataset of independent variables [2]. More concretely, in Logistic Regression, a logit transformation is applied on the odds - i.e., the probability of success over the probability of failure (encoded by 1 and 0 respectively). This is also commonly known as the 'log odds' and the logistic function is represented by

$$\ln(\frac{p_k}{1-p_k}) = \theta_0 + \theta_1 X_1 + \cdots + \theta_n X_n \tag{1}$$

$$logit(p_k) = \frac{1}{1+e^{-p_k}}. \tag{2}$$

where, $logit(p)$ is the dependent variable and the log odds is assumed to be linear w.r.t. the feature vector $X$. It is easy to show that using (1) and (2), we have

$$logit(p_k) = \ln(\frac{p_k}{1-p_k}) = \theta_0 + \theta_1 X_1 + \cdots + \theta_n X_n. \tag{3}$$

[1]Principal Component Analysis

The parameter $\theta$ which is an $n+1$ dimensional vector is usually estimated through MLE[2]. After obtaining the parameter vector, we are able to estimate the likelihood of success w.r.t. an input vector by plugging the vector into (3). Moreover, classification can be performed by thresholding on the output of the logit transform, usually set to $\geq 0.5$ for 1 and 0 otherwise.

*2) SVM:* SVM[3] is a classification method that aims to maximize the margin of the decision boundary (a hyperplane) in order to be able to generalize as well as possible [3]. The idea is to define the margin as twice the distance between the hyperplane and the closest samples from each class. If the hyperplane is defined as

$$w^T x + b = 0 \tag{4}$$

we can prove that without loss of generality, the margin is the distance between the two hyperplanes

$$\begin{aligned} w^T x + b &= 1 \\ w^T x + b &= -1 \end{aligned} \tag{5}$$

that correspond to the hyperplanes touching the closest samples from each class. With this assumption, it can be proven that the margin is given by $m = \frac{2}{\|w\|}$. In order to maximize this margin a loss function is defined which is minimized through a sequential optimization technique. A special variant of the SVM called 'soft margin SVM' allows samples to enter the band between the two hyperplanes described in (5) but penalizes the loss function for every deviant sample. More concretely, the objective of SVM is

$$\begin{aligned} &minimize \;\; \frac{1}{n}\sum_{i=1}^{n} \xi_i + \lambda \|w\|^2 \\ &s.t. \;\; y_i(w^T x_i + b) \geq 1 - \xi_i \;\; and \;\; \xi_i \geq 0, \;\; for \; all \;\; i \end{aligned} \tag{6}$$

where $\xi_i$ is the smallest non-negative number that satisfies $y_i(w^T x_i + b) \geq 1 - \xi_i$, also called the 'slack variable'. The C-SVM variant of the soft margin SVM was adopted in this study.

*3) Naive Bayes:* A Naive Bayes classifier is a probabilistic machine learning model that is used for a classification task. The crux of the classifier is based on the celebrated Bayes theorem which states that [4]

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}. \tag{7}$$

Where $X$ is an $n$ dimensional feature vector defined as $X = [x_1, x_2, \cdots, x_n]^T$. Therefore, by substituting for $X$ in (7) and expanding using the chain rule we get,

$$P(y|x_1, x_2, \cdots, x_n) = \frac{P(x_1|y)P(x_2|y)\cdots P(x_n|y)P(y)}{P(x_1)P(x_2)\cdots P(x_n)} \tag{8}$$

However, since the denominator of (8) is invariant over the entire dataset, we can say that

$$P(y|x_1, x_2, \cdots, x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y) \tag{9}$$

The Naive Bayes classifier simply uses (9) to predict the most likely class $y$ given predictor variable $X$ as

$$y = \underset{y}{\operatorname{argmax}}\, P(y)\prod_{i=1}^{n} P(x_i|y). \tag{10}$$

*4) Decision Tree:* Decision trees (DTs) are decision support tools that use a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. In machine learning specifically, decision trees usually represent a classification model where starting from a root node that contains the entirety of the dataset, and splitting based on some criteria (gini index [5] or entropy) we eventually arrive at leaves that contain sufficiently pure subsets of the original dataset. Figure 1 shows an example of a decision tree.

