Student Number: 246529

1. Introduction

In this report, I will address the task of binary classification, specifically distinguishing between happy and sad images. I will use a number of different machine learning techniques to develop a robust classification model capable of accurately predicting the emotional labels of other, potentially unseen, data sets. I will also deal with challenges posed by the data set, such as missing data, and the incorporation of confidence labels to accurately portray my results.

1.1. Approach

The machine learning approach I have decided to use for this task is Random Forest Classifier. This approach combines the ability of decision trees and ensemble learning to solve the problem of binary classification.

A decision tree is used to make predictions based on learning simple decision rules inferred from the data features. In this context, a random forest uses ensemble learning as a collection of decision trees whose results are aggregated into one final result. The idea behind this is that the aggregated predictions of several independent models can lead to a more accurate and robust prediction.

Random forests reduce variance by using random sampling to create diverse subsets of the data for each decision tree. This technique helps reduce over fitting and increases the generalisation ability of the model.

In addition, this also allows each decision tree to focus on a different subset of features, capturing different aspects of the data, helping to mitigate the impact of highly correlated features and improving the overall accuracy of the model.

The classification process begins by creating multiple decision trees each with their respected subset of features, ensuring diversity and reducing the risk of overfitting.

Each tree is constructed using a recursive process. At each step, the algorithm selects the best split among a subset of features based on a specific criterion such as ‘gini’, ‘entropy’ or ‘log_loss’. The splitting process continues until a stopping criterion is met, such as reaching the maximum depth set by the classifier.

Each tree then predicts the class label, and the class with the majority vote becomes the final prediction.

Random forest is well suited for high-dimensional data sets because it can effectively handle many features with their specific abilities mentioned above, that make them very powerful models for binary classification.

2. Methodology

To train my data I undertook many different machine learning processes to allow me to gain the best predictions

Figure 1: Combined Data Accuracy Scores Comparison

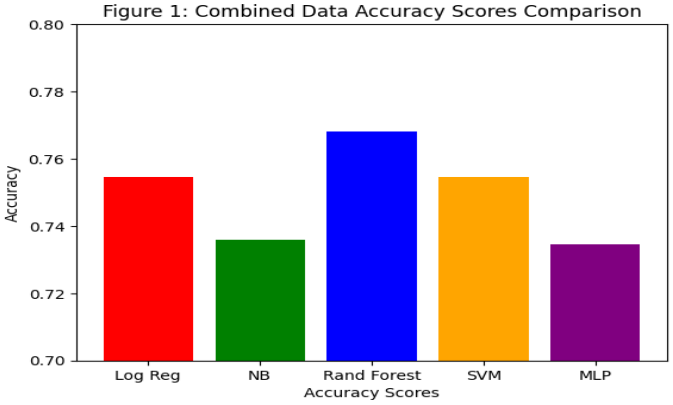


Figure 2: CNN Data Accuracy Scores Comparison

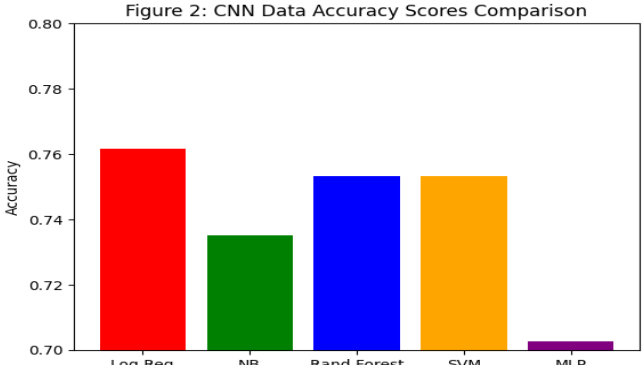
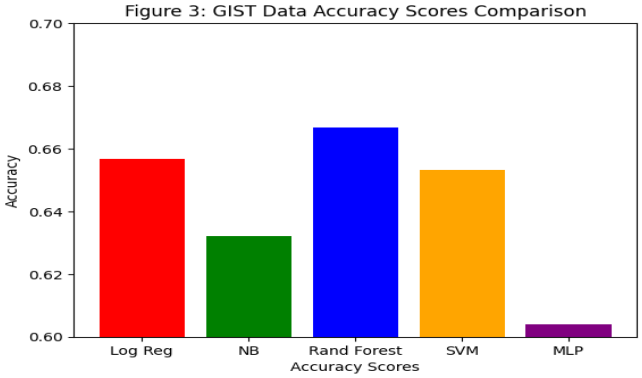


Figure 3: GIST Data Accuracy Scores Comparison



along with the most accurate results. This included using multiple classifiers, feature selection, data preprocessing, hyper parameter tuning and the handling of missing data and confidence labels.

