# Student Number: 183708
# Kaggle Team Name: AlexHolder


## Abstract

**This report addresses the Kaggle Competiion entitled "Brighton, a memorable city!". This is a binary classification task of memorable versus not memorable image data. Labelled samples are already given as feature vectors and so this paper's aim is to offer a guide as to the steps one might take in pre-processing this type of data and also demonstrate and compare the performances of various classifiers in regards to the task.**


***Keywords*—** Binary Classification, Pre-Processing

## 1. Introduction

Provided for this task are 247 samples of labelled training data and 11874 sets of feature vectors for testing. Additionally we are provided with 2219 more labelled training samples, the proportions of the binary classes in the test data and also a confidence label for each of the original and additional training samples. Using this information provided we can show in both our pre-processing and our choice of classifier, possible ways of dealing with the task. Model selection was done with intention of providing simple solutions as to properly examine the effects of different classifiers and their hyper-parameters on performance. The steps in our pre-processing was done to make our data agreeable with the classifier and overcome the built-in issues of the information given.

## 2. Hurdles in our Data

Each one of the sets of information provided pose their own challenge: Our additional data contains missing values and so is incomplete, the proportion of the two classes in the training data compared to the test data shows an imbalance, annotation confidence shows only approximately 9 percent of all memorable class0 data was labelled with complete certainty furthering the imbalance and finally training feature vectors were extracted from images in the city of Brighton whilst our test data was collected in London, presenting a domain adaptation problem.

## 3. Deciding Approach

For this classification task two types of models were used: The first being a Multi-layered Perceptron (MLP) from both Sklearn's API and Keras' deep learning API, and the second being Sklearn's AdaBoost estimator, however this only had a very specific function. Testing originally started using the Sklearn MLP due to its simplicity to implement, its efficiency dealing with highly-dimensional data [2] and due to its use of back-propagation [7] (essentially a generalization of the least mean squares algorithm for the linear perceptron) it can classify non-linearly separable data [8, 1]. The Sklearn MLP uses a logistic activation function (sigmoid) whereas its Keras counterpart makes use of the rectified linear unit (ReLU) in order to avoid the "vanishing gradient" problem [11]. The AdaBoost estimator, which works by combining multiple weaker classifiers to form an eventual stronger classifier, was chosen due to the lack of parameter tweaking needed and its ability to avoid overfitting [10]. As an estimator on its own AdaBoost did not return the most accurate of results but together with the Sklearn and Keras MLPs it formed part of a voting system, where final predicted classes was done by a rounded element wise average across all classifiers' outputs which was fairly successful.

## 4. Methodology

Model selection was done by testing and evaluating different classifiers based on how they performed when tested on a split of the training data with performance being measured in two ways. First by comparing the proportion of classes in the predictions against the true values pulled from the test proportions spreadsheet it an idea on which classes were getting over or under represented and second by evaluating a confusion matrix on the predictions versus actual classes and using Sklearn's classification report class we can get a more in-depth understanding of the first evaluation system and compare metrics such as precision, recall and F1-score.

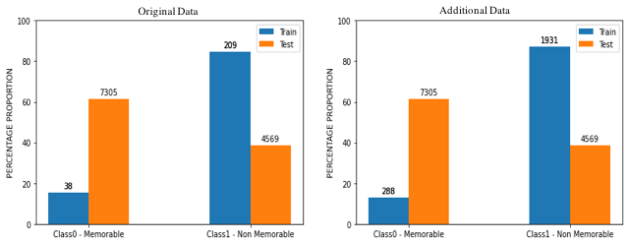


Figure 1. Data Distribution of Original and Additional Data

### 4.1. Feature Pre-Processing

Below shows our pre-processing pipeline in the order those steps were carried out.

#### 4.1.1 Additional Data

First step in our pre-processing pipeline was dealing with the additional training data. Using Sklearn's SimpleImputer class [6] NaN cells were replaced with the mean of that column (feature) and then the whole additional data was concatenated with the already complete data. With this it increased the total training samples from 247 to a total of 2466 meaning our model would be better able to predict the characteristics of the data.

#### 4.1.2 Annotation Confidence

Initially trials were made that kept all of the 100 percent certainty data and a selection of the less than certain data however those

results proved worse than when all the 66 percent data was discounted and so only data chosen by all moderators was used going forward. This did cut our training data's class1 samples in half nearly and reduce class0 samples from 326 to 29.

#### 4.1.3 Feature Selection

Each instance of the data was given as a 4608 dimensional feature vector. The first 4096 were extracted from the fc7 activation layer of CaffeNet (CNN) [3] and the remaining 512 were GIST features [4]. We used Sklearn's SelectKBest class [6] using $Chi^2$ as the scoring function. The higher the $Chi^2$ score the greater the chance that the feature and target class were independent and so that feature should be included [9]. Splitting the data into the different CNN and GIST extracted features meant we now had two possible sets of training data however under testing, no selections of the GIST features gave accuracies higher than the optimum 1500 selected CNN features and incorporating the GIST into a MLP voting system alongside the CNN features failed also. For these reasons only 1500 features were selected from only the CNN extracted data, based on their $Chi^2$ scores.