There are several ways to train a decision tree. Two of the most popular algorithms are CART [6] and ID3 [7]. Both methods use a splitting goodness criterion to divide the dataset into two partitions at each split in a greedy manner that best improves the purity of the resultant nodes. It has been shown that constructing the optimal decision tree is NP-complete [8], hence why we have to resort to greedy approaches in order to feasibly construct decision trees.

*5) Random Forest:* An extension of DTs and a subset of ensemble machine learning models that aims to improve the generalization and stability of a DT by generating a myriad of DTs, each trained on a subset of the dataset and on a subset of the available features and then producing a final prediction by performing a majority voting on all of the trained DTs [9].

[2]Maximum Likelihood Estimation

[3]Support Vector Machine

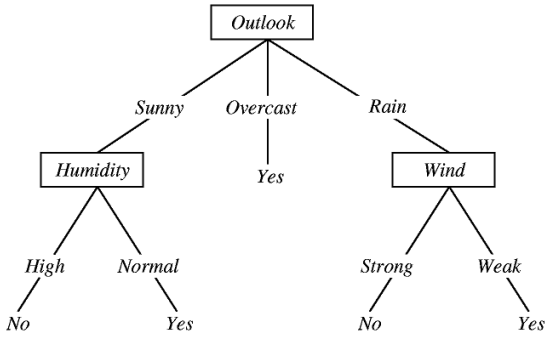


Fig. 1: An example of a decision tree.

*6) k-NN:* Arguably one of the simplest classifiers available, k-NN[4] classifies any given data point on the basis of its closeness to previously seen samples in a predefined neighborhood. More concretely, the $k$ closest samples to any given sample are specified based on a distance metric (usually Minkowski distance of order 2, $d(p,q) = \|p-q\|$) and then the class of the sample in question is determined by a majority voting from these $k$ samples. The main idea here is that if in a fixed neighborhood, a given sample is closest to samples of some class $\omega$, then the sample itself probably also belongs to the same class. [10]

Moreover, there is a trade-off between overfitting and underfitting w.r.t. the hyperparameters $k$. However, a statistical analysis at [11] shows that a generally good optimal value for $k$ is $\sqrt{N}$ where $N$ is the number of available samples.

Although simple, the k-NN classifier has some strong consistency results. As the number of samples ($N$) approaches infinity, the two-class k-NN is guaranteed to yield an error rate no worse than twice the Bayes optimal error rate.

For a general case Cover and Hart prove in [12] an upper bound error rate of

$$R^* \leq R_{kNN} \leq R^*(2 - \frac{MR^*}{M-1}) \tag{11}$$

where $R^*$ is the Bayes optimal error rate, $R_{kNN}$ is the error associated with k-NN classifier and $M$ is the number of classes. For the two-class case, using (11) and as $R^*$ approaches 0, we have

$$R^* \leq R_{kNN} \leq 2R^* \tag{12}$$

*7) AdaBoost:* Originally conceived by Yoav Freund and Robert Schapire in 1995 [13], AdaBoost is a statistical classification meta-algorithm that can be used in conjunction with many other types of learning algorithms to improve performance. The idea is to start from an ensemble of weak classifiers that perform marginally better than a random classifier. Then at every step, by increasing the weight of the misclassified samples by this ensemble, we make the entirety of the classifier more robust to classification errors.

More concretely, a boosted classifier is a classifier of the form

$$F_T(x) = \sum_{t=1}^{T} f_t(x) \tag{13}$$

where each $f_t$ is a weak learner that as input a vector $x$ and returns a value representing the class of that input vector. At each iteration $t$, a weak learner is selected and assigned a weight $\alpha_t$ such that the total training error $E_t$ of the resulting $t$-staged boosting classifier (in (14)) is minimized.

$$E_t = \sum_i E[w_{i,t}F_{t-1}(x_i) + \alpha_t h(x_i)] \tag{14}$$

Here $F_{t-1}(x)$ is the boosted classifier that has been built up to the previous stage of the training and $f_t(x) = \alpha_t h(x)$ is a weak learner that is being considered for addition to the final ensemble of classifiers. Moreover, at each iteration a weight $w_{i,t}$ is assigned to each sample in the training set equal to the current error $E(F_{t-1}(x_i))$ so that over time, the classifier becomes more robust to classification errors.

