Student Number: 249763

1. Introduction

The report goes over how different machine-learning techniques predict whether an image's content is happy or sad. High dimensionality of 2304 features was used as the dataset, which included class labels with varying levels of confidence and both complete and incomplete data.

2. Approach

Random Forest Classifier, was one ML technique that I explored, it is an algorithm that constructs multiple decision trees after being trained, and each tree is built from a bootstrap sample created by the Random Forest, The randomness of the model comes from the decision on the split of each node of the tree, it is made on a random subset of the features, preventing overfitting. In terms of assumptions, the main one is that the training data provides a good representation of the population.

Support Vector Machines (SVM) is another ML technique for binary classification. It is great with high dimensional spaces such as our dataset and it is versatile, as we can use custom kernels for better accuracy. This approach first looks for a hyperplane with the greatest margin between classes, the closest data points from the classes to the hyperplane are the support vectors. This model's key assumption is that all samples are distributed identically and independently.

Multi-Layer Perceptron (MLP) is a type of Neural network model used for binary classification. I used this model as it is capable of learning non-linear models which the other two techniques are not so good at. MLP, has three layers, an input, a hidden, and an output layer, each of the layers consists of nodes that are interconnected. The algorithm first assigns all nodes in the hidden layer inputs from the input layer, associated weights are multiplied and summed to the inputs. Finally, an activation function is applied to the sum and the output is produced. An assumption MLP makes is the availability of a labelled dataset.

3. Methodology

Data Preprocessing:

Before training my choice of models, I spent some time preprocessing my dataset, to improve my accuracy scores. The second set of training data consisted of missing values (NaN values), to tackle this I used a SimpleImputer class from scikit-learn, the imputation strategy of calculating the mean of each feature in the training set and then replacing the missing values with the corresponding mean in the validation, test and a train data.

I also rescaled the training set using StandardScaler from scikit-learn. The scaler standardized features so they have a standard deviation of 1 and a mean of 0. Scaling makes the data consistent as it brings all training data close to being normally distributed, this plays a big role in how well the classifiers will perform, for example, SVM works better in the smaller scale of features, as it uses a distance function when trying to determine the similarities between data points.

Model Selection was used when splitting training data into validation and training sets, to prevent overfitting and perform better with new data.

I opted in for feature selection using PCA from the scikit-learn library. I used this tool because the training set has a high dimensionality. I used the parameter "n_components" as 100 to reduce the dimensionality to 100 principal components. This is beneficial as the classifiers will avoid overfitting, improving their overall performance.

For the Random Forest Classifier, I began by initializing one with the hyperparameters "n-estimators" equal to 100, and "random state" equal to 42, then I reshaped my training sets into 2D arrays, so I could fit the model with the correct type. To test the classifier, I simply called the classifier and gave it the test set, so I could store it in my CSV file with all predictions.

To train the SVM classifier, I used the hyperparameter "kernel" as "rbf" to initialize it, and fed it, with the same training set using fit. I tested the SVM classifier the same way as the Random Forest classifier,

To train and test the MLP classifier, I carried the same steps as the previous two classifiers.

4. Results/ Discussion

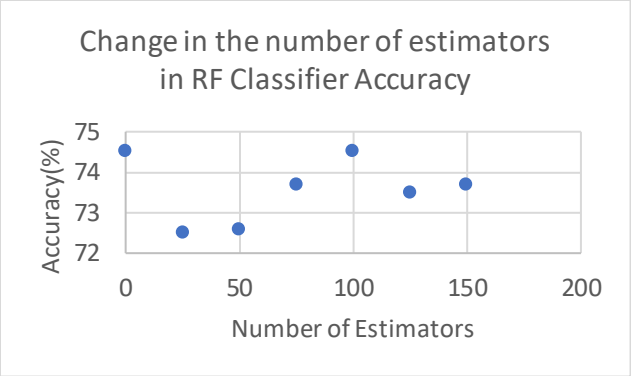

Figure 1: scatter graph showing the relationship between the number of estimators and the accuracy (%)

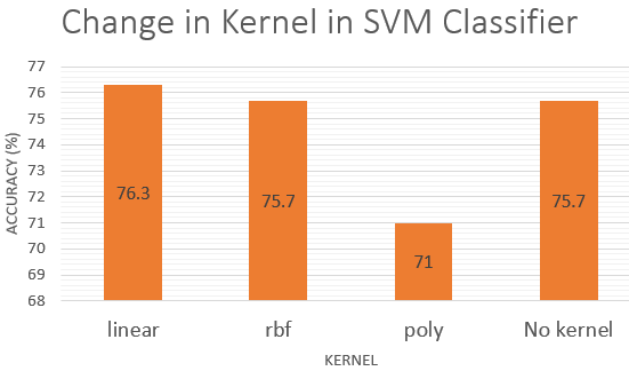

Figure 2: bar graph showing different kernels' accuracy (%).

