Applying Data Preprocessing, Clustering and Classification Algorithms to Mushroom, Abalone and Pulsar Datasets

Harry Baines, 35315878

School of Computing and Communications
MSc Data Science, SCC.403 Data Mining

***Abstract*—This paper explores both supervised and unsupervised machine learning techniques, by implementing and evaluating supervised classification models trained on mushroom and abalone datasets, and unsupervised clustering algorithms trained on pulsar data. Before applying these algorithms, the data was preprocessed using traditional data preprocessing algorithms such as normalization and standardization. Principal Component Analysis (PCA) facilitated dimensionality reduction and extraction of new, orthogonal features. Synthetic Minority Over-sampling Technique (SMOTE) was applied to the abalone data to overcome the high-class imbalance problem. Imputation enabled missing values to be estimated for the mushroom data. We utilised the Python programming language to facilitate the implementation of data preprocessing algorithms and evaluation of classification and clustering algorithms. We find that the k-means algorithm performs best when separating data points into clusters, and the logistic regression classifier performs best for predicting the positive class for both abalone and mushroom datasets.**

***Index Terms*—Data Pre-processing, Classification, Clustering.**

I. Introduction

In this paper we apply a selection of unsupervised clustering algorithms to the HTRU2 dataset which describes a sample of pulsar candidates collected during the High Time Resolution Universe Survey. Pulsars are a rare type of highly-magnetised rotating Neutron stars which produce radio emission detectable on Earth. Machine learning tools are being used to automatically label pulsar candidates, with a candidate representing a potential signal detection averaged over many rotations of the pulsar. The dataset consists of 8 features containing 16,259 spurious observations caused by noise, with 1,639 real pulsar examples checked by human annotators [1]. Many machine learning techniques have been applied to the pulsar data, including Neural Networks [2] for prediction and AdaBoost and Gradient Boosting Classifiers for classification [3].

To deal with class imbalance we analyse an abalone dataset consisting of 8 features and a class label indicating if the abalone belongs to age class 19 or not [4]. This dataset has been constructed specifically for handling class imbalance, although in the literature machine learning techniques such as k-Nearest Neighbor and Decision Tree classifiers were applied to a variation of this dataset with a continuous age output variable [5]. Finally, we analyse different techniques for handling missing data with a mushroom dataset consisting of 22 features and a single class label [6].

II. Data Preprocessing

Before applying machine learning algorithms we must perform data preprocessing, a fundamental step in data mining performed before classification and clustering. Our data may need to be re-scaled and missing values or anomalies (outliers) could be present for features in the data. Therefore we must ensure our data is formatted appropriately before proceeding with machine learning.

A. Standardization

Standardization involves re-scaling features in a dataset into a more suitable range to facilitate comparisons between features containing values in different ranges. The meaning and units of different features will vary, therefore it is necessary to standardize them. After standardizing our data all features will have a mean of 0 and a standard deviation of 1. We do this by calculating a z-score for each observation x for all features using the following formula:

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

where x represents an individual data value for a given feature, with μ and σ representing the mean and standard deviation for the feature associated with the data value under consideration respectively (see Appendix V-A for a comparison of selected original and standardized features). From the plots we can see the overall distribution of the data remains constant across plots for the plotted features, however the range of the the standardized data takes a different range.

B. Normalization

Similar to standardization, normalization (also known as Min-Max scaling) transforms our data containing features with different ranges into a more suitable range for feature comparison. This technique becomes useful when performing Principal Component Analysis which aims to maximise variance between features. Having features in different ranges would mean features with a larger range (variance) would contribute to the highest variance (see the following section for further details).

We can use the following formula to normalize a feature to the range [0,1]:

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Similarly to standardization, normalizing features preserves the distribution of the data, however all data points now lie in the range [0,1]. One can also choose to scale features to the range [-1,1] by calculating $2 \times x_{new} - 1$ per feature where x_{new} is calculated as previously described.

C. Principal Component Analysis

Principal Component Analysis (PCA) is a widely used unsupervised dimensionality reduction technique in data mining. Having highly correlated features in a dataset can cause overfitting - PCA aims to reduce the number of correlated variables while retaining as much variability as possible from the original dataset [7]. This technique can also be used as an Exploratory Data Analysis (EDA) technique, to facilitate exploration of relationships between variables and identify patterns in the data.

