Practical 2: Classification of Seal Images

Table of Contents

0. Instructions

0.1 Running the application

- **Source code**
  - Jupyter notebook file –Has a detailed view of each step taken to approach the problem.
- **Additional Libraries**
  - Imbalanced-Learn was installed for implementing the resampling techniques. To install imbalanced-learn simply type:

```
pip install -U imbalanced-learn
```

  - LightGBM was installed for evaluating prediction performance. To install LightGBM type:

```
pip install lightgbm
```

- **Code that takes long to run**
  - t-SNE scales quadratically in accordance to the number of samples (N), therefore by default set N=10000, but if you would like a more high-fidelity plot like the one seen the report, you can increase the samples to 40000, but it can take a while to run.
- **Figures and Tables:** Each figure and table have a corresponding method, and this has been noted in the comments of the methods.

0.2 Evaluation Metrics

- $\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \text{ x } 100$
- $\text{Precision} = \frac{TP}{TP+FP} \text{ x } 100$
- $\text{Recall/True-Positive Rate} = \frac{TP}{TP+FN} \text{ x } 100$
- $\text{Balanced Accuracy} = \text{average of recall obtained on each class.}$
- $\text{F1 Score} = 2 \times \frac{Precision\ X\ Recall}{Precision+recall} \text{X } 100$

* For the multi-class datasets, the **macro** average was used for Precision, Recall and F1 score as it calculates the metrics of each label to find their unweighted mean

1. Introduction

Monitoring seals and their different stages of development can be very useful for institutions that are concerned about species preservation and climate change. However, manually sifting through months of data can be time-consuming and labour intensive. Moreover, classifying aquatic animals poses several challenges such as background noise, distortion of images, the presence of other water bodies in images and occlusion [1]. Lastly, real-world datasets often have issues with data imbalance [2], which creates a challenge for machine learning models as most algorithms are designed to maximise accuracy and reduce the error.

Interestingly, with the developments in machine learning and image feature extraction, seal classification can be automated. Additionally, sampling strategies, such as over- and undersampling, are extremely popular in tackling the problem of clas.s imbalance where either the minority class is oversampled, the majority class is undersampled or a combination of the two [3]. Consequently, the practice of combining multiple models to form an ensemble has been shown to address the weaknesses of individual models, thus creating a greater single model overall [4]. This study trains several machine learning models with different resampling techniques, to classify seals from imbalanced datasets of features extracted from images.

In Section 2.2, this study visualises the dataset by producing a visual representation of the hog features, the class distribution and the variance using two different dimensionality reduction techniques. Section 2.3 outlines the process by which subsets of features were chosen with a data-driven approach and how the data was scaled, reduced, and resampled to help improve classification performance. Section 2.4 discusses how the best resampled dataset was chosen for further training and how three different classification models were hyper-tuned using randomised search with 3-fold cross-validation. Section 3 evaluates the performance of the models on the validation set by critically assessing their performance on each class to choose the top-performing model on each dataset. Lastly, Section 4 highlights the performance on the test set showing that the method described in this study, yielded a near state of the art accuracy of 98.908% on the binary dataset and 97.964% multi datasets, thus securing a top 4 finish in the machine learning competition.

2. Method

2.1 Loading the data

2.1.1 Loading and cleaning the dataset

For both the binary and multi-classification datasets, each column had no null values, indicating that there were no instances of missing data. All the data types for the inputs were float64, indicating consistency in the data format. There were also no duplicate values found across both the binary and multi datasets. Hence, no changes were made to the dataset as it appeared clean.

2.1.2 Train/Validation Split

This study found it useful to create a validation set to evaluate the performance of each model by seeing how well each model could generalise on unseen data. The training data was split with an 90/10 split in a stratified fashion, to prevent having a non-representative distribution of the target classes in the validation group.

2.2 Analysing and visualising the dataset

2.2.1 Visualising the HoG Features

To gain some visual intuition of the Histogram of Orientated Gradient (HOG) features, I sliced the first 900 columns and picked an image from each class found in the Y_train datasets **(see Figure 1).** Despite the images being in very low resolution, there are still some visual differences between each class. For instance, the whitecoat image has less dark pixels when compared to the moulted pup. Observing the visual differences between the classes can also be useful for trying to get an intuitive understanding of why a classifier misclassifies a particular class.

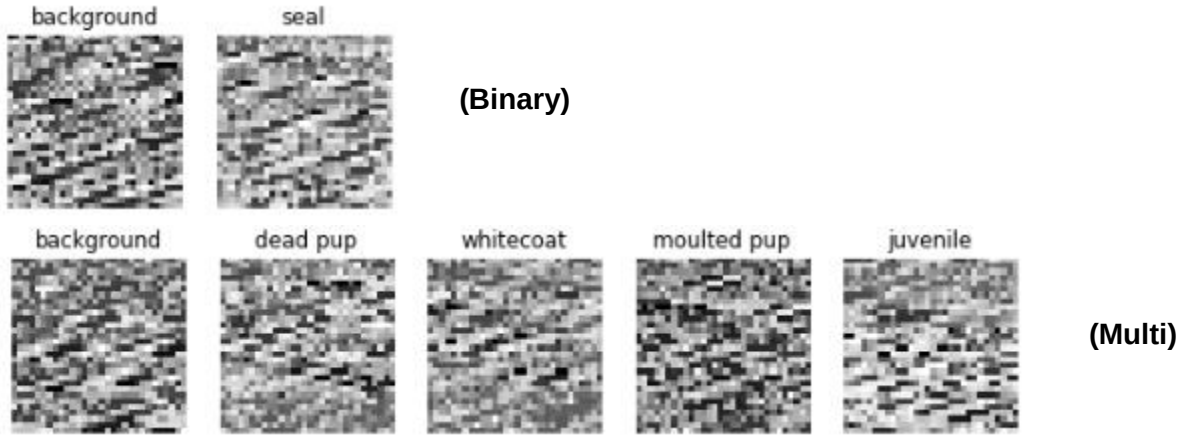

Figure 1: *Visual representation of the HoG Features of each class in the binary and multi datasets.*

2.2.2 Visualising the class distribution of the data

I discovered that the data was heavily imbalanced after I visualised the frequency distribution of the classes (**see Figure 2**). From **Fig. 2,** it is evident that both datasets are heavily skewed towards the background class, where it accounts for 87.5% of the entries in both datasets. In contrast, the juvenile and dead pup classes account for 0.40% and 0.45% respectively, highlighting a high class imbalance when compared to the background class.