*8) MLP:* Considered to be one of the oldest classification techniques, MLPs[5] are extremely strong classifiers that are able to create non-convex decision surfaces and have the ability to approximate any function to an arbitrary precision. MLP is a

[4]k-Nearest Neighbors
[5]Multi-Layer Perceptrons

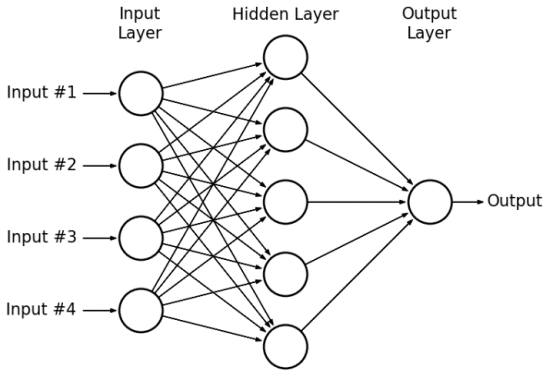


Fig. 2: Structure of an MLP

fully connected class of feedforward ANNs[6]. Figure 2 shows the structure of 2-layer MLP. A matrix of weights is associated with each layer and the output of each layer $l$ is computed as

$$\begin{aligned} a_l &= f(n_l) \\ n_l &= w^T a_{l-1} + b_l \end{aligned} \tag{15}$$

where $f(.)$ is a non-linear activation function (such as $tanh$, $relu$ or $sigmoid$). Through a process of back propagation, the weights of this network are updated by means of SGD[7] which, in essence, calculates the gradient of the cost function that penalizes the network output by the amount it differs from the expected output, w.r.t. to each individual weight in each layer. This gradient is then deducted from the weights of each of the parameters with the hopes of nudging the network in the direction of a minimum.

Due to the convergence stability issues of the initial versions of SGD, several improvements, including momentum SGD [14] which is used here were proposed that improve convergence speed and stability of the learning process.

### *B. Experiment Setup*

The fabled Python machine learning suite, Scikit-learn [15] was used in order to implement the classifiers described above. The implementation python notebook file, 'methods_comparison.ipynb' is readily available and is accompanied by this article.

The hyperparameters with which each of the classifiers were deployed are as follows:

- Logistic Regression:
  - Regularization: $L2$ norm of weights added to the loss function
- C-SVM:
  - C: $1$
  - Kernel: Linear
- Naive Bayes:
  - Variance Smoothing Epsilon Value: $10^{-9}$
- Decision Tree:
  - Splitting Criterion: Gini index
  - Min. Samples to Split: $2$
  - Min. Impurity Decrease: $0$
- Random Forest:
  - Decision Tree Properties: Same as above.
  - Max. Number of Features Used for an Individual Tree: $\sqrt{N}$ where $N$ is the number of features.
  - Number of Decision Trees Built: $100$
- k-NN:
  - Metric: Minkowski distance
  - k: $\sqrt{N}$ where $N$ is the number of available samples.
- AdaBoost:
  - Number of Weak Learners: $200$

[6]Artificial Neural Networks
[7]Stochastic Gradient Descent

  - Weight applied to each Weak Classifier at every Iteration (Learning Rate): 1
- MLP:
  - Two Hidden Layers with 100 and 50 neurons each to ensure non-convex decision surface.
  - Activation of Hidden Layers: $relu$
  - Last Layer Activation: $sigmoid$
  - Optimizer: Variable Learning Rate Momentum SGD
  - Initial Learning Rate: 0.0001
  - Momentum ($\gamma$): 0.9
  - Maximum Iterations: 2000

## III. Experiments and Results

A global random state seed was set so that the experiments are repeatable with the same results presented in this paper. Each of the classifiers discussed in previous sections was trained on the dataset in order to perform the binary classification task of voice gender recognition. The experiments were run twice, once without and once with PCA to reduce the dimensionality of the feature space.

### *A. Before PCA*

Before performing dimensionality reduction, the results of the experiments can be observed in figures 3 and 4. As we can see, Random Forest and AdaBoost have had the best performance on the dataset and Naive Bayes classifier is the worst by a notable margin.

Because Scikit-learn uses Decision Trees as the weak learners for its AdaBoost implementation, the mean accuracy achieved by Random Forest and AdaBoost is similar, although as seen in the confusion matrices, the behavior of the two classifiers is not exactly the same.

