Student Number: 169414 & Kaggle Team Name: GJ63

1. Introduction

This report will outline a method of binary classification that attempts so classify images into two classes, 1, meaning that the image is set in a sunny environment and 0, meaning that the image is not. A 259-image training set of concatenated features extracted through Convolution Neural Networks and GIST has been provided and will be used to train the classifier.

The classifier type used for the final implementation is Support Vector Machines (SVM), however several classifiers were tested as explored in Section 2. An SVM is a discriminative classifier formally defined by a separating hyperplane. The classifier outputs a hyperplane whereby classes are separated on either side [1]. This decision boundary is then used in an attempt to classify further unseen data. Reasoning for adopting SVM's as the classifier lies in its ability to classify high-dimensional data. This ability is due to, as the name suggests, SVMs using a small amount of support vectors to create the decision boundary, instead of all the data, making way for a linear hyperplane, as expressed↓

$$K(X, Y) = X^T Y$$

A further feature of interest is the automatic regularization process that occurs within SVM, equated by picking the widest separation margin of data [2]. Further tuning of the C parameter will produce a good model of classification.

2. Method

2.1 Data Preparation

To prepare the data for pre-processing, it was necessary to clean it. The training and testing data was read in via numpy, the first row and column, containing unnecessary identification data, were then deleted. The training data was subsequently concatenated with the additional training set to provide a broader data set. Finally, the prediction column was extracted and saved to a numpy array and deleted from the data.

2.2 Pre-Processing

The next step was to apply a series of data pre-processing techniques. The first was to impute the missing values in the training data as opposed to deleting the corresponding features at a cost of less training data, necessary to be in compliance with scikit-learn estimators, as well as creating a cleaner data set. The imputation strategy was a mean on axis 0 and implemented with sklearn.

After imputation, the data is then standardized whereby each value in the dataset will have the mean value subtracted and divided by the standard deviation of the dataset. Standardizing of data saw a raw improvement of 12.3% in accuracy.

The next and final step of pre-processing was to binarize the data. Binarizing of data works in similar ways to step functions. If the value is above a threshold, it is stepped to 1, if not it is 0. The threshold starting at 0.30 and resulted in an accuracy drop of 4%. The value was then switched to 0.7, resulting in a further accuracy drop of 2%. The next was to find middle ground at 0.5, creating an accuracy of 62%.

2.3 Classifier Implementation

2.3.1 Classifier Testing

To confirm that SVMs were the optimal solution, a test was run to determine the accuracy of each raw classifier for the data set.

Classifier	Cross Validation	Kaggle
MLP	0.66 +/- 0.05	0.61
Logistic Regression	0.60 +/- 0.00	0.56
SVM	0.68 +/- 0.06	0.63

Table 1 SVM Testing

Further optimisation of the parameters for the different classifiers was performed, but as expected, the SVM classifier performed the best and thus will be used for the remainder of the report.

2.3.2 Method

As aforementioned, the classifier used was Support Vector Machines due to the high dimensionality of the data.

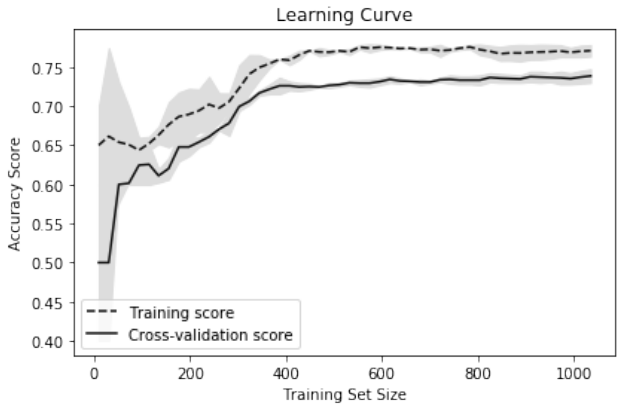

Figure 2 Learning Curve

Figure 1 presents the learning curve of the classifier. As seen, the curves present the presence of overfitting, whereby the classifier has become too used to the training set. Although not present in the data used to produce the learning curve, a bias towards seaside environments will be present. The kernel used in figure 1 was RBF as explored below.

Although a linear kernel was assumed optimal, cross validation was run on several different kernels. The results of these can be found in table 2 .