In many cases, not all features of a dataset contribute well to the generalization capacity of a model, with many features potentially decreasing performance [8]. PCA enables new variables to be constructed which are commonly known as principal components, a set of uncorrelated/orthogonal linear combinations of features present in the original dataset which maximize variance. The principal components are ordered by descending variance, such that the first principal component has the highest variance.

In the following scree plot, we can visualise the percentage of explained variance for each principal component obtained from the pulsar dataset, in order to decide how many components we wish to retain for further analysis:

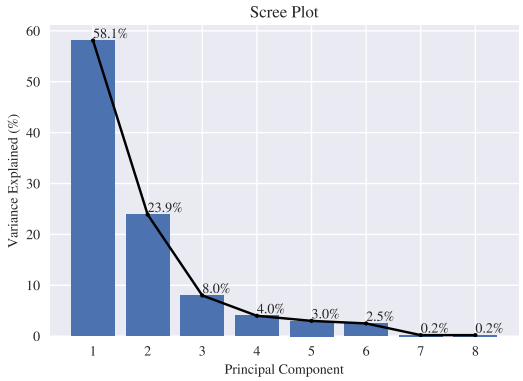

Fig. 1. Scree plot of principal component explained variance

Figure 1 shows 82% of variance is explained by the first 2 principal components, with 90% of variance explained by the first 3 principal components. Should we choose to retain only the first 2 principal components, we can discard the remaining 6 components without losing too much information, as 82% of the variation from the original data is explained by the first 2 components.

Applying PCA to the pulsar data containing 8 dimensions will enable a good characterization of the data to be seen in a 2-dimensional space. Given the points in the new projected space correspond directly to observations in the original data, we can colour the data points according to their class label. Figure 2 shows the result of PCA on the pulsar data:

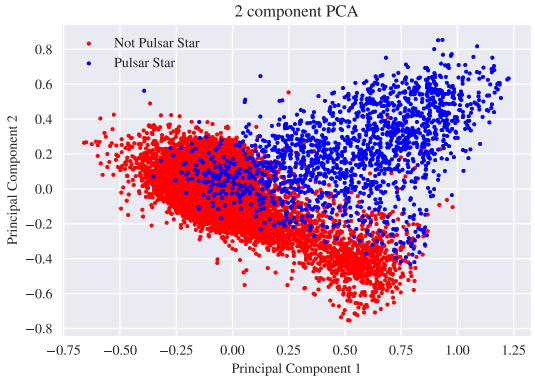

Fig. 2. Plot of 2 principal components with largest variance

From Figure 2 we notice a reasonable separation between two separate clusters, with blue points and red points indicating pulsar and not pulsar data points respectively. We notice many data points corresponding to pulsar observations overlap onto data points labelled as not a pulsar which is likely to result in different results when clustering algorithms have been applied. In Section III we apply clustering algorithms to this result in an attempt to automatically cluster data points in the projected space into 2 distinct clusters. A common distance metric used for clustering algorithms, known as the Euclidean distance, loses its meaning in high-dimensional space. Therefore in order to facilitate the visualisation and identification of distinct clusters we can use our PCA plot of the 2 principal components with the largest variance for clustering analysis.

PCA was also applied to the abalone dataset (see Appendix V-B and Appendix V-C) and the mushroom dataset (see Appendix V-D and Appendix V-E). In particular the mushroom dataset contains 22 features/dimensions, therefore PCA can facilitate dimensionality reduction and visualisation of clusters in the projected PCA space.

D. Anomaly Detection

Anomaly detection is a technique used to identify observations which differ significantly from the other data points. A well-known probabilistic inequality known as Chebyshev's inequality is given by $P(|X - \mu| \geqslant k\sigma) \leqslant \frac{1}{k^2}$, which gives a bound of what percentage of data points fall outside of k standard deviations from the mean [9]. The inequality states that $\sim \frac{8}{9}$ of the data for a given distribution lies within the range $[-3\sigma, 3\sigma]$ if $k = 3$. Data points outside this range ($\sim \frac{1}{9}$) are candidates for potential outliers/anomalies. From the standardized pulsar plot (see Appendix V-A) we notice most of the standardized data points lie within $\pm 3\sigma$ from the mean with some points lying outside this range. We cannot conclude however these points are anomalous as they are merely candidates. Therefore we do not remove these

data points from our analysis although they are taken into consideration.