Traditional machine learning approaches applied to highly imbalanced datasets may tend to be biassed towards the prediction of the majority class despite the practitioner desiring for both high sensitivity and specificity [2]. Indeed studies have shown that the predicting capabilities of a logistic regression model can be greatly affected by the distribution of the classes [5]. Random Forests can also suffer from the curse of extremely imbalanced data as they are constructed to minimise the overall error rate, thus optimising for the prediction accuracy of the majority class [6]. Moreover, neural networks also suffer from class imbalance, as its learning algorithm considers all individual errors as equally important [7]. Empirical studies have reported success in improving neural networks' performance on imbalanced data through the use of various resampling techniques [8]–[10]. To address the problem of dataset imbalance, different resampling techniques were investigated to see how resampling the dataset affected classifiers prediction performance.

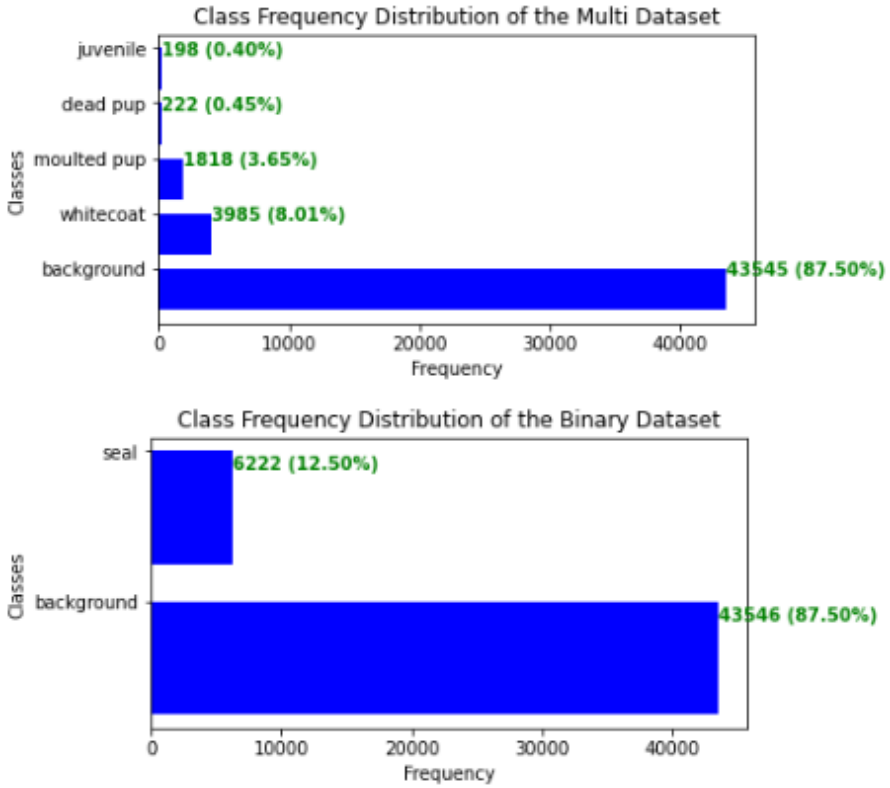

Figure 2: *Horizontal bar graphs showing the class frequency distribution of the binary and multi datasets.*

2.2.3 Visualising variance using PCA and t-SNE

Given the high-dimensional nature of the dataset (964 features), it made sense to implement dimensionality reduction in order to visually explore the whole dataset. The techniques implemented in this study were PCA (Principal Component Analysis) and t-SNE (t-Distributed Stochastic Neighbouring Entities).

PCA

PCA is a popular linear feature extraction technique for reducing the number of dimensions in a dataset whilst retaining most information. PCA preserves variances by identifying the axis that accounts for the largest amount of variance in the training set [11]. It then finds a 2nd axis which is orthogonal to the 1st one, that accounts for the largest amount of remaining variance. In higher dimensions, PCA would find the 3rd axis and then up to an nth axis, based on the number of dimensions in the dataset. The vector that defines each axis is called the Principal Component [11].

Given that PCA maximises the variance of each component, I made sure to standardise the data using Sklearn's StandardScaler, to prevent PCA giving extra weight to larger variables which could lead to biased outcomes. For visualisation purposes, I reduced the dimensions of the dataset down to 3 dimensions and plotted the 1st and 2nd Principal components (PC) against each other (see **Figure 3**). Using the explained variance ratio, I found that the first three components accounted for 20.6% of the dataset's variance, where 9.1% of the dataset's variance

lied in the first axis, 6.5% in the 2nd axis and 5% in the 3rd axis. This suggests that for training purposes, much more components would be needed for a higher variance.