2.1. Data preprocessing

I began by concatenating the two training data sets into one and dealing with the missing values embedded in them. To achieve this, I shuffled the data randomly and took the mean number of each row to fill the missing values. The reason I shuffled the data was to reduce bias and improve generalisation by removing any potential patterns or correlations that may exist between adjacent

features. This also helps ensure statistical validity of tests and measures used during my model selection.

Before applying my data to the classifiers, I also scaled it using PCA and Standard Scaler. Standard scaler is used to standardise features by transforming them to have zero mean and unit variance. This brings all features to a similar scale, preventing those with larger variances dominating those with smaller variances, leading to improved model performance. Principle Component Analysis is a technique aimed to reduce dimensionality in the data set, while retaining the most important information.

These methods of preprocessing are not always relevant for some classifiers as some such as Random Forests and Naïve Bayes are less sensitive to feature scaling, and sometimes perform worse which is reflected in my code. Nonetheless, it is a good practice to consider in most cases as it leads to improved stability and interpretability of the models.

2.2. Model and Feature selection

The classifiers I used were Logistic Regression, Support Vector Classification, Multi-layer Perceptron, Naïve Bayes and finally Random Forest. By using a broad range of classifiers with different characteristics, it gave me many more results on which to base my predictions and choose a model that most optimally fit the data.

As part of the data, we were provided with two types of features: CNN features and GIST features. While both instances of image representation, GIST features are computationally efficient, but less powerful than CNN features.

To address this difference, when training my classifiers, I also trained them solely on both the CNN and GIST features respectively and produced an accuracy score for each of these. By evaluating these features separately, it allowed me to directly compare their performance and assess which set of features is more effective for a particular classifier. The aim of this was to see if I could potentially use two different classifiers on each type of feature in the test data and combine the predictions.

Considering confidence labels in binary classification provides benefits by helping predict classes based on the confidence scores of each feature. This can help deepen the understanding of each classifier's behaviour and its reliability across different confidence levels.

2.3. Hyper-parameter tuning

Hyperparameter tuning is an incredibly important function in all machine learning classifiers. This is because it helps optimise the performance of the models by selecting the best combination of hyperparameters. They control the behaviour of the learning algorithm and in turn significantly impact the model's performance.

I used a popular technique called Grid Search to tune my parameters. Grid Search performs an extensive search of different parameters on the data to pick out which combination works most optimally together, maximising the model's performance.

Some limitations, however, include it taking a very long time to produce the results, especially when dealing with a large parameter grid, and also assuming the best values already lie within the predefined grid, potentially missing out on optimal values not included.

3. Results

Firstly, as previously mentioned, I split both the CNN and GIST features, and experimented with each set individually, to test the performance of the classifiers in more depth, with a potential to combine classifiers. This included me using grid searches to find the best tuned hyperparameters for each classifier 3 times, one for CNN features, one for GIST features, and one for the combined data.

As shown in Figure 3, the y-axis shows a scale from 0.6 to 0.7. This shows the GIST features on their own were the worst performing. Some reasons for this may include that there were less GIST features to train on and therefore produced a lower-than-average accuracy score, or that GIST features are generally lower dimensional and lack power to distinguish between classes. The data shown in Figure 2 further backs this up, as the y-axis now ranges from 0.7 to 0.8 across the whole range of classifiers. CNN features are generally more suitable to image data that involve complex patterns or grained details, as they are designed to extract hierarchical representations of images, as well as having far more data to train on.

Overall, using a combined set of both features gave the best overall accuracy scores, as highlighted in Figure 1, where both the GIST and CNN features are combined. By concatenating the two types of features, this would have allowed the classifiers to train on different aspects potentially leveraging the highest performing parts of each data set. The three best performing classifiers across the board were SVM, Random Forest and logistic regression. This is unsurprising, as they are the classifiers that deal the best with large, highly dimensional data sets like the ones provided.

Once I had decided that Random Forest was the best classifier to use, I tested out applying the use of the confidence labels to the data to see if it would improve the accuracy, however it didn't seem to work.

Overall, one way I think I could improve the data scores I was achieving would be to use validation sets. They provide greater insight in to selecting the best model by helping with tuning hyperparameters and detecting overfitting to ensure their effectiveness and reliability.

References

[1] Maxwell, A.E., Warner, T.A. and Fang, F. (2018). Implementation of machine-learning classification in remote sensing: an applied review. *International Journal of Remote Sensing*, 39(9), pp.2784–2817.

[2] Speiser, J.L., Miller, M.E., Tooze, J. and Ip, E. (2019). A comparison of random forest variable selection methods for classification prediction modeling. *Expert Systems with Applications*, 134, pp.93–101.

[3] Albawi, S., Mohammed, T.A. and Al-Zawi, S. (2017). Understanding of a Convolutional Neural Network. 2017 International Conference on Engineering and Technology (ICET), [online] pp.1–6.