Image Classification Project

Scott Lee

Introduction

The purpose of this assignment is to build a classifier that best labels photo sentiment.

In this assignment the template code will be altered to attempt to find a classifier model that best fits the classification task. Over the course of this assignment, multiple models will be examined to achieve the best results when classifying images.

This assignment is programmed in python, and all external packages/libraries are listed in the submitted files.

Note on Submitted Code Files

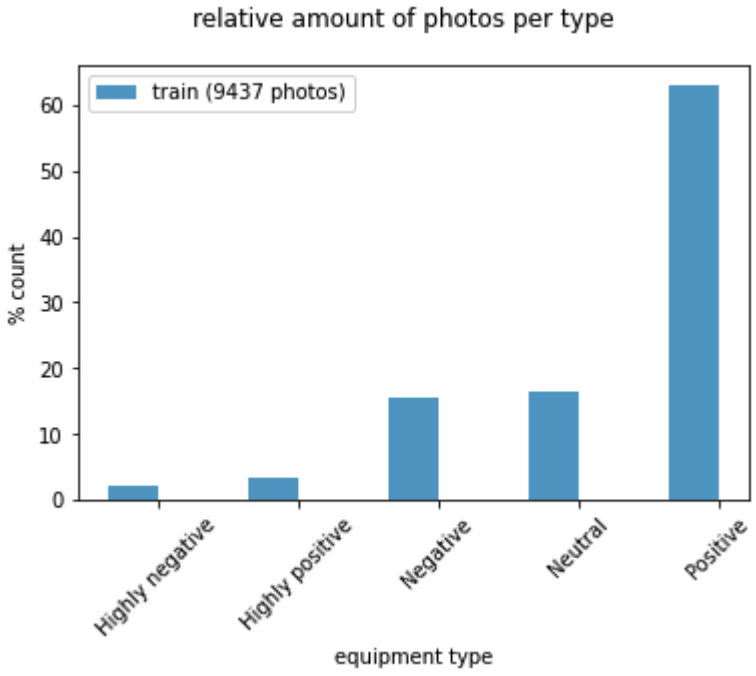
I have submitted two supplementary scripts as a single pdf file due to the file submission cap of 4. The first is the exploratory data analysis (EDA) file which was run on my local machine using the local directory of feature extracted csv's. The second is the model .ipynb file which was run in the Colab environment using the shared directories. EDA was done on the local machine as it was easier to explore the data, and keep it separate from the model code which was often being tuned as I progressed through experimentation. The actual model code was run in the Colabs environment as it had greater processing power. As I was only able to submit one .ipynb file, I have submitted the model, however the script for my EDA is available in .pdf form.