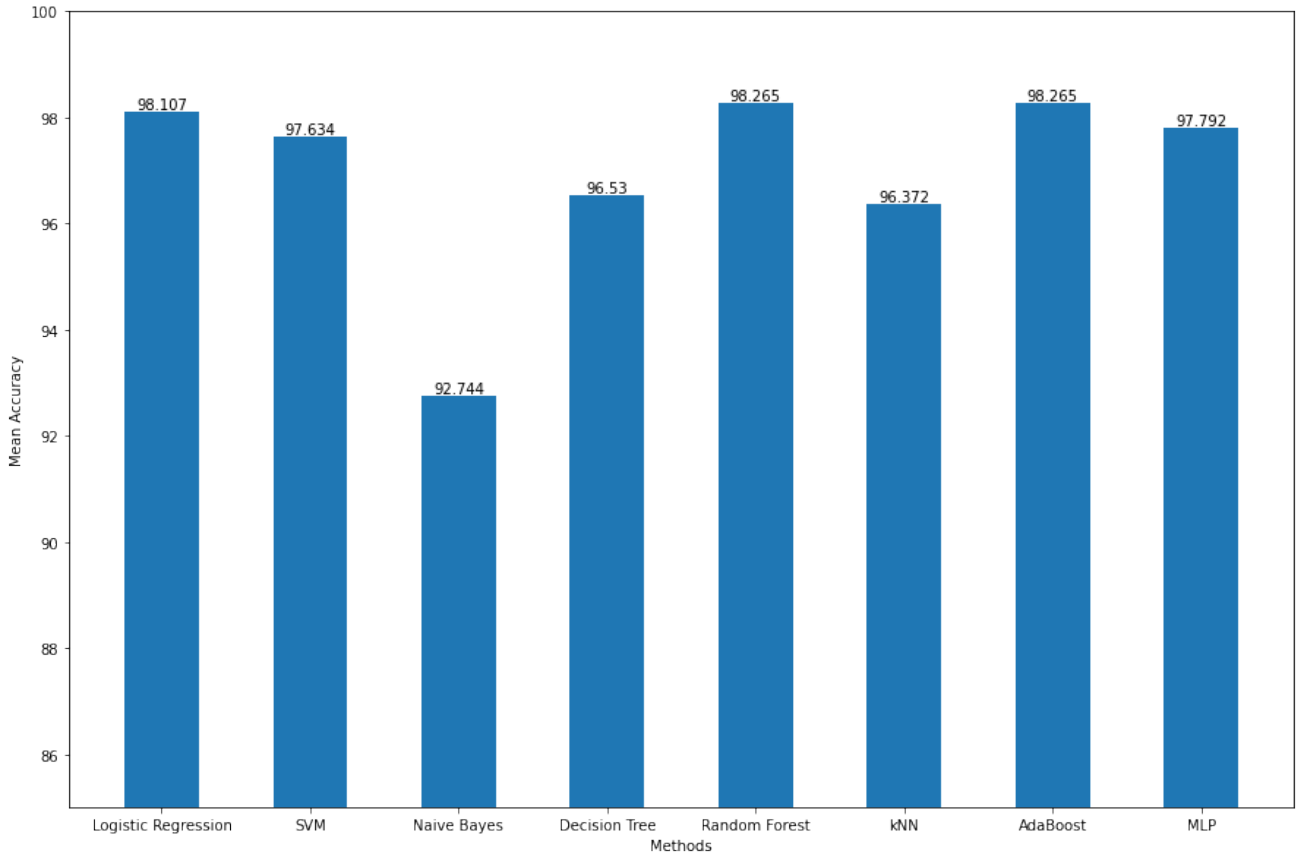


Fig. 3: Confusion Matrices of Each Classifier on the Dataset Before PCA

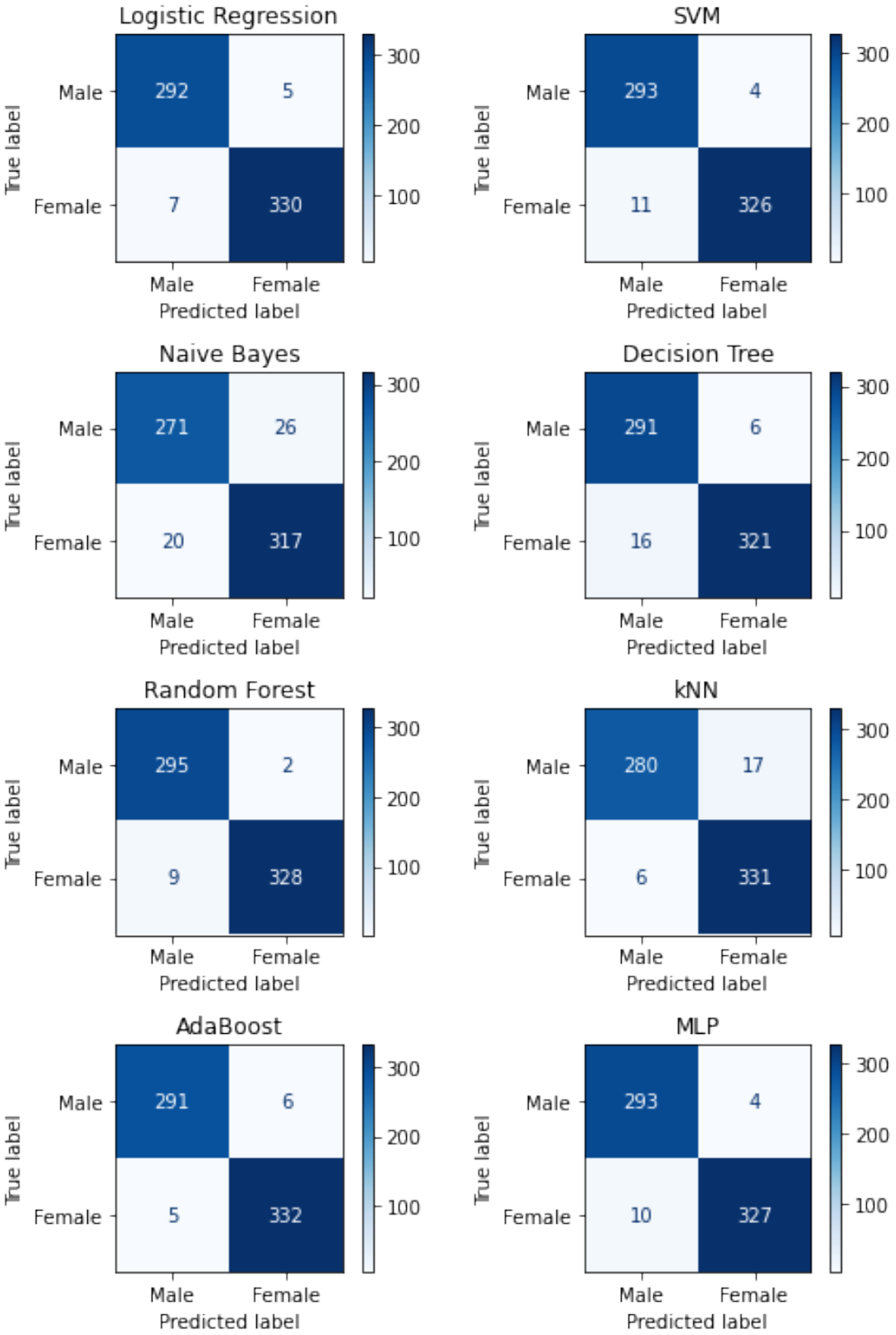


Fig. 4: Confusion Matrices of Each Classifier on the Dataset Before PCA

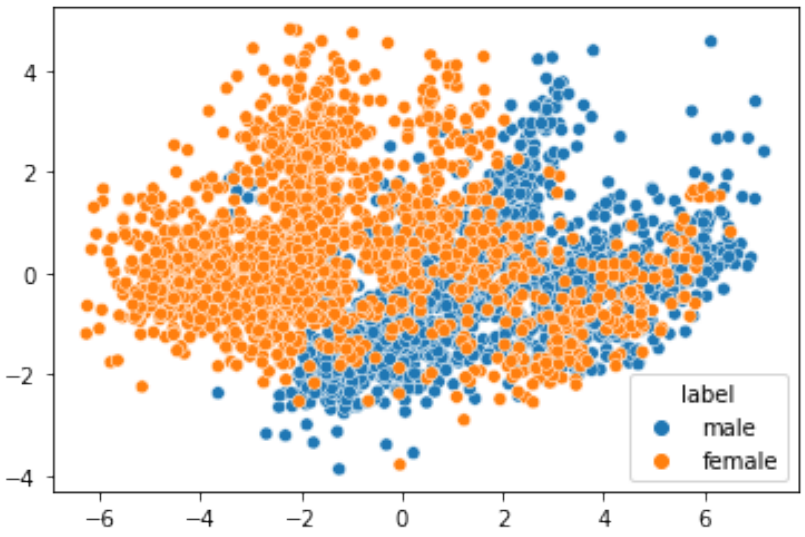


Fig. 5: visualization of the Dataset over the Two Principal Axes

### *B. After PCA*

Before we move on to the experiments, it is not without merit to take a look at the distribution of the classes over the two principal axes to see how separable they are if we only chose to take the two components corresponding to the two biggest eigenvalues. As is evident in figure 5, even the two most informative axes alone are already enough to give a reasonable representation of the dataset. In our experiments however, we have use all the principal axes up to the point where 90% of the information is preserved. This results in a dimensionality reduction from the original 17-d feature space, to only 8 features, while still giving comparable results to the experiments performed before PCA as we can see in figures 6 and 7.