III. Clustering

In this section we explore clustering algorithms, a set of unsupervised learning techniques which attempt to identify clusters of similar data points without the need for labelled observations. We implement and evaluate k-means and Agglomerative Hierarchical Clustering methods on the pulsar dataset following dimensionality reduction using PCA as previously described. Mean-Shift clustering was also considered as it had the notable advantage of not having to define the number of clusters beforehand like in k-means, however due to it's quadratic time complexity ($O(N^2)$) it became infeasible to cluster the pulsar data points due to a large number of pulsar observations.

A. K-means Clustering

K-means clustering is a widely used iterative clustering algorithm in data science. Initially the number of clusters k is specified. The centroids are initialised by randomly selecting starting positions for the centroid locations. The sum of squared distances between all data points and centroids are calculated, with each data point assigned to the closest centroid (cluster centre). The centroids are re-computed on each iteration for the clusters by calculating the average of all the data points belonging to each cluster. The algorithm iterates until the assignment of data points to clusters is not changing [10].

Given that the pulsar data comprises 8 features, it would be impractical to attempt to visualise the clustered results in 8 dimensions. Therefore we can apply Principal Component Analysis to reduce the dimensionality of the pulsar data into 2 principal components, which are linear combinations of the original features, and retain the largest explained variances of all calculated components (see Figure 2). We apply k-means clustering to this result and we obtain the following plot:

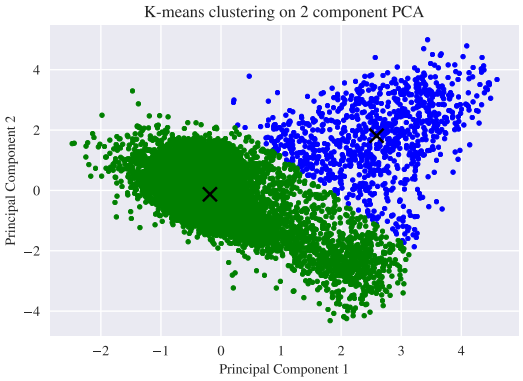

Fig. 3. K-means clustering on 2 principal components with largest variance

From Figure 3 we see 2 separate clusters which correspond to the original PCA result in Figure 2. The k-means algorithm was able to separate the data points into separate clusters with reasonable accuracy, however many of the pulsar observations which overlapped into the not a pulsar category were clustered as not a pulsar. Due to random assignment of the initial cluster centres, the results do vary upon repeated iterations of the k-means algorithm.

B. Agglomerative Hierarchical Clustering

Another clustering algorithm utilised in our analysis is Agglomerative Hierarchical Clustering. This algorithm begins by assigning all N data points to an individual cluster. On each iteration, we continuously merge the closest pair of clusters, based on a calculated distance metric, until only one cluster remains containing all N data points [11]. A symmetric matrix of distance similarities must be calculated prior to clustering - a distance metric such as the Euclidean distance calculated between all pairs of points enables their similarity to be calculated and stored in a distance matrix. Agglomerative clustering relies on the notion of a single linkage algorithm, which takes a distance matrix as input and links pairs of data points close together into binary clusters. The linkage function continuously forms larger clusters until all observations in the original data set are linked together [12]. The observations can then be grouped into a hierarchical cluster tree (visualised as a dendrogram), at which point we decide where to cut the tree which yields the final number of number clusters.

Following Principal Component Analysis on the pulsar dataset, we can apply the Agglomerative Hierarchical Clustering algorithm to automatically cluster the data into distinct clusters without the need for labels. The following plot shows the produced dendrogram:

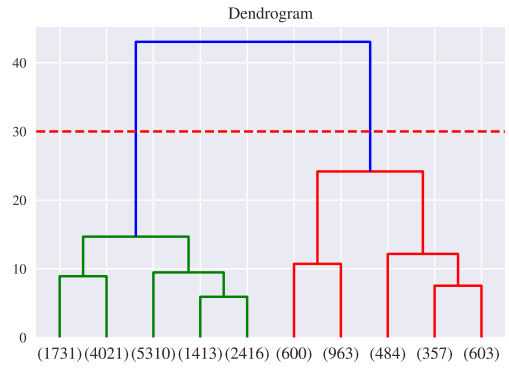

Fig. 4. Pulsar Dendrogram

In Figure 4 we visualise the last 10 clusters of the dendrogram tree from the linkage matrix, where the first level is the one with the highest distance. The blue vertical line has the largest distance of all vertical lines. We can plot a red line which cuts the dendrogram through 2 lines and agrees with the data, that is, we expect two separate clusters with one for pulsars and one for not pulsars. We can conclude the hierarchical clustering algorithm could reasonably separate the pulsar data points into 2 distinct clusters.