#### 4.1.4 Re-Sampling

This new total data however showed a massive imbalance 1 favoring class1 data, whereas we know our test data to contain more class0 non-memorable data. To overcome this the minority, class0, was up-sampled using Sklearn's resample method [6] to create a class proportion in the training data equal to that of the test data. This was done as most classifiers will be sensitive to imbalanced data and an equal proportion was a logical representation of what the classifier would ideally predict. Experimenting with up-sampling by favouring class0 showed little to no effect.

#### 4.1.5 Feature Scaling

MLPs are sensitive to feature scaling and so different scalers were tested on the two MLPs and standardized data returned the best results so both train and test data followed route.

### 4.2. Model Selection

Before a classifiers submissions were submitted to Kaggle they were used to predict classes on a subset of the training data. Initially split sizes were kept small and had only 1/4 of the data withheld from the classifier and then later increased to the point where the classifier was testing on 80% of the data. The Sklearn MLP did best out of all with layer sizes of (200,200,200,200) and the ADAM optimizer (chosen for its efficiency), the Keras MLP and the AdaBoost estimator with an example performance shown below:

| Predicted | 0.0 | 1.0 | All |
|---|---|---|---|
| True | | | |
| 0.0 | 1524 | 0 | 1524 |
| 1.0 | 38 | 889 | 927 |
| All | 1562 | 889 | 2451 |

Here we can see our MLP correctly predicted all class0 data and only a small minority class1 was incorrectly predicted as class0. From here various metrics such as F1-score can be calculated to further examine data. In terms of tuning the hyper-parameters for each classifier variations were appended to a list then the classifier iterated through trying the various combinations and returning whichever gave the best accuracy on splits of our training data and also plotting learning curves with different parameters 2. Figure 2 shows the an Sklearn MLP with varying regularization and one can see from the right hand plot that too high a regularization term can produce too high a range of results.

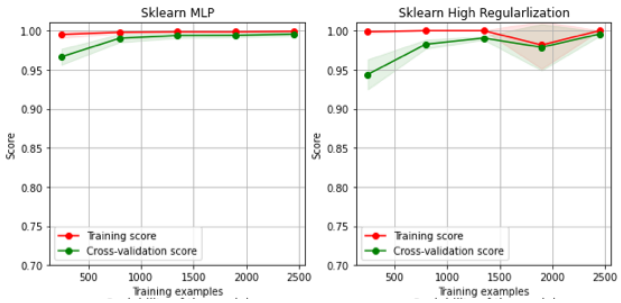


Figure 2. Learning Curves with varying sizes of regularization

Both the Keras and AdaBoost classifiers performed acceptably but they were mainly implemented to attempt to solve the domain adaptation problem of our training images coming from Brighton city whereas test data was extracted from London. AdaBoost is efficient at generalizing and the Keras MLP allows for dropout layers which can also help a model from overfitting. Averaging these classifiers predictions with Sklearn's MLP gave the second highest result from the Kaggle public leaderboard. This was second to the Sklearn MLP by itself.

## 5. Improvements

It is difficult to tell if the voting system helped to generalize the collected models' predictions or if it exacerbated errors within each. In the future a dedicated classifier such as the Transfer Component Classifier [5] from the libTLDA library might be included in the voting system to properly address the domain adaptation problem. A more exhaustive grid-search may also have helped in fine tuning each models parameters further.

## 6. Conclusion

This paper gives a walk-through of possible steps one can take for a similar binary classification task, with a focus on the Sklearn libraries. Pre-processing and model selection are incredibly important to performance and so are the metrics used to evaluate such things.

## References


[1] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989. 1

[2] K.-L. Du and M. N. S. Swamy. *Neural Networks in a Softcomputing Framework*. Springer Publishing Company, Incorporated, 1st edition, 2010. 1

[3] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 2

[4] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42:145–175, 05 2001. 2

[5] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011. 2

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, 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. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 1, 2

[7] F. F. Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. *American Journal of Psychology*, 76:705, 1963. 1

[8] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning Internal Representations by Error Propagation*, page 318–362. MIT Press, Cambridge, MA, USA, 1986. 1

[9] R. Spencer, F. Thabtah, N. Abdelhamid, and M. Thompson. Exploring feature selection and classification methods for predicting heart disease. *Digit Health*, 6:2055207620914777, 2020. 2

[10] S. Wu and H. Nagahashi. Analysis of generalization ability for different adaboost variants based on classification and regression trees. *Journal of Electrical and Computer Engineering*, 2015:1–17, 01 2015. 1

[11] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. 05 2015. 1