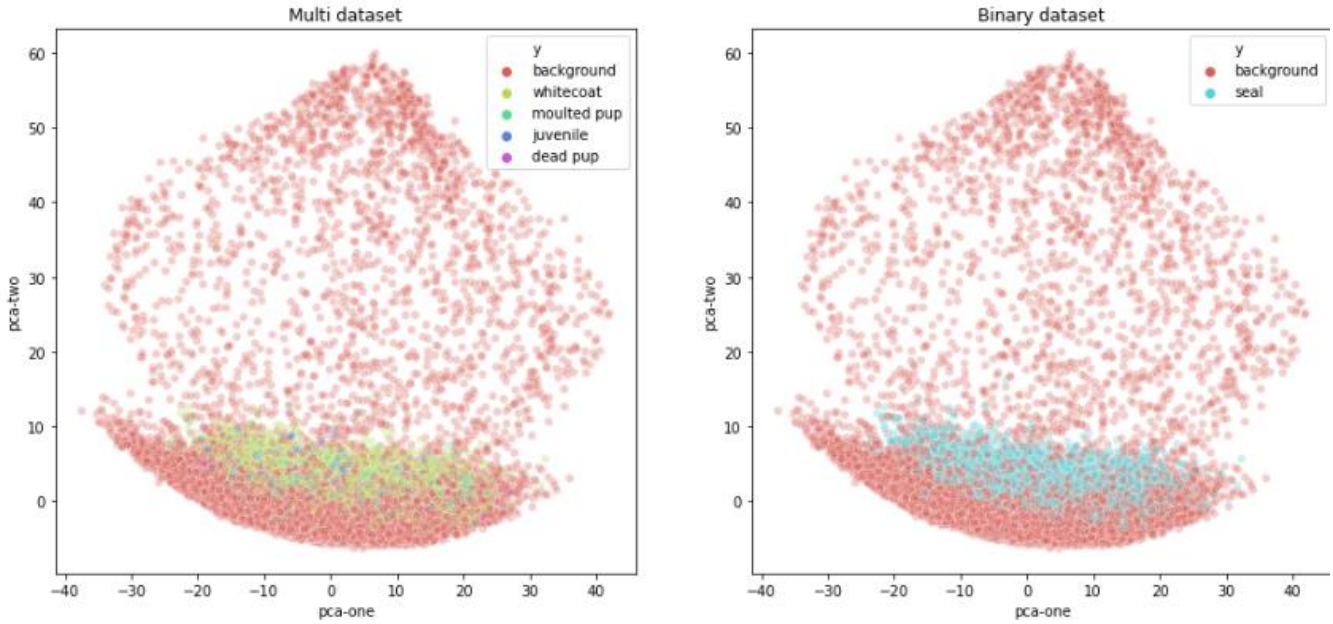

Figure 3: *Graph showing the relationship between the 1st & 2nd PCs in the binary and multi datasets*

As you can see from **Figure 3**, PCA has tried to linearly separate the background class from the seal classes in both datasets; however, it was unable to do so as the data appears mixed. This may suggest that the data is non-linear as PCA can only form clusters on linear data [12]. Hence due to this limitation of PCA, I investigated another dimensionality reduction technique for visualisation – t-SNE.

t-SNE

t-SNE is a non-linear technique for dimensionality reduction that is particularly well suited for the visualisation of high-dimensional datasets. It reduces dimensionality while trying to keep similar instances close and dissimilar instances apart [11]. It also reduces the tendency to crowd points together in the centre of the map [13].

In contrary to PCA, it is a mathematical technique and not a probabilistic one. According to [13], "*t-Distributed stochastic neighbour embedding (t-SNE) minimises the divergence between two distributions: a distribution that measures pairwise similarities of the input objects and a distribution that measures pairwise similarities of the corresponding low-dimensional points in the embedding*". Therefore, it looks at the original data that is entered into the algorithm and looks at how to best represent this data using fewer dimensions by matching both distributions. One drawback of t-SNE is that its computational complexity and memory scales quadratically in the number of data points [13]. Therefore, I made sure to reduce the number of dimensions of the datasets to 100 using PCA and then picked a random subset of 40000 samples before implementing t-SNE.

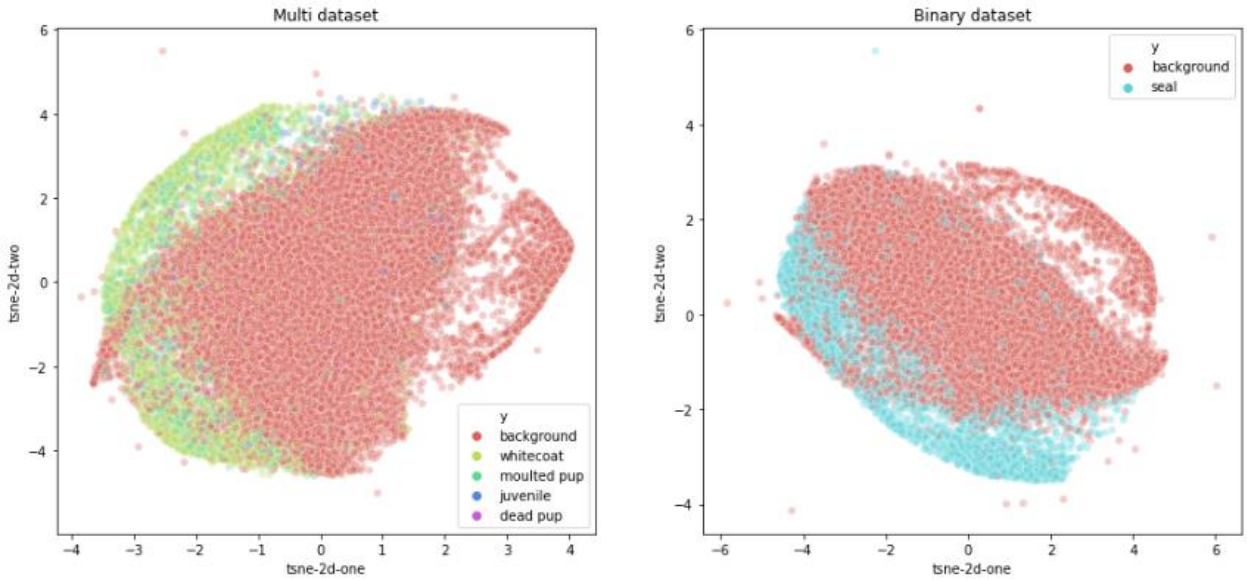

Figure 4: *Graph showing the relationship between the 1st & 2nd dimensions of the embedded space after implementing t-SNE on the binary and multi datasets*

From **Figure 4**, we can see that t-SNE has separated the classes much better than PCA, especially in the binary dataset where there is a cluster of blue dots at the centre of the X-axis. For the multi dataset, whitecoat class appears to have formed clusters at the top left side and bottom left side the plot. The other classes appear to be less identifiable due to their low instances in the dataset.

2.3 Preparing the inputs and choosing suitable features

2.3.1 Choosing the initial subset of features

The input columns in the training sets contained the HoG features of the images, a normal distribution and three colour histograms extracted from the image. I made sure that the dataset was scaled to ensure that the feature values were all weighted equally in their representation. For choosing the right subset of features, I did this empirically by running logistic regression on four feature subsets. These subsets were:

- HoG Features
- HoG Features + Normal Distribution
- HoG Features + Three colour histograms
- Full Dataset (All columns)

After running logistic regression with 3-fold cross-validation on the training set, I found that the HoG Features + Three Colour Histograms dataset performed the best in terms of accuracy, recall, precision and F1 score in both the binary and multi datasets (**see Table 1**).

Feature Subset	Dataset	Accuracy	Precision	Recall	F1 Score
All Columns	Binary	89.10%	67.45%	24.78%	36.25%
	Multi	88.19%	88.19%	88.19%	88.19%
Hog Features	Binary	96.27%	88.61%	80.54%	84.38%
	Multi	87.50%	87.50%	87.50%	87.50%
Hog Features + Normal Distribution	Binary	96.26%	88.64%	80.41%	84.32%
	Multi	95.13%	95.13%	95.13%	95.13%
Hog Features + Colour Histogram	**Binary**	**96.62%**	**89.59%**	**82.59%**	**85.95%**
	Multi	**95.90%**	**95.90%**	**95.90%**	**95.90%**

Table 1: *Table showing the classification performance results of Logistic Regression with 3-fold cross-validation on four chosen feature subsets from the binary and multi training sets.*

Therefore, I chose the HogFeatures + Colour (HogColour) feature subset for implementing dimensionality reduction, resampling and training.

2.3.2 Dimensionality reduction with PCA

Most machine learning algorithms take much longer to train when working with high-dimensional data and can struggle to find an optimal solution [11]. Therefore, I computed the minimum number of dimensions required to preserve 99% of the training set's variance (see **Figure 5**). As a result, only 474 dimensions were needed, which is 49% of the original number of dimensions.

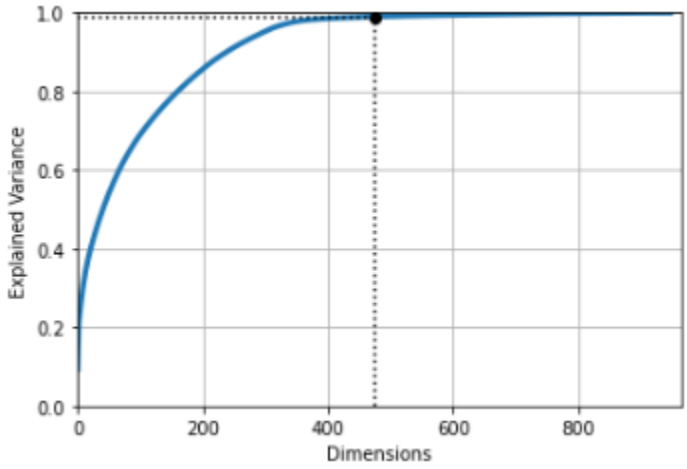

Figure 5: *Graph showing the relationship between explained variance and the number of dimensions on the HogColour dataset*

Next, I fitted and transformed the HogColour binary and multi training sets by reducing the dimensions down to 474 using PCA. I also transformed the validation and testing sets using the fit from the training sets. After that, I ran logistic regression with 3-fold cross-validation on the reduced datasets to observe any changes in classification performance. I found that accuracy had increased to 96.8% from 96.62% on the binary dataset but decreased slightly to 95.23% from 95.9% on the multi dataset. Despite the slight reduction seen on the multi dataset, training time had decreased by 59.08% going down from 62.8ms to 25.7ms.

2.3.3 Resampling techniques

After running PCA on the HogColour dataset, I proceeded by addressing the dataset imbalance issue by implementing a few resampling techniques. The aim was to create additional resampled datasets for training from the reduced HogColour dataset, to find out which resampling technique yielded the best classification performance.

Undersampling

For undersampling the binary dataset I reduced the background class to the same size as the seal class, which was 6222 samples. For the multi-class, I reduced the background class down to the second largest class, which was the whitecoat class, and it had 3985 samples. The undersampling was done using the resampling module from Sckit-Learn, which removes random samples from the majority class.

Oversampling

I implemented two different oversampling techniques. The first was a simple oversampling technique that created additional copies of the minority class. The second was using imblearn's SMOTE or Synthetic Minority Oversampling Technique. SMOTE uses a nearest neighbours algorithm to generate synthetic data that can be used for training [14]. SMOTE potentially performs better than simple oversampling techniques and its widely used. For example, SMOTE was used for detecting network intrusions [15], predicting the distribution of species [16] and for detecting breast cancer [17].

For the binary dataset, I increased the sample size of the minority class to 43545 to match the majority background class. Whilst for the multi dataset, I implemented SMOTE to increase the samples of the dead pup, woulted pup, and juvenile classes to match the class sizes of the background and whitecoat classes (All equal to 3985). Hence, the resampled multi dataset had a combination of oversampling and undersampling

2.4 Selecting and training classification models

In the following section, I use logistic regression to find the top-performing resampling dataset and then introduce new models to find the top-performing model after fine-tuning.

2.4.1 Models

Logistic regression

Logistic regression was mainly used as a baseline classifier to help with empirically discovering top-performing datasets and to compare against other classifiers. I used Scikit-Learn's 'sag' solver which implements Stochastic Average Gradient descent. A variation of gradient descent and incremental aggregated gradient approaches that uses a random sample of previous gradient values. I used this solver because it tends to work faster on big datasets than the default "liblinear" solver [18].

Multi-layered Perceptron Classifier (MLP)

After visualising the dataset, I could see that the data had non-linear relationships; hence I chose to test the performance of an MLP classifier as it has the advantage of being able to learn non-linear models. The MLP classifier implements an algorithm that trains using backpropagation. It differs from logistic regression in that between the input and the output layer; there can be one or more non-linear layers, called hidden layers. The MLP classifier also supports multi-class classification by applying Softmax as the output function.

LightGBM

Both Logistic regression & the MLP classifier train using gradient descent, therefore I decided to choose an alternative classifier that had a tree-based learning algorithm. Microsoft's LightGBM released in 2017, is a very popular powerful algorithm that is said to have faster training speed than most algorithms, lower memory usage and better accuracy. It is a gradient boosting decision tree, unlike XGBoost, implements Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) to drastically reduce training times while still retaining a high accuracy [19]. GOSS helps to reduce the data size by removing significant proportions of data instances with small gradients while reduces the number of features by implementing a greedy algorithm [19]

Ensemble Voting Classifier

The ensemble voting classifier combines conceptually different machine learning classifiers and uses a average predicted probabilities (soft vote) to predict the class labels. Such a classifier can be useful for a set of equally well-performing model in order to balance out their individual weaknesses [20]. Hence I combined logistic regression, MLP and LightGBM after in a single model after fine-tuning, to help overcome their weaknesses.

2.4.2 Finding the best resampled dataset

For finding the best-resampled datasets, I chose the datasets that enabled the logistic regression to have the highest balanced accuracy on the validation set.

After training the logistic regression classifier on the resampled datasets, I made predictions on the unseen validation set to see how well the classifier could generalise **(see Table 3)**. I found that training on the "PCA Only (Control)" dataset yielded the highest accuracy and F1 score at 96.8% and 86.59% respectively but had the lowest balanced accuracy. This was because the PCA control dataset enabled the logistic regression classifier to predict the background class at a 99% accuracy, granting it a much higher precision than if it was trained on the other datasets. This is important because having a high accuracy on the background helps reduce the number of false positives, granting more reliability on the classifier. Interestingly, resampling did help improve overall recall as each resampled dataset had a higher recall than the control and detected more seals correctly. However, the difference in accuracy, precision and f1 score between the control and the resampled datasets was too great as having a hence I decided to choose the control for hyperparameter turning.

Binary Dataset	Recall	Accuracy	Precision	Balanced Accuracy	F1 Score
PCA + SMOTE	89.97%	94.96%	74.86%	92.82%	81.72%
PCA + Oversampled	91.11%	94.14%	70.50%	92.88%	79.58%
PCA + Undersampled	91.51%	93.71%	68.59%	92.77%	78.45%
PCA Only (Control)	**90.7%**	**96.823%**	**86.90%**	**90.07%**	**86.59%**

Table 3: *Table showing the classification performance on the validation set after training logistic regression on the resampled binary datasets.*

For the Multi-dataset, I tested the under- and oversampled dataset against the PCA only (Control) dataset (**see Figure 6**). Training on the PCA only dataset yielded a 94% accuracy while the under- and oversampled dataset, had a lower accuracy of 85%. The high accuracy from the control dataset was driven by the 99% accuracy seen on the background class. Despite the resampled dataset having a much lower accuracy, it had a higher recall of 65.8% compared to 56.3%. Thus further suggesting that resampling does help improve recall and the number of seals to be detected correctly. However, once again, the overall accuracy discrepancy between the PCA only and the under- and over-sampled dataset was too great hence I moved forward with control PCA dataset.

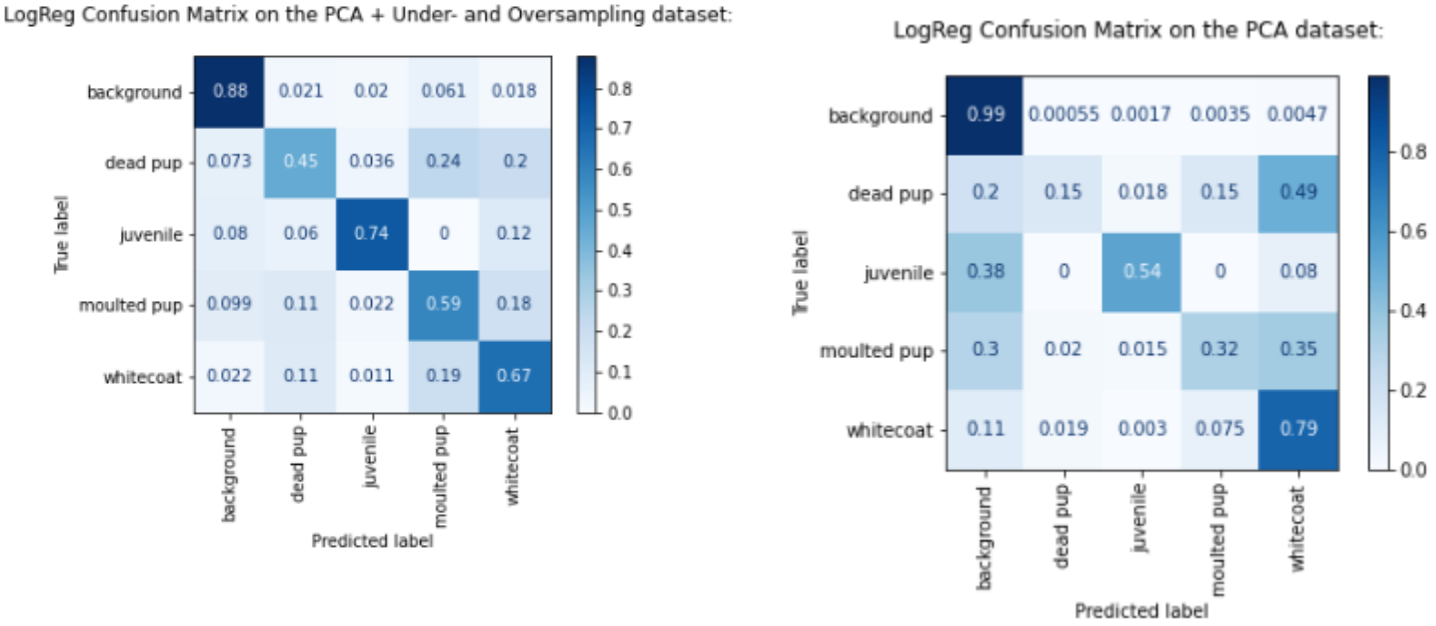

Figure 6: *Confusion matrices showing the classification performance of the logistic regression classifier on the validation set after training on the under- and oversampled dataset and the PCA only (control) dataset. Values are normalised.*

2.4.3 Fine-tuning the models

Randomised search with cross-validation

Due to the size of the datasets and the opportunity to test a large hyperparameter space, I decided to run randomised search as opposed to grid search to tune the hyperparameters of the RF, MLP and Logistic Regression classifiers. Indeed [21] found randomised search to outperform grid search at a fraction of the computational and time cost when compared with neural networks configured by a pure grid search. With the same computational budget, random search tends to find better models by effectively searching a larger, less promising configuration space [21]. Therefore, I trained in total 3000 models (500 combinations x 3 classifiers x 2 datasets) with 3-fold cross-validation and picked the top-performing model from each classifier after running on each dataset.

For LightGBM, as recommended by Microsoft I focused on learning rate and number of leaves to help improve accuracy. Number of leaves controls the maximum tree leaves for on each tree, a

higher number can cause overfitting, hence I tested a sensible range between 1 to 100. For learning rate it was important to start small but have a large number of iterations hence I kept at a range between 0.1 and 1. The top performing learner for binary had a learning rate of 0.1 and number of leaves = 31 whilst for multi learning rate=0.1 and number of leaves =35.

For the MLP classifier I set the activation function to ReLU due to its benefit of reducing the likelihood of a vanishing gradient. The solver was set to "adam" because it's a stochastic gradient optimiser that works well on large datasets [22]. I decided to only tune the learning rate, hidden layer sizes, momentum and alpha (regularisation parameter). The learning rate controls the step size, so a small learning rate may take too long to converge whilst a large one may cause the algorithm to miss a global minimum. Momentum helps the algorithm converge during gradient descent by preventing oscillations that could inhibit convergence to a good solution. The hidden layer sizes control the number of neurons in the hidden layer and also the number of hidden layers. While the alpha parameter helps control the L2 penalty term for regularisation. The hyperparameter space for the random search was 10 x 10 x 10 x 2 dimensions, where I chose 10 different values between 0.001 and 0.9 for the learning rate, momentum and alpha while picking 2 different hidden layer and neuron configurations. The top 5 performing models for both binary and multi datasets had learning rates of 0.001, suggesting that a low learning rate was beneficial for finding a good solution. The top-performing model on the binary dataset had momentum=0.2, learning rate=0.001, 2 hidden layers with 100 neurons in each and alpha=0.8. The top-performing model on the multi dataset had momentum=0.4, learning rate=0.001, 2 hidden layers with 100 neurons in each and alpha=0.6.

For logistic regression, I mainly tuned the 'C' parameter which controls the strength of the L2 regularisation. I tested a range of float values between 0 and 10 but found that changing this parameter didn't make a large impact on performance. Nonetheless, the top-performing model on the binary dataset had C=0.03 and on the multi dataset had C=2.5.

3. Evaluation on the validation set

Model	Dataset	Recall (%)	Accuracy (%)	Precision (%)	f1 score (%)	Balanced Accuracy (%)
Logistic Regression	Binary	82.77	96.88	91.47	86.90	90.87
	Multi	52.65	94.486	60.88	55.08	52.65
LightGBM	Binary	82.26	97.02	93.15	87.37	90.69
	Multi	43.27	94.38	57.80	43.27	43.27
MLP Classifier	Binary	**87.91**	**97.82**	94.34	**91.01**	**93.58**
	Multi	**56.15**	95.12	71.35	**60.46**	**56.15**
Voting Classifier	Binary	86.37	97.81	**95.72**	90.81	92.91
	Multi	49.16	**95.33**	61.16	53.79	49.05

Table 4: *Table showing the classification performance on the validation set after training the models on the PCA only binary & multi datasets. The numbers highlighted in green indicate the highest score on that dataset*

After picking the top-performing models, I evaluated their performance on the validation set. The MLP classifier was the top-performing model on the binary dataset; where it had an accuracy of 97.82% and a balanced accuracy of 93.58%. The voting classifier came in very close in 2nd place, with an accuracy of 97.79% and balanced accuracy of 92.9%. It also had the highest precision, suggesting that ensembling helped to improve the ratio of true positives and false positives. From the receiver operating characteristic (ROC) curve below (**see Figure 7**) one can visually see how the MLP classifier has a higher true positive rate and larger area under the curve than the other classifiers.

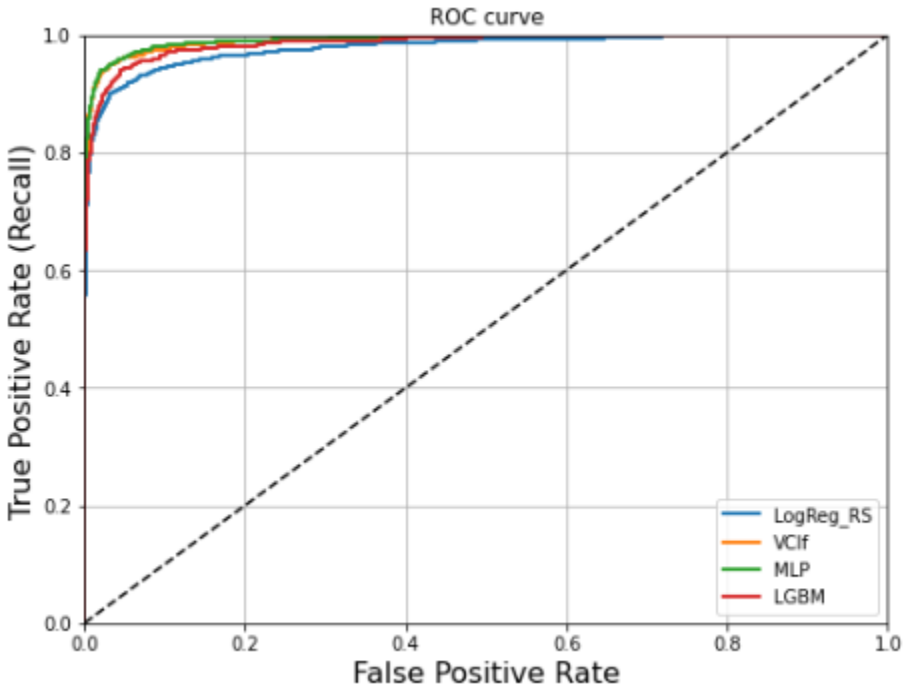

Figure 7: *Graph showing the ROC curve of the models based on predictions made on the binary validation set*

On the multi-dataset, the results were more interesting. The MLP classifier performed the best in terms of recall, f1 score and balanced accuracy **(see Table 4),** however, the Voting classifier had a higher accuracy overall (95.33% vs 95.12%). From the confusion matrices **(see Figure 8**), we can see that combining the models did help classify the background and the whitecoat classes better than the MLP alone. However, the MLP classifier was able to predict the moulted pup, juvenile and dead pup classes far better than the voting classifier. Since the voting classifier votes 'softly' by returning the class label with the highest average probability, LightGBM and Logistic regression may have had much weaker predictions for the moulted pup, juvenile and dead pup classes than the MLP classifier, dragging down the weighted average probability and causing the voting classifier to predict them poorly. Therefore, for the voting classifier to have better predictions on all classes than the rest of the models, models should be combined in a way to increase the weighted average of probabilities. Nonetheless, the voting classifier achieved a higher accuracy overall on the multi dataset, hence there are some merits to its performance.

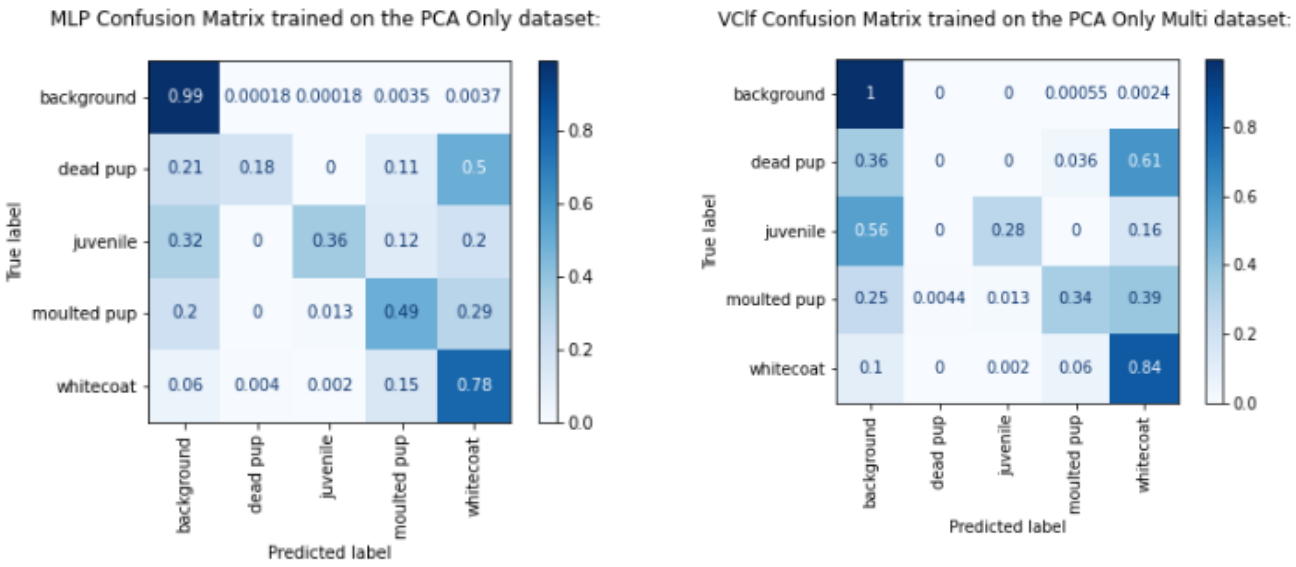

Figure 8: *Confusion matrices showing the classification performance of the MLP & Voting Classifiers (VCLF) on the PCA only multi dataset*

5. Predictions on The Test Set

After picking the top-performing models for each dataset, where it was MLP for binary and Voting Classifier for Multi. The classifiers were able to get a 98.908% accuracy on the binary dataset and a 97.964% accuracy on the multi dataset, resulting in a top 4 finish in the Machine learning competition. This suggests that the methodology described for addressing this problem achieves near state-of-the art performance on this dataset.

5. Conclusion

The purpose of this study was to classify seals using features extracted from an imbalanced dataset of images. Section 2.2 provided a visual representation of the dataset by showing the frequency of the class distribution and the relationship between the features using PCA and t-SNE.

Section 2.4 outlined how a subset of features was chosen and how the data was processed. Indeed, this study found that using the HoG features + three colour histogram features helped improve the accuracy and balanced accuracy of the logistic regression classifier. Reducing the dimensionality of the data also played an essential role in reducing the training times and improving accuracy on the binary dataset. Importantly, this study showed that resampling an imbalanced dataset by undersampling, oversampling or both does help improve overall recall but can lead to lower accuracies if too many samples are removed.

Section 2.4 described how the classifiers were trained and which top models were chosen after randomised search. Logistic regression trained on the resampled datasets showed improvements in recall but had less accuracy when compared against the datasets that were not resampled

(PCA only). Section 2.4 also discussed the benefits of randomised search, which included the ability to search within a large hyperparameter space while controlling the computational budget.

Section 4 evaluated the performance of Logistic regression, LightGBM, MLP classifier and the Voting classifier on the validation set and critically assessed their predictions. It was evident that combining the algorithms in a Voting classifier does have some benefits but can be limited if all the models do not have a similar performance. The MLP classifier performed far greater than LightGBM and Logistic regression across both datasets, which limited their potential synergistic relationship on the binary dataset. But on the multi dataset, the voting classifier had higher accuracy overall mainly because combining the models helped predict the background and whitecoat classes better. Nonetheless, from the prediction performance on the test set we saw that this overall approach leads to near state of the performance on this dataset. Thus, future research would investigate other resampling methods and how to best combine models for ensemble learning to ensure that the ensemble performs the best across on all classes.

6. References

[1] D. Rathi, S. Jain, and S. Indu, “Underwater Fish Species Classification using Convolutional Neural Network and Deep Learning,” 2018.

[2] A. Anand, G. Pugalenthi, G. B. Fogel, and P. N. Suganthan, “An approach for classification of highly imbalanced data using weighting and undersampling,” *Amino Acids*, vol. 39, no. 5, pp. 1385–1391, 2010.

[3] Y. Tang, Y. Q. Zhang, and N. V. Chawla, “SVMs modeling for highly imbalanced classification,” *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 39, no. 1, pp. 281–288, 2009.

[4] T. G. Dietterich, “Ensemble methods in machine learning,” in *International workshop on multiple classifier systems*, 2000, pp. 1–15.

[5] C. Salas-Eljatib, A. Fuentes-Ramirez, T. G. Gregoire, A. Altamirano, and V. Yaitul, “A study on the effects of unbalanced data when fitting logistic regression models in ecology,” *Ecol. Indic.*, vol. 85, pp. 502–508, 2018.

[6] C. Chen, A. Liaw, and L. Breiman, “Using Random Forest to Learn Imbalanced Data,” *Discovery*, no. 1999, pp. 1–12, 2004.

[7] C. L. Castro and A. P. Braga, “Novel cost-sensitive approach to improve the multilayer perceptron performance on imbalanced data,” *IEEE Trans. neural networks Learn. Syst.*, vol. 24, no. 6, pp. 888–899, 2013.

[8] F. Fernández-Navarro, C. Hervás-Martinez, and P. A. Gutiérrez, “A dynamic over-sampling procedure based on sensitivity for multi-class problems,” *Pattern Recognit.*, vol. 44, no. 8, pp. 1821–1833, 2011.

[9] Y.-M. Huang, C.-M. Hung, and H. C. Jiau, “Evaluation of neural networks and data mining methods on a credit assessment task for class imbalance problem,” *Nonlinear Anal. Real World Appl.*, vol. 7, no. 4, pp. 720–747, 2006.

[10] J. Lan, M. Y. Hu, E. Patuwo, and G. P. Zhang, “An investigation of neural network classifiers with unequal misclassification costs and group sizes,” *Decis. Support Syst.*, vol. 48, no. 4, pp. 582–591, 2010.

[11] A. Géron, “Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems SECOND EDITION,” 2019.

[12] J. Lever, M. Krzywinski, and N. Altman, “Points of Significance: Principal component analysis,” *Nat. Methods*, vol. 14, no. 7, pp. 641–642, 2017.

[13] L. van der Maaten and G. Hinton, “Visualizing data using t-SNE,” *J. Mach. Learn. Res.*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[14] N. V Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: synthetic minority over-sampling technique,” *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.

[15] D. A. Cieslak, N. V Chawla, and A. Striegel, “Combating imbalance in network intrusion datasets.,” in *GrC*, 2006, pp. 732–737.

[16] R. A. Johnson, N. V Chawla, and J. J. Hellmann, “Species distribution modeling and prediction: A

class imbalance problem," in *2012 Conference on Intelligent Data Understanding*, 2012, pp. 9–16.

[17] A. Fallahi and S. Jafari, "An expert system for detection of breast cancer using data preprocessing and bayesian network," *Int. J. Adv. Sci. Technol.*, vol. 34, pp. 65–70, 2011.

[18] Scikit-Learn, "sklearn.linear_model.LogisticRegression," 2020.

[19] G. Ke *et al.*, "LightGBM: A highly efficient gradient boosting decision tree," *Adv. Neural Inf. Process. Syst.*, vol. 2017-December, no. Nips, pp. 3147–3155, 2017.

[20] Sklearn, "Emsemble Methods," 2020. [Online]. Available: https://scikit-learn.org/stable/modules/ensemble.html.

[21] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *J. Mach. Learn. Res.*, vol. 13, no. 10, pp. 281–305, 2012.

[22] Scikit-Learn, "MLP Classifier," 2020. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html.