IV. Classification

In this section we implement and analyse various supervised classification algorithms which will be applied to the abalone and mushroom datasets. An abalone classification model will enable a prediction to be made on whether an unseen observation belongs to age group 19 or not. A mushroom classification model will enable mushrooms to be classified as poisonous or edible based on categorical measurements.

Due to the wide variety of available classification algorithms each with their own time and memory complexities, the following four classification algorithms were chosen and implemented for the abalone and mushroom datasets:

- **Logistic Regression**: a statistical model that models a binary dependent variable based on one or more explanatory variables
- **Radial Basis Function Neural Network (RBFNN)**: a non-linear classifier which measures the inputs similarity to examples from the training set
- **Shallow Neural Network**: an artificial neural network with one hidden layer
- **Decision Tree**: a tree-like classifier which continuously splits a dataset according to a certain parameter to classify items

The k-Nearest Neighbor (kNN) classifier was also implemented, however given the time complexity of kNN is $O(DN^2)$ (where D is the number of dimensions in the dataset and N is the number of observations) the algorithm proved too time consuming to collect results over many iterations. This was mainly due to pairwise calculations between a new point and all dataset points to find the nearest neighbors.

To summarise the results of our classification models we generate a confusion matrix - a table which provides totals for the number of correctly predicted and incorrectly predicted labels. Positive classes represent a value of 1 with negative classes taking a value of 0. The following values comprise a typical confusion matrix:

- True Positives (**TP**): the number of correctly classified positive classes
- True Negatives (**TN**): the number of correctly classified negative classes
- False Positives (**FP**): the number of incorrectly classified positive classes
- False Negatives (**FN**): the number of incorrectly classified negative classes

In order to evaluate classification performance we calculate various metrics based on the output of our classifier and the true labels. We first calculate our confusion matrix values as previously described. We then proceed to calculate the following metrics:

- **Accuracy** ($\frac{TP+TN}{TP+TN+FP+FN}$): a ratio of correctly predicted observations to the total number of observations
- **Precision** ($\frac{TP}{TP+FP}$): a ratio of correctly predicted positive observations to the total number of predicted positive observations
- **Recall** ($\frac{TP}{TP+FN}$): a ratio of correctly predicted positive observations to all observations in actual positive class
- **F1 Score** ($2 \times \frac{Recall \times Precision}{Recall+Precision}$): a weighted average of Precision and Recall

A. Abalone Classification

In this section we implement the previously described classification algorithms. Following EDA (Exploratory Data Analysis) the Sex column was categorical with 3 possible values. Therefore we employ the label encoding technique to convert the categorical values to numerical.

The following table outlines the results obtained for different techniques to handle class imbalance, notably undersampling of the majority class and SMOTE to oversample the minority class:

ML Classifier	Sampling	Test Data (average after 10 iterations)			
		Accuracy	Precision	Recall	F1
RBFNN	Undersampling	48.6%	53.9%	55.4%	51.2%
	SMOTE	67.6%	64.1%	80.6%	70.7%
Logistic Regression	Undersampling	45.0%	59.0%	27.0%	36.5%
	SMOTE	81.0%	81.0%	80.4%	80.7%
Shallow NN	Undersampling	45.0%	29.5%	59.5%	39.3%
	SMOTE	67.6%	64.3%	79.6%	71.1%
Decision Tree	Undersampling	47.5%	0.0%	0.0%	0.0%
	SMOTE	73.9%	66.6%	95.1%	78.3%

TABLE I
ABALONE CLASSIFICATION RESULTS FOR DIFFERENT CLASS IMBALANCE SAMPLING METHODS

In Table I we notice a significant improvement across all evaluation metrics when the SMOTE oversampling method is used compared to undersampling. Given there was 32 positive classes in the abalone dataset, undersampling the majority negative class would balance the classes with 32 each. This is not sufficient data for the classifiers to learn the relationship from the features to the class labels, hence the poor evaluation metrics recorded per classifier.