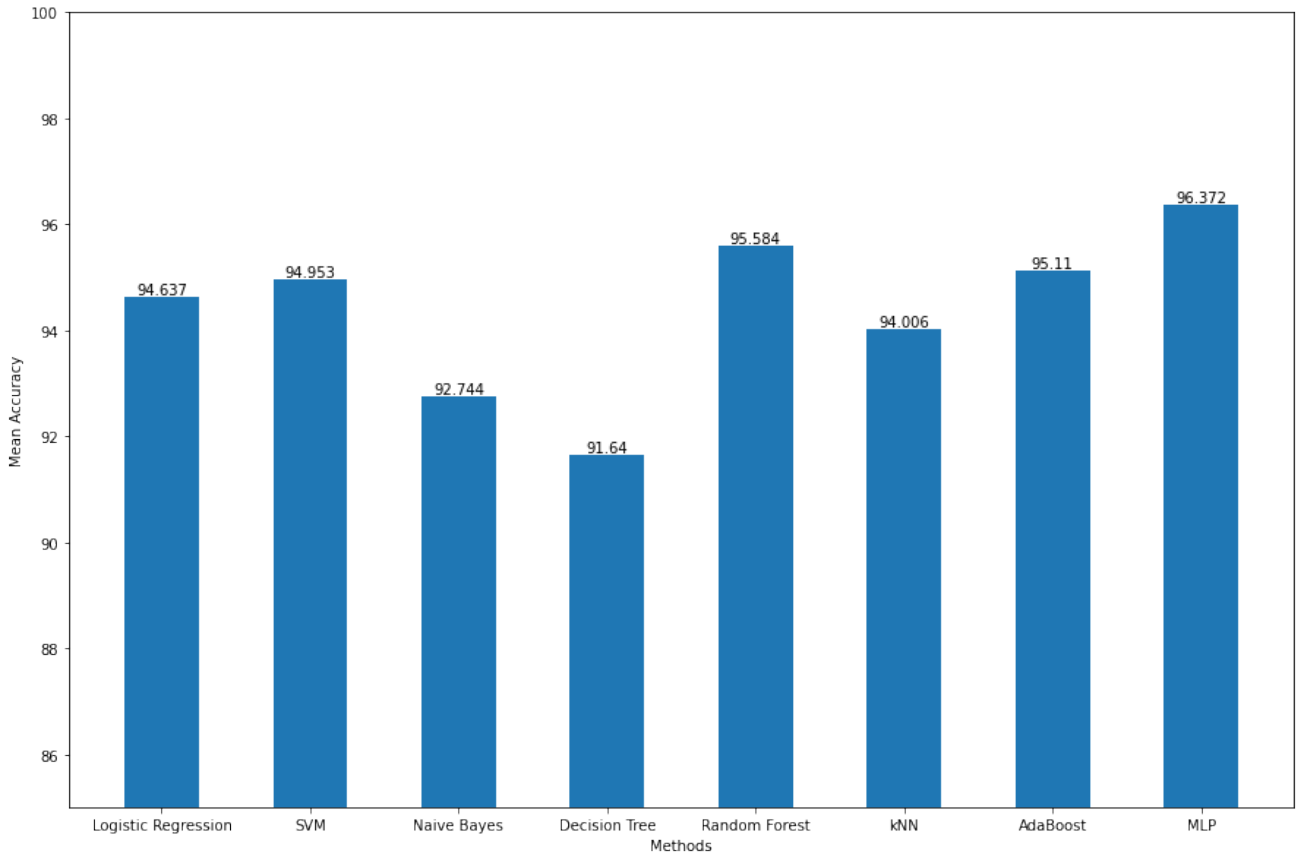


Fig. 6: Confusion Matrices of Each Classifier on the Dataset After PCA

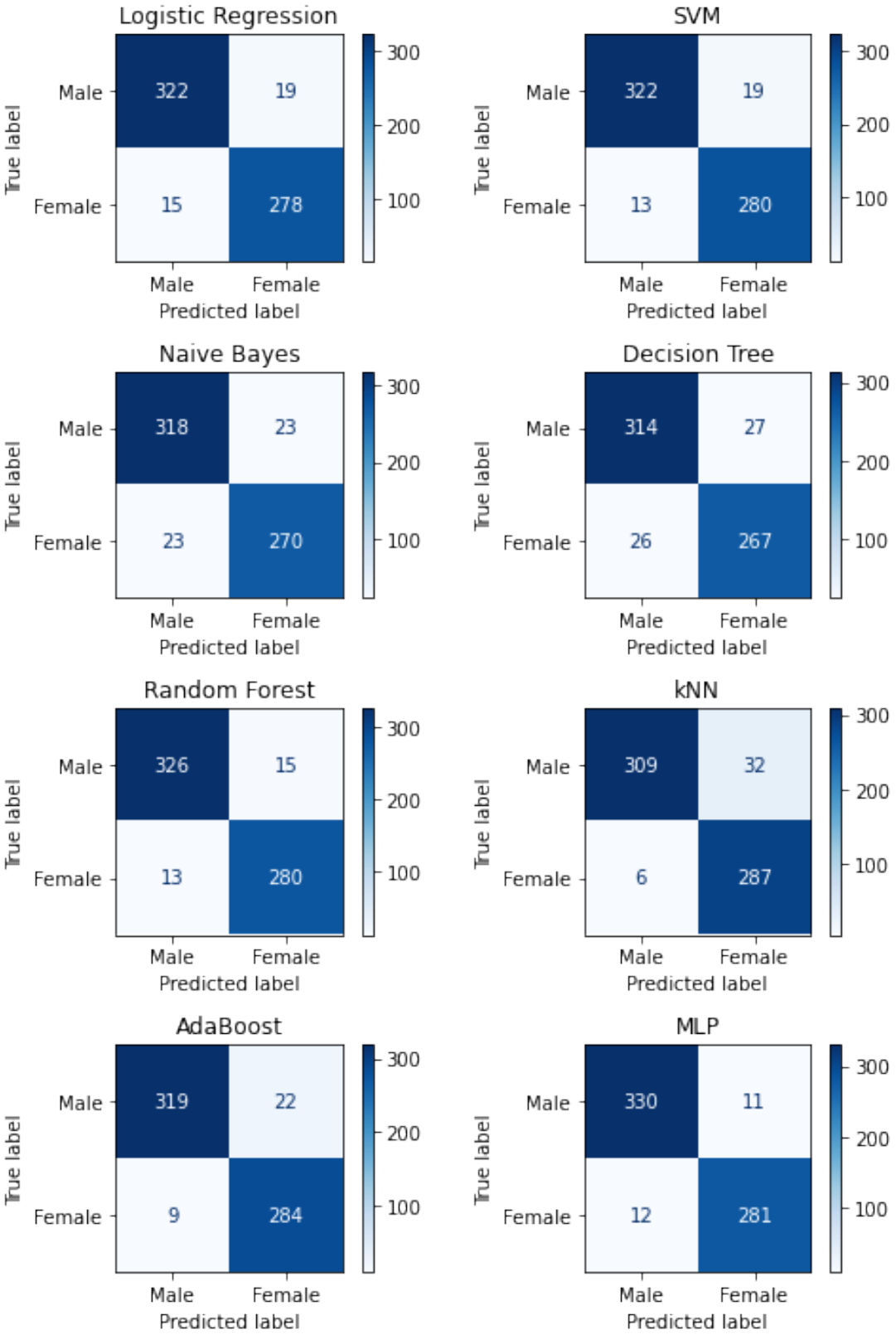


Fig. 7: Confusion Matrices of Each Classifier on the Dataset After PCA

We can see that in the reduced space, MLP performs the best followed by Random Forest, while Decision Tree comes in last, even below Naive Bayes. Interestingly enough, the mean accuracy of the Naive Bayes classifier is affected the least by performing PCA on the dataset. Overall, an average of about $\tilde{3}\%$ mean accuracy was lost going from the complete feature space to the reduced feature space which is negligible against the gain obtained from reducing the feature space down to only 8 features.

It is also important to note that in both comparisons, Random Forest has performed strictly better than a single Decision Tree.

## IV. Conclusion

In this study, we have compared the performance of eight different popular machine learning classification frameworks on a tabular dataset before and after dimensionality reduction. In addition to this comparison, we have also shown the powerful effect that dimensionality reduction can have on some datasets where reducing the feature space to half can result in negligible losses of accuracy.

Overall, the more complex methods like AdaBoost, MLP and Random Forests seem to have had the best classification results over all.