Table 2 Kernel Test

Kernel	Cross Validation	Kaggle
linear	0.69 +/- 0.05	0.62
polynomial	0.60 +/- 0.00	0.56
sigmoid	0.73 +/- 0.06	0.6
rbf	0.75 +/- 0.07	0.68

With a base cross validation score of 0.75 +/- 0.07, efforts were now focused on optimizing the classifier. The two main parameters we will be focusing on are *C* and gamma. In terms of the hyperplane, *C* determines the weighting of correct vector separation over the largest minimum margin when computing the decision boundary. A higher *C* value will increase the weight of correct vector separation. To determine the appropriate *C* value, utilizing sklearn, a grid search was run. The grid search returned 1 as the optimal *C* value. Cross validation provided us with a result of 0.75 +/- 0.07 for this. Finally, the introduction of class weights increased the cross validation to 0.78 +/- 0.004. Class weights are equated as a decimal proportional to the prevalence of each class.

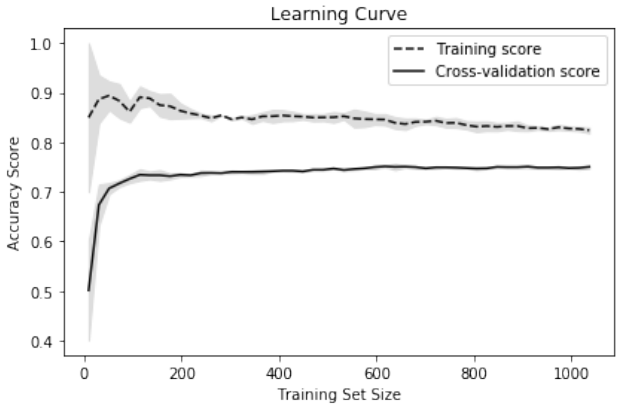

Figure 3 Optimized Learning Curve

Analysis of the learning curve shown in Figure 3, presents improvements in behaviour. It is evident that overfitting has decreased, but is present.

Figure 4 shows the ROC curve of the model. The area under the curve (AUC) allows us to determine the predictive capabilities of the model. This result was 0.78, interpreted as an accuracy of 78% on a split train/test set of data. As shown, the line follows the Y axis closer until a True Positive Rate (TPR) of approximately 0.3, whereas in an ideal world, the line would travel as close to the y axis as possible, presenting mode with a better accuracy.

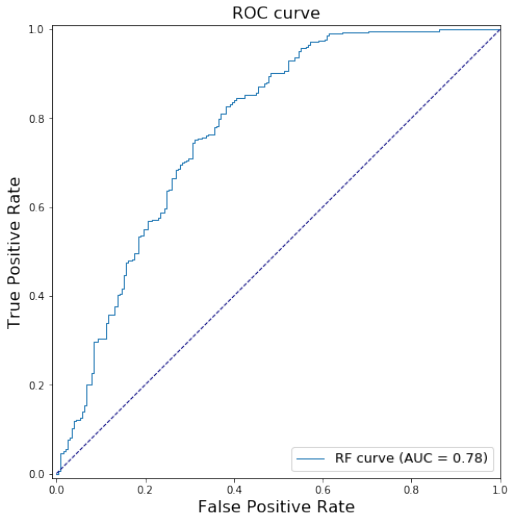

Figure 4 ROC Curve

3. Conclusion

To conclude, the final model scored an accuracy of 68% on the 25% test set available on kaggle. Whilst this model does classify at a greater accuracy than random, unreliable predictions mean that it would not be classed as a good classifier.

Although tested, further improvements could be found within restricting the amount of training data, or features within. Potentially deleting data instead of imputing missing values could also be an improvement worth testing. Further relating to the testing data, it is possible changing the domain type from supervised to unsupervised to prevent overfitting could increase the score.

A further method that could be possible is to separate the GIST and CNN extracted features. These two training sets could then be tested independently with different classifiers to determine which was best for the feature size.

Finally, details of annotation confidence have been provided. It appears that there is some variation in confidence levels, potentially affecting the performance of the classifier. To better the model, the confidence data could be taken into account to determine which training data is use, removing unconfident labels and features.

References

[1] Patael, S (2017). *SVM(Suport Vector Machne) – Theory*. Chapter 2.

[2] Calvetti et al (2003). *Tikhnov Regularization of Large Linear Problems.*

[3] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011

[4] Unknown (2017). *Understanding ROC curves with python*. *https://stackabuse.com/understanding-roc-curves-with-python/*