In order to evaluate the performance of our classification algorithms we can plot an ROC (Receiver Operating Characteristic) curve, which plots the true positive rate (sensitivity) against the false positive rate (1 - specificity) for varying classification threshold values. It shows the tradeoff between sensitivity and specificity (any increase in sensitivity will be accompanied by a decrease in specificity). The closer the curve is to the left side of the plot and the top of the ROC plot, the more accurate the classifier is at predicting the positive class. The closer the curve comes to the 45-degree diagonal of the ROC plot, the less accurate the classifier is at predicting the positive class. Lowering the classification threshold classifies more items as positive, thus increasing both false positives and true positives.

The following plot gives the ROC for classification algorithms applied to the abalone dataset:

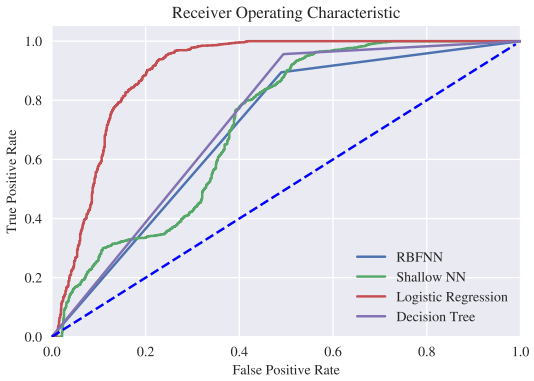

Fig. 5. ROC plot of classification algorithms for abalone data

In Figure 5 we see logistic regression is the superior classifier compared to the others because at all thresholds the true positive rate is higher and the false positive rate is lower. The area under the curve (AUC) for logistic regression is larger than the AUC for the other classifiers. From this we can conclude logistic regression performs the best when predicting the positive class (i.e. if the abalone belong to age group 19).

B. Mushroom Classification

In this section we will evaluate the classification algorithms as previously described after applying them to the mushroom datasets. A mushroom classification model will enable mushrooms to be classified as poisonous or edible based on categorical measurements.

An important preprocessing step to perform on the mushroom data was label encoding of all 22 categorical features into numerical values. We then utilised one-hot encoding to create new columns of 0's and 1's, thus creating unique representations for the original categorical values.

A challenging task in data preprocessing is handling missing values. If missing values were not handled properly by the data scientist, inaccurate inferences may be drawn from the data following classification [13]. To deal with missing values, one can simply remove rows or columns containing these missing values. However, removing entire rows or columns from a large dataset containing a small number of missing values would mean potentially valuable information can be discarded, thus affecting classification performance. Hence we utilise the method of imputation, that is, substituting missing values with appropriate replacements. Various forms of imputation exist, such as median and mean imputation, alongside more sophisticated techniques such as Random Forests, a non-parametric imputation method using decision trees to estimate missing values [14].

Following EDA we discovered the stalk-root column contained all the 2480 missing values out of a total 8124 observations encoded as '?'. We apply mode imputation to the mushroom dataset, such that we can replace missing values in the stalk-root column with the most frequently occurring value in that column. Performing imputation in this way ensures no potentially valuable information is discarded. However given 2480 missing values for stalk-root column represent $\sim 30\%$ of the total information in that column, it may be difficult to conclude our classification results are accurate given only $\sim 70\%$ of the values in the stalk-root were true values. Therefore, we seek to compare and contrast classification performance with this column removed against mode imputation.

1) Mode Imputation: We apply mode imputation to the stalk-root column by replacing missing values with the column's modal value. We divide the mushroom dataset into training and testing sets with an 70%-30% split respectively. After applying each of the four classification algorithms previously described we obtain the following average metrics after 10 iterations:

ML Classifier	Test Data (average after 10 runs)			
	Accuracy	Precision	Recall	F1
Logistic Regression	98.9%	100.0%	97.7%	98.9%
RBFNN	88.4%	94.0%	81.0%	86.9%
Shallow ANN	100.0%	100.0%	100.0%	100.0%
Decision Tree	88.4%	82.0%	96.9%	88.8%

TABLE II
MUSHROOM CLASSIFICATION RESULTS FOLLOWING MODE IMPUTATION

From Table II we notice the shallow neural network had values of 100% for all metrics and had the largest F1 score of all models. Comparing F1 scores across models, we see logistic regression was a better model than the RBF neural network and decision tree models.