## ACKNOWLEDGMENT

The author would like to thank Dr. Mohammad Reza Ahmadzadeh for their invaluable guidance over the course of the semester.

## REFERENCES

[1] "Gender Recognition by Voice." [Online]. Available: https://www.kaggle.com/datasets/primaryobjects/voicegender
[2] J. Cramer, "The Origins of Logistic Regression," *SSRN Electronic Journal*, 2003. [Online]. Available: http://www.ssrn.com/abstract=360300
[3] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995. [Online]. Available: http://link.springer.com/10.1007/BF00994018
[4] Vikramkumar, V. B, and Trilochan, "Bayes and Naive Bayes Classifier," arXiv, Tech. Rep. arXiv:1404.0933, Apr. 2014, arXiv:1404.0933 [cs] type: article. [Online]. Available: http://arxiv.org/abs/1404.0933
[5] F. A. Farris, "The Gini Index and Measures of Inequality," *The American Mathematical Monthly*, vol. 117, no. 10, pp. 851–864, 2010. [Online]. Available: https://www.jstor.org/stable/10.4169/000298910x523344
[6] R. Lewis, "An introduction to classification and regression tree (cart) analysis," 01 2000.
[7] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, Mar. 1986. [Online]. Available: https://doi.org/10.1007/BF00116251
[8] L. Hyafil and R. L. Rivest, "Constructing optimal binary decision trees is np-complete," *Information Processing Letters*, vol. 5, no. 1, pp. 15–17, 1976. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0020019076900958
[9] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324
[10] E. Fix and J. L. Hodges, "Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties," *International Statistical Review / Revue Internationale de Statistique*, vol. 57, no. 3, p. 238, Dec. 1989. [Online]. Available: https://www.jstor.org/stable/1403797?origin=crossref
[11] microhaus (https://stats.stackexchange.com/users/294515/microhaus), "Why is $k = \sqrt{N}$ a good solution of the number of neighbors to consider?" Cross Validated, uRL:https://stats.stackexchange.com/q/535051 (version: 2021-07-20). [Online]. Available: https://stats.stackexchange.com/q/535051
[12] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.
[13] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Computational Learning Theory*, ser. Lecture Notes in Computer Science, P. Vitányi, Ed. Berlin, Heidelberg: Springer, 1995, pp. 23–37.
[14] S. Ruder, "An overview of gradient descent optimization algorithms," arXiv, Tech. Rep. arXiv:1609.04747, Jun. 2017, arXiv:1609.04747 [cs] type: article. [Online]. Available: http://arxiv.org/abs/1609.04747
[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Muller, J. Nothman, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," arXiv, Tech. Rep. arXiv:1201.0490, Jun. 2018, arXiv:1201.0490 [cs] type: article. [Online]. Available: http://arxiv.org/abs/1201.0490