Student Number: 215758

1. Introduction

Determining if a feature-set of an image is either sunny (1) or not sunny (0) is a binary classification problem as there is only two choices. To solve this problem we need a mathematical model.

There are two data-sets, one full of data, and the other partially complete (the second data-set). To use this data-set we have to fill in the blanks with some data, this is called imputing, there are several methods which can be used, we will be using K-Nearest Neighbors (KNN) [1].

The data contains feature sets which are the sets of data to represent the image in a reduced form. One is generated from a convectional neural network (CNN) and the other from a GIST descriptor.

GISTs are global, and as the amount of sun there is in a picture effects the whole image, it is a good suit for this problem[3]. CNNs can are a type of neural network which would be trained from a certain set of data, this could either help more by finding patterns throughout the image, or it could be a model which is not working well for what we want [6]. There is more CNN data than GIST so the CNN may give us more insight.

A State Vector Machine (SVM) can be used for classification tasks, it can use non-linear boundaries which may help if the problem is not linearly separable[8]. The model we will be testing out is a Support Vector Classifier (SVC).

We also will try and test out the effectiveness of a Multi-Layered Perception (MLP) too, this is also appropriate due to it being able to learn classes, it does this through trying to replicate how neurons work and the connections between them. [5][7]

We expect the classifier to do well, around 70% precision and in the other measures too.

2. Methods

2.1. Preprocessing

On the second training data-set, some entries are marked as NaN, these need to be replaced with a valid entry. Using imputing, we can fill them. Originally our technique was to use the mean of the column and fill the invalid entries with them. Next we experimented with KNN, it made the accuracy increase as well as it consistently staying at that level over different trains using five nearest neighbours.

When in the phase of using the original imputing techniques, we started experimenting with removing certain aspects of the data-set. Originally without altering we were getting precision's between 69% to 77%, however it fluctuated between that range, so we switched to just training from the data which had 1.0 certainty, however not all of the data would be correct due to the imputing potentially being wrong.

For each data-set we split it into two, the data which was 1.0 certainty and the data which wasn't. Combined the two certain data-sets together and same with the other. Proceeding this, the 0.66 data was used solely for testing. With the certain data, it was split into testing and training. The two training sets are cut to be the same size and combined, then shuffled.

We thought that using both the CNN and GIST data as they are both reading from the same image and may give more insight on the image. As sunniness is something which would effect the whole picture, using them separately would both be good, however we do not know the model of the CNN which may be incorrectly extracting features and adds another layer of inaccuracies. Experimenting with this we found out that only using the CNN gave better results than using both or only GIST, so we stuck with that. But analysing further previously when using both, we had better scores within the classification report 2, but a slightly higher average precision in the the precision recall curve 1 just using the CNN so we reverted back to using both.

2.2. Model

We tried using a MLP with the original impute method however the precision was low when using the whole data-set, 69%, using it on the formatted 0.66 only data, it was returning 72%, with using the 1.0 data instead, it was 76% using the original imputing technique.

Next we tried, and set with, using an SVC, we chose this because this being a binary classification problem these models are effective for this situation. It is non-linear so there is the benefit is that you can capture much more complex relationships without having to perform difficult transformations on your own[4]. This is implemented using the sci-kit learn library's SVC method:

```
svm.SVC(
  class_weight='balanced',
  kernel='rbf'
  )
```

We set the parameter $class\_weight$ to be $balanced$, meaning depending on the class type, it adds a weight to it to try and balance out the impact of a piece of data on the whole result. The parameter $kernel$ is set to rbf as it is a general purpose kernel[2].

3. Results

We used the certain data to train on only, it gave out some good results but can be improved upon.

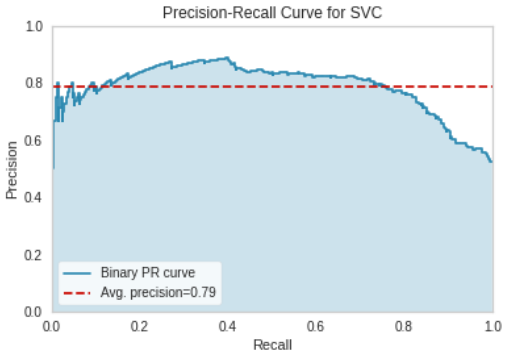

Figure 1. Precision Recall Curve

In figure 1 the curve has a large amount of area under the curve meaning it has a good performance but is still can be improved upon as it still can be unreliable, there is an average precision of 79%.

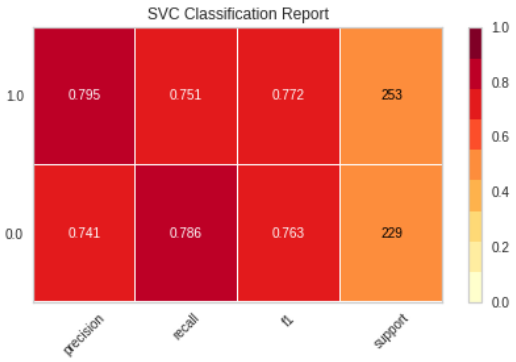

Figure 2. Classification Report

In figure 2, each score is at or above 74% for each statistic. The support shows that the data was fairly evenly spread between the two classes, and the amount of data for the not sunny class (0.0) tallies with the lower scores for those measures.

4. Discussion

The results were in the 70% range which is decent but not enough to be reliable for something being used in production, in the future we would use a data-set which is 100% certain and is filled already.

The certainty of each piece of data should add a weight to the model to make a more certain piece of data have more worth than a lower ranking piece. This would of helped with adding more data for the network to train from which would affect the network in such a way where there is a net gain rather than accidentally damaging it's effectiveness.

The KNN helped with improving the precision, if the data was fully filled out, it would of made the effectiveness of the classification model better, and would be able to truly rely on the certainty of each entry as the labels which include NaN will make them less reliable and the certainty isn't something to go off of but is better than using lower certainties with NaN values.

Using seemingly 100% certain data gave better results, but this may contain data which was imputed, so these cannot be taken for granted, however it has helped overall due to there not being much truly certain data.

The use of the SVC was appropriate as it is well suited for classification and worked well, it seemingly was capped off due to the sparsity of 100% certain cases, as well as the chances of those entries having NaN values in turn making them less certain.

Overall the effectiveness of the classifier is dependent on the quality of data given.

References

[1] L. Beretta and A. Santaniello. Nearest neighbor imputation algorithms: A critical evaluation. *BMC Medical Informatics and Decision Making*, 16, 07 2016. 1

[2] DataFlair. Kernel functions-introduction to svm kernel amp; examples. 1

[3] M. Douze, H. Jégou, H. Sandhawalia, L. Amsaleg, and C. Schmid. Evaluation of gist descriptors for web-scale image search. *International Conference on Image and Video Retrieval*, 07 2009. 1

[4] KDnuggets. What is a support vector machine, and why would i use it? 1

[5] P. Marius, V. Balas, L. Perescu-Popescu, and N. Mastorakis. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8, 07 2009. 1

[6] K. O'Shea and R. Nash. An introduction to convolutional neural networks. *ArXiv e-prints*, 11 2015. 1

[7] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958. 1

[8] D. Srivastava and L. Bhambhu. Data classification using support vector machine. *Journal of Theoretical and Applied Information Technology*, 12:1–7, 02 2010. 1