2) Eliminating Rows with Missing Values: Given 2480 missing values were present in the mushroom dataset, we can simply remove all rows containing at least one missing value. Similarly to before we divide the data into training and testing sets with an 70%-30% split respectively. After applying each of the four classification algorithms previously described we obtain the following average metrics after 10 iterations:

ML Classifier	Test Data (average after 10 runs)			
	Accuracy	Precision	Recall	F1
Logistic Regression	87.1%	89.9%	74.5%	80.9%
RBFNN	99.9%	100.0%	99.6%	99.8%
Shallow ANN	98.5%	100.0%	96.3%	97.7%
Decision Tree	89.7%	100.0%	73.0%	84.4%

TABLE III
MUSHROOM CLASSIFICATION RESULTS AFTER REMOVAL OF MISSING VALUES IN ROWS

From Table III we notice an overall worse performance for logistic regression and the decision tree models which both have lower F1 scores compared to mode imputation. Both neural network models achieve almost 100% across all metrics.

3) Eliminating Columns with Missing Values: We can simply remove the stalk-root column, thus preserving potentially valuable information for values of the other attributes. Similarly to before we divide the data into training and testing sets with an 70%-30% split respectively. We utilise the method

of K-Folds Cross Validation to repeatedly split the dataset into k distinct subsets/folds. To train the classifier we use k-1 subsets with the last subset used for testing data. Computing the accuracy across all folds will enable an average accuracy to be computed after which we can test the model against the test set.

After applying each of the four classification algorithms previously described we obtain the following average metrics after 10 iterations:

ML Classifier	Test Data (average after 10 runs)			
	Accuracy	Precision	Recall	F1
Logistic Regression	98.9%	100.0%	97.7%	98.8%
RBFNN	88.3%	94.0%	81.0%	87.0%
Shallow ANN	99.1%	100.0%	98.1%	99.0%
Decision Tree	88.7%	82.8%	96.5%	89.1%

TABLE IV
MUSHROOM CLASSIFICATION RESULTS AFTER REMOVAL OF STALK-ROOT

From Table IV we notice the logistic regression and shallow neural network models have the highest F1 score compared to the models previously analysed. The shallow neural network again returned values close to 100%. A 100% precision for logistic regression and the shallow neural network means the algorithms returned substantially more correctly predictive positive results than irrelevant ones, while high recall means that an algorithm returned most of the relevant results.

Given the metrics for the decision tree vary across the tests, we can visualise how the accuracy varies as the maximum depth parameter of the tree is increased:

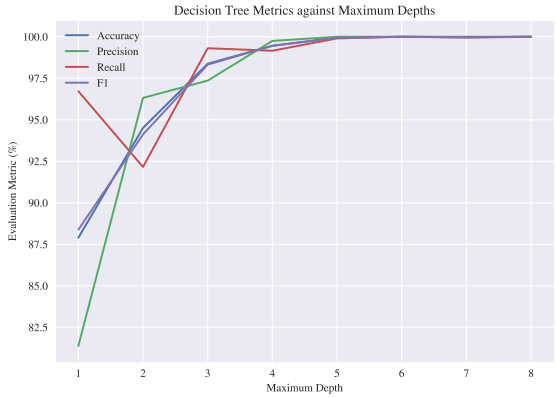

Fig. 6. Classification metrics after varying maximum depth of decision tree for mushroom data

From Figure 6 we see all our evaluation metrics reach 100% when the maximum depth is increased to 5, thus making the decision tree the joint best model for classifying mushrooms as edible or poisonous along with the shallow neural network and logistic regression models.

Following mushroom classification with mode imputation we can plot the following ROC curve:

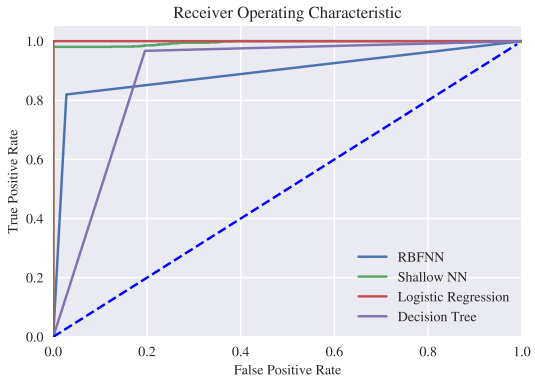

Fig. 7. ROC plot of classification algorithms for mushroom data

From Figure 7 we see the logistic regression and shallow neural network classifiers perform best at predicting the positive class compared to the RBF neural network and decision tree classifiers. From this we can conclude both logistic regression and the shallow neural network are almost perfect at predicting the positive class (whether a mushroom is poisonous or edible).