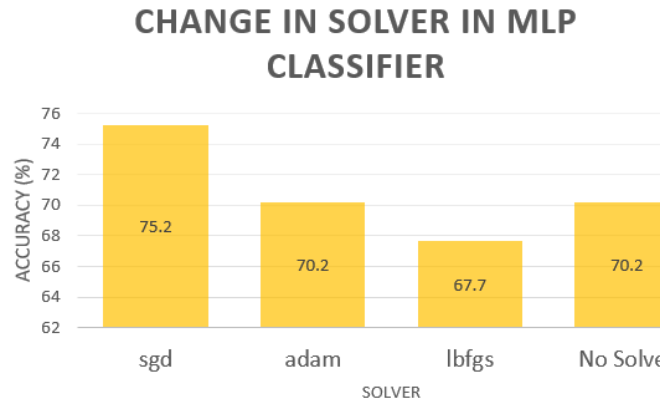

Figure 3: bar graph showing the accuracy (%) of different solvers

From figure 1, I can imply that having no estimators means that the default number is 100 estimators that RF classifier uses. Estimators play as the number of decision trees in the Forest, having over 100 estimators hinders accuracy according to the scatter graph, this could mean that the increase in trees is causing overfitting, this is when the classifier is fitting too close to the training data instead of new data. The opposite could be happening when the number of estimators is lower (underfitting).

One hyperparameter which I changed for the SVM classifier was the kernel. SVM kernels refer to the set of mathematical functions used by the algorithm. The linear kernel for SVM tops the bar chart as seen in figure 2. It was a surprise to see a linear kernel perform better as I was assuming my data was non-linear, this means that there is a somewhat clear relationship between features and labels.

Figure 3, shows that Stochastic Gradient Descent (sgd) is the most suitable solver for MLP. A solver in MLP is for weight optimization. Although, SGD performed best among the rest, it could of performed even better if there was no scaling.

There could have been better hyperparameters to choose from and alter to see an improvement in performance, for instance in MLP the parameter "hidden layers" could have been increased to see how more layers improve the accuracy. Furthermore, when preprocessing, there could have been more work around the confidence labels and seeing a change in performance for different training sets.

One major aspect, which was failed to be successfully explored was combining classifiers. The SVM classifier is great at dealing with high dimensionality, whereas the MLP classifier is not, so to improve the accuracy the SVM and MLP classifiers could have joined forces to achieve a high accuracy score. In addition, MLP is more sensitive to feature scaling, hence the likelihood of low accuracy scores across all solvers, with the aid of the SVM classifier there could have been a greater success.

This experiment has taught me that there is no perfect classifier for binary classification, however, with enough trial and error in finding optimum hyperparameters and potentially a grouping of classifiers for better performances. One massive and personal takeaway was that not all preprocessing methods are necessarily useful for each classifier, some classifiers such as MLP perform better without the feature scaling.

5. Conclusion

In conclusion, further exploration is required to find an optimized classification performance and in the future, there will be more precautions with the choice of preprocessing methods.

References

Elhag, A, A. (2022) *High dimensional learning*. Available at: https://towardsdatascience.com/high-dimensional-learning-ea6131785802 (Accessed: 13 May 2023).

Gandhi, R. (2018) *Support vector machine – introduction to machine learning algorithms*. Available at: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 (Accessed 12 May 2023).

Goyal, K. (2021) *Data processing in machine learning: 7 easy steps to follow*. Available at: https://www.upgrad.com/blog/data-preprocessing-in-machine-learning/ (Accessed: 13 May 2023).

Gramfort, A. et al (2010) *scikit – learn*. Available at: https://scikit-learn.org/stable/index.html (Accessed: 17 May 2023).

Gupta, A. (2021) *Analytics vidhya*. Available at: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/ (Accessed: 12 May 2023).

Kumar, A. (2022) *Data analysis*. Available at: https://vitalflux.com/how-know-data-linear-non-linear/#comments (Accessed: 15 May 2023).