Pre-Processing and Features Used

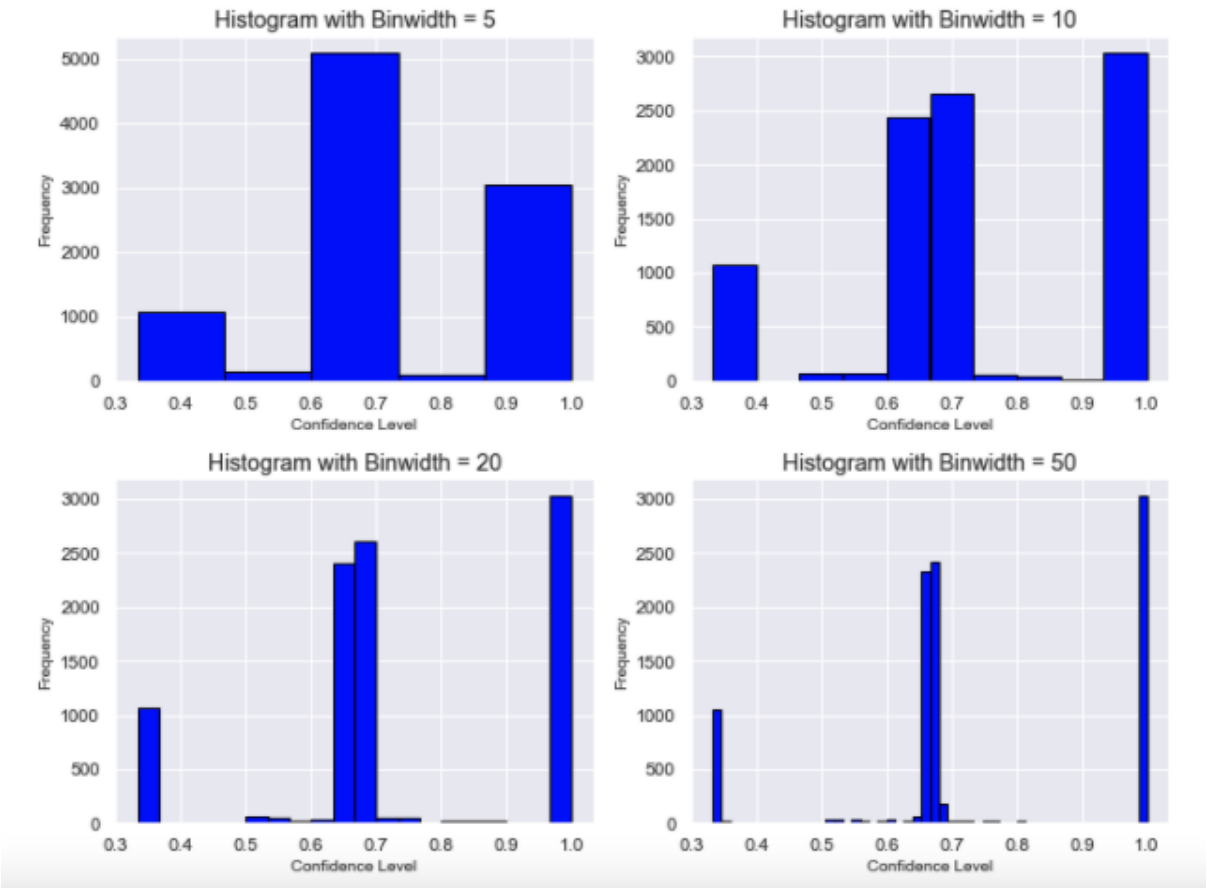
Before a model can be run, it is important to get an understanding of the underlying data.

Exploratory Data Analysis (EDA):

The sample EDA provided was helpful to begin to understand the tendencies of the data. It was clear from initial EDA that the majority (over 60%) of the ‘Label’ data was classified as Positive.



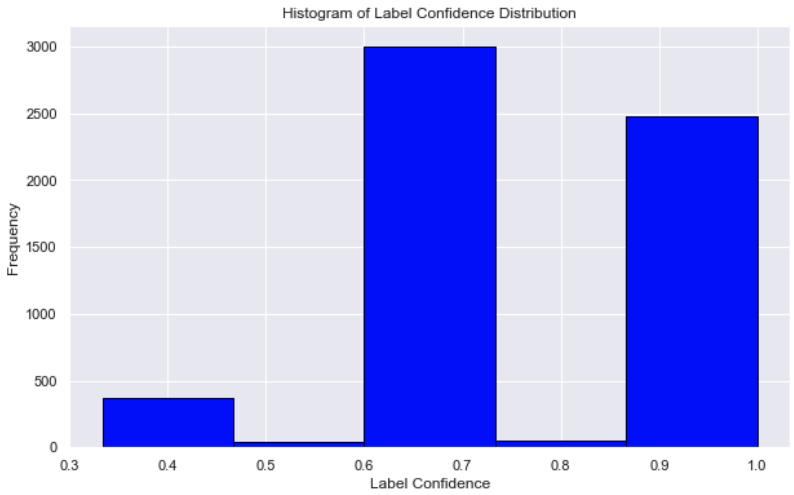
Casual observation of the ‘Confidence Label’ data showed there was no uniformity in the level of confidence in a given label. In this study, the matplotlib and seaborn packages were used to further understand if there was a grouping tendency in the ‘Confidence Label’ data. Using histograms of various bin width:



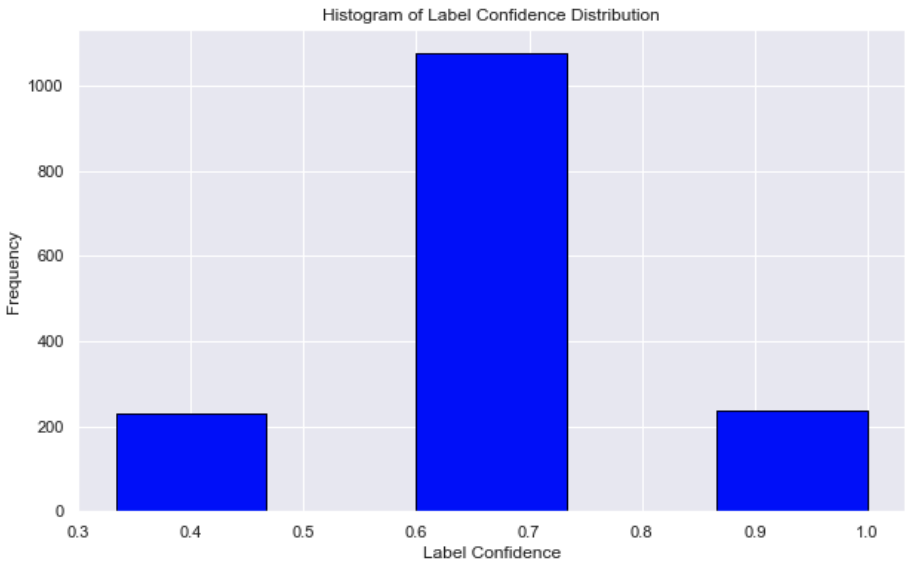
Referring to the top left histogram you can see a large portion of the data distributed centrally around the 0.6-0.7 mark. There was also a large distribution of data across the 0.9-1.0 mark. This suggested, that when looking simply at the distribution of data, the majority of data was over 0.6 label confidence.

Further analysis examined the distribution of ‘Confidence Labels’ across the different ‘label’ distributions. This was to get an understanding of the confidence across certain labels.

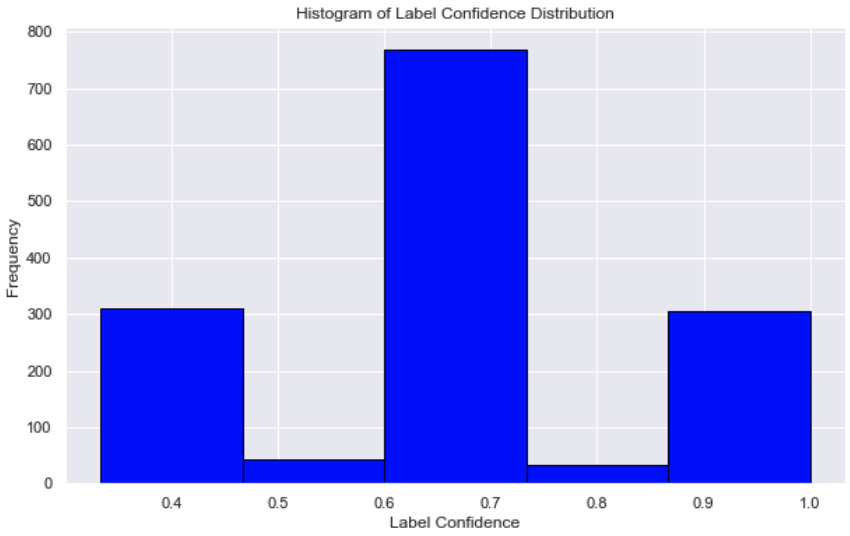
Positive:



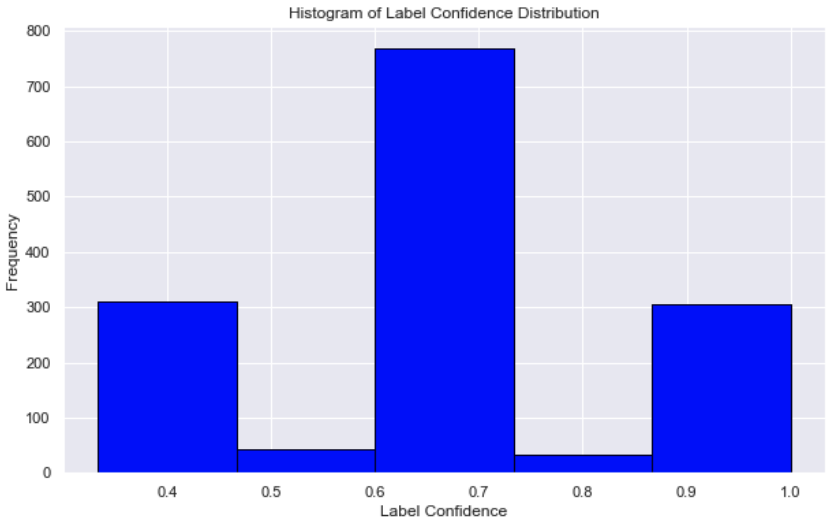
Neutral:



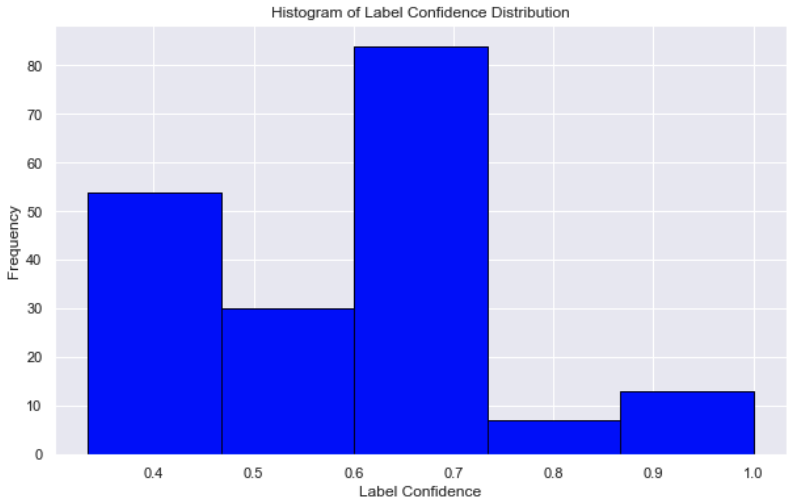
Negative:



Highly Positive:



Highly Negative:



Interestingly from the above, the highly negative and highly positive data labels have relatively poor label confidence.

Positively labeled data has quite a high level of confidence.

Negative and Neutral labelled data has a moderate level of label confidence with the central tendency of the observations to be in the 0.6-0.7 confidence range, and an approximately even distribution across the higher (close to 1.) and poorer (close to 0.4) confidence labels.

EDA Remarks

The EDA has shown that some weighting of labels and label confidence will likely be required throughout model development to improve overall accuracy.

The heavy weighting of observations to being positive will likely bias the models to classify an image as positive. However it would be inappropriate to simply exclude some of the positive images as that, in my view, would be manipulating the data set as we are unable to determine the reason for the bias. For example is it because the images selected were taken from sources that contain more positive images, or is there an inherent bias in people classifying images to classify them as positive. As we do not know the answers to theses questions, we must continue to use the full range of labels data.

Another consideration is the implications of lower levels of label confidence. I would consider anything below 0.5 to be low, and require that it be treated with care. A ‘Label Confidence’ below 0.5, to me, would suggest we are in fact more confident that the label is incorrect then correct. This is different to the uneven distribution of the data labels, as this is telling us that for certain labels (i.e. highly positive and highly negative) we have very little confidence that the observations are accurate. This is a concern as incorrect training data could largely impact the overall accuracy of the model. Therefore it is likely that I will consider excluding some lower levels of label confidence from the training dataset.

Models

Model Selection

Prior to selecting a model, a broad assessment was carried out on a variety of different models to assess their accuracy and likelihood of success.

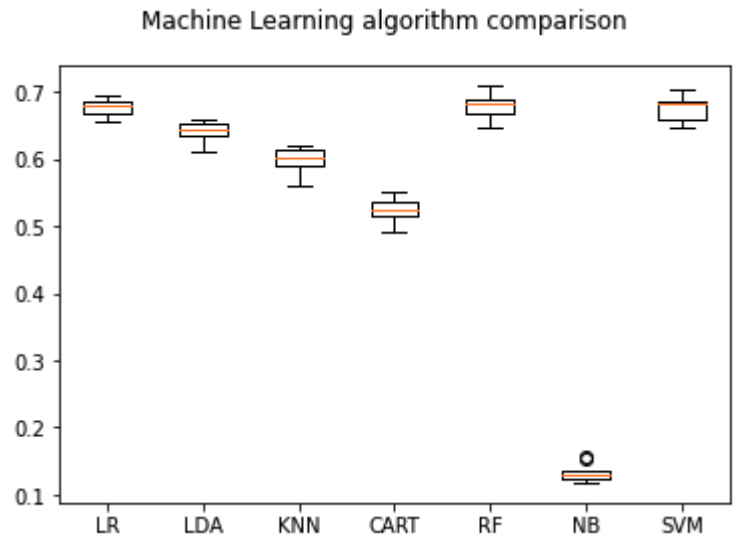
The example models provide in the assignment template were:

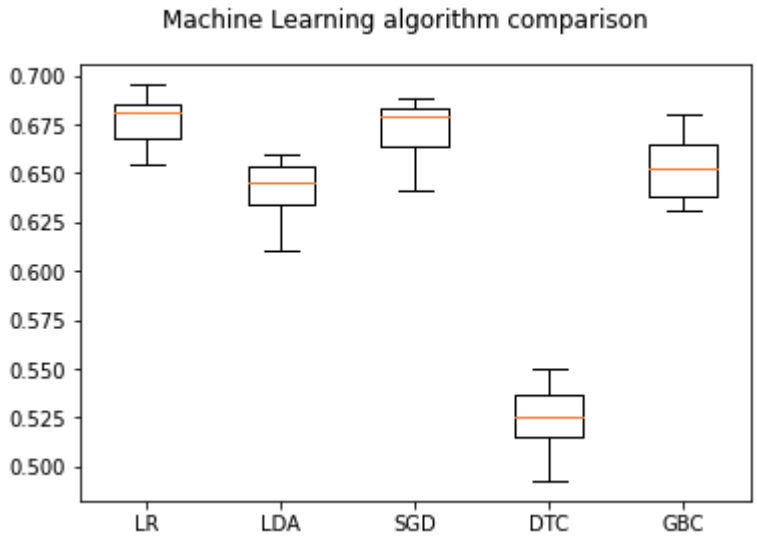
- Logistic regression
- Linear Discriminant Analysis
- K Nearest Neighbors
- Decision Tree Classifier
- Random Forest Classifier
- Gaussian Naive Bayes (NB)
- Support Vector Machine.

In addition to these models, the below models were also assessed for suitability.

- Stochastic Gradient Descent (SGD)
- Gradient Boosting Classifier

With the tuning of different parameters, differing levels of accuracy were observed, but broadly Random Forest, SVM and Logistic Regression were generating the most accurate models and thus were studied in greater detail.





Random Forest

Referring to the above machine learning algorithm comparison, the Random Forest was the most accurate so it made sense to explore it further. This has several parameters that can be tuned to investigate their influence on the overall accuracy.

Support Vector Machine (SVM)

SVM appeared to be second most accurate so will be utilised in the experimentation of the models. This is expected to perform close too, but not quite as well as the Random Forest model.

Logistic Regression

Logistic Regression was the third most accurate method, and thus was investigated further. It is expected to be the poorest performer of the three, however multiple parameters will be tuned to see what level of accuracy is able to be obtained.

Experiment Setup

As the base code was provided for the image preprocessing and to run the basic Random Forest, SVM and Logistic Regression models, certain parameters were selected to be tuned to assess what influence they had over the accuracy of the model. Specifically:

- The K-Fold in Cross Validation
- Pixel size of images
- Reducing the dataframe size based off of ‘Label Confidence’ variable
- The number of trees in the Random Forest model.

Computational Considerations

Using the Google Colab environment, these experiments took 1-2 hours to run due to the computational requirements. This impacted the number of experiments that could practically be run.

K-Fold CV

K-Fold would be shifted between 3, 5, 10 (default) and 20.

20 seems excessive, but I would like to validate this assumption. 3 and 5 are commonly used K-folds, so it makes sense to apply them (Brownlee, 2021).

Pixel Size

Adjusted between (500,500) being the default, (100,100) and (50,50). Pixels were reduced to improve the computational efficiency of the model. This variable was monitored to see if it had a degrading effect on the model. For efficiency, the lowest possible pixel dimension that did not negatively impact the accuracy of the model would be used.