V. CONCLUSION

In this report we successfully implemented clustering and classification machine learning algorithms to predict the age of abalone and predict if a mushroom is edible or poisonous. Logistic regression was the highest performing classifier for the abalone and mushroom datasets, with the shallow neural network also a high-performing classifier. We successfully separated the PCA clusters for the pulsar data into 2 distinct clusters from the original 8-dimensional space using both k-means and agglomerative hierarchical clustering algorithms. It was found however that the pulsar dataset had heavy class imbalance, so future work could involve reducing the number of spurious examples to reduce the total number of observations.

We envisage future work could involve using a dimensionality reduction and classification technique known as Linear Discriminant Analysis (LDA). Similar to PCA, new principal components can be created which projects data points onto a new axis enabling a separation of classes to be visualised. Hence, LDA can be used as a classification technique. Further classification algorithms, such as Support Vector Machines, Deep Neural Networks and Random Forests would enable a wider range of classifiers to be compared. Further clustering algorithms such as DBSCAN could be used in an attempt to better separate the clusters produced from our PCA analysis. A more sophisticated anomaly detection technique known as Recursive Density Estimation could also be applied to aid detection of anomalous data points [15].

REFERENCES

1 [Online]. Available: https://archive.ics.uci.edu/ml/datasets/HTRU2

2 Morello, V., Barr, E. D., Bailes, M., Flynn, C. M., Keane, E. F., and van Straten, W., "SPINN: a straightforward machine learning solution to the pulsar candidate selection problem," *Monthly Notices of the Royal Astronomical Society*, vol. 443, no. 2, pp. 1651–1662, 07 2014. [Online]. Available: https://doi.org/10.1093/mnras/stu1188

3 Bethapudi, S. and Desai, S., "Separation of pulsar signals from noise using supervised machine learning algorithms," *Astronomy and computing*, vol. 23, pp. 15–26, 2018.

4 [Online]. Available: http://sci2s.ugr.es/keel/dataset.php?cod=115

5 Han, Y., "Machine learning project - predict the age of abalone," Ph.D. dissertation, 06 2019.

6 [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Mushroom

7 Jolliffe, I. T. and Cadima, J., "Principal component analysis: a review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 12, 2016. [Online]. Available: Citethisarticle:JolliffeIT,CadimaJ.2016%Principalcomponentanalysis: areviewand%recentdevelopments.Phil.Trans.R.Soc.A374:%20150202. http://dx.doi.org/10.1098/rsta.2015.0202

8 Gonsales, A., "An approach to choosing the number of components in a principal component analysis," Dec 2019. [Online]. Available: https://towardsdatascience.com/an-approach-to-choosing-the-number-of-components-in-a-principal\-component-analysis-pca-3b9f3d6e73fe

9 Amidan, B. G., Ferryman, T. A., and Cooley, S. K., "Data outlier detection using the chebyshev theorem," in *2005 IEEE Aerospace Conference*. IEEE, 2005, pp. 3814–3819.

10 Dabbura, I., "K-means clustering: Algorithm, applications, evaluation methods, and drawbacks," Sep 2019. [Online]. Available: https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-\drawbacks-aa03e644b48a

11 Day, W. H. E. and Edelsbrunner, H., "Efficient algorithms for agglomerative hierarchical clustering methods," *Journal of Classification*, vol. 1, no. 1, pp. 7–24, Dec 1984. [Online]. Available: https://doi.org/10.1007/BF01890115

12 Bustamam, A., Fitria, I., and Umam, K., "Application of agglomerative clustering for analyzing phylogenetically on bacterium of saliva," *AIP Conference Proceedings*, vol. 1862, no. 1, p. 030126, 2017. [Online]. Available: https://aip.scitation.org/doi/abs/10.1063/1.4991230

13 "Missing values in data." [Online]. Available: https://www.statisticssolutions.com/missing-values-in-data/

14 Rai, S. S., "3 methods to handle missing data." [Online]. Available: https://blogs.oracle.com/datascience/3-methods-to-handle-missing-data

15 Bezerra, C. G., Costa, B. S. J., Guedes, L. A., and Angelov, P. P., "Rde with forgetting: An approximate solution for large values of $$k$$with an application to fault detection problems," in *Statistical Learning and Data Sciences*, Gammerman, A., Vovk, V., and Papadopoulos, H., Eds. Cham: Springer International Publishing, 2015, pp. 169–178.

APPENDIX

A. Original VS Standardized Pulsar Plots

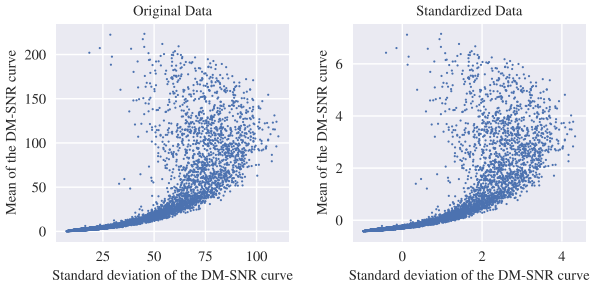

Fig. 8. Mean of the DM-SNR curve against standard deviation of the DM-SNR curve before and after standardization

B. Abalone PCA plot

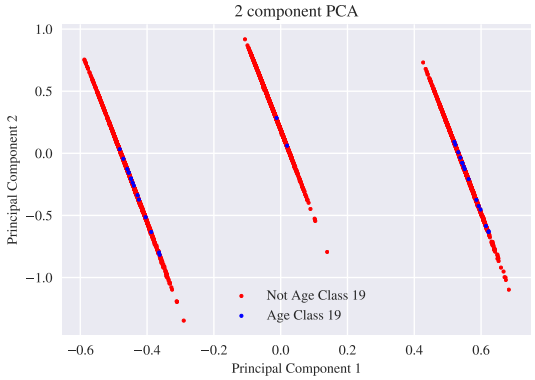

Fig. 9. Plot of 2 principal components with the largest variance for the abalone dataset

C. Abalone Scree plot

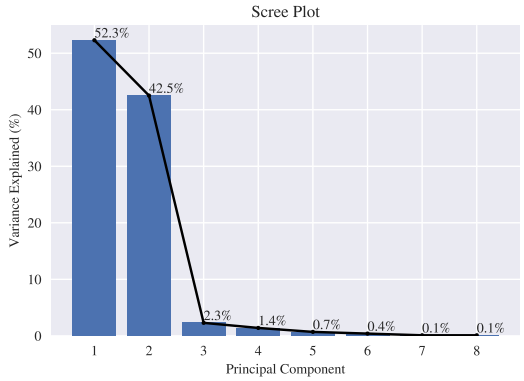

Fig. 10. Scree plot of principal component variance for the abalone dataset

D. Mushroom PCA plot

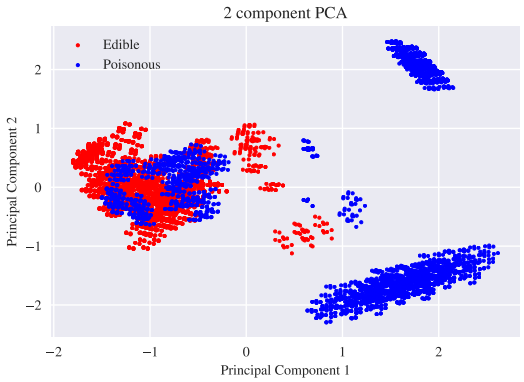

Fig. 11. Plot of 2 principal components with the largest variance for the mushroom dataset

E. Mushroom Scree plot

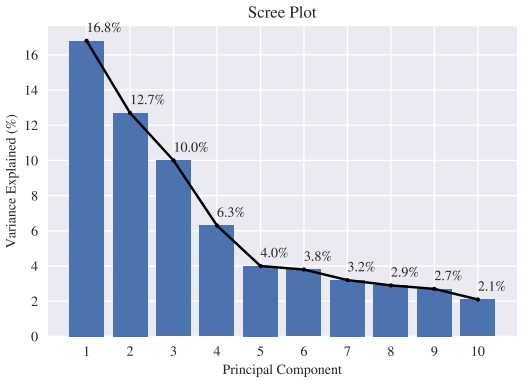

Fig. 12. Scree plot of principal component variance for the mushroom dataset