Label Confidence DataFrame Reduction

This is expected to have a significant influence on the accuracy of the model. Criteria will be set to to ‘cut-off’ data below a certain certain ‘label confidence’ threshold.

These will be varied across a range of 0.4 to 0.75 based off of the EDA, and how the models fare.

This is done as I want to explore in detail the impacts of weighting the training data heavily towards the very reliable data, and see the impacts as I gradually reduce the confidence of the model.

Number of Trees

This section is only applicable to the Random Forest model. The following tree numbers will be assessed:

- 50
- 100
- 250
- 464* (Płoński, 2021)

Depending on the outcome, I may choose to tune further from there.

Experiment Results

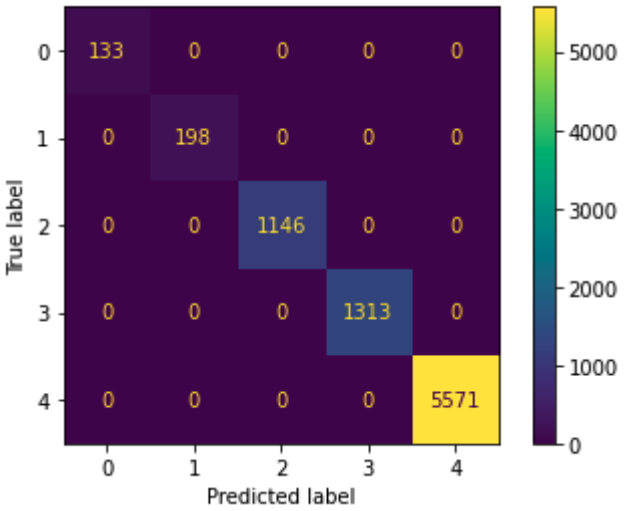
Random Forest

This particular model has the highest number of observations as it was performing the best and subsequently tuned the most to try and incrementally increase the performance.

K-Fold CV	Pixel Size	Label Confidence	Number of Trees	Accuracy
3	500	0.5	100	0.71863
5	500	0.5	100	0.71863
10	500	0.5	100	0.71863
20	500	0.5	100	0.71863

K-Fold CV	Pixel Size	Label Confidence	Number of Trees	Accuracy
10	500	0.65	100	0.71482
10	500	0.40	250	0.71768
10	500	0.5	300	0.72053
10	500	0.4	464	0.71673
10	500	0.5	350	0.72148
10	500	0.5	375	0.72148
10	500	0.5	400	0.72243
10	100	0.6	400	0.71673
10	500	0.6	400	0.71768

Confusion Matrix for optimum Model



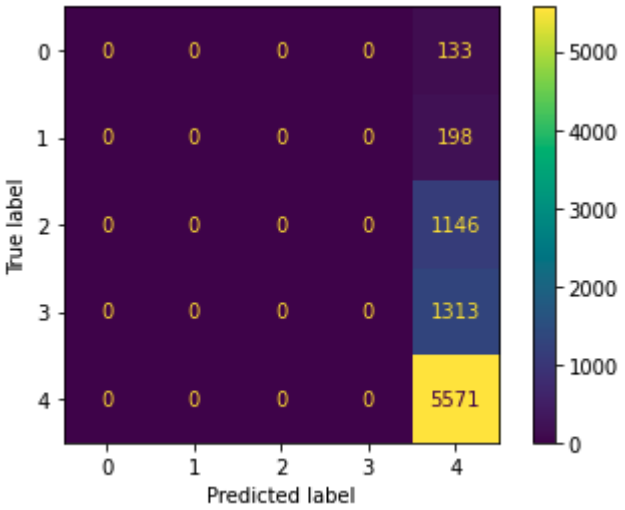
Overall, this model performed the best with the best in terms of accuracy when uploaded to the Kaggle data set. K-fold k selection did not seem to influence the model in a measurable way. Most influential was the combination of the number of trees and the different label confidence cut off points used.

The confusion matrix shows 100% accuracy which is actually a concern, as it suggest the model is overfitting (learning the right labels as opposed to the underlying relationships). However despite this concern, this model still performed the best.

SVM

K-Fold CV	Pixel Size	Label Confidence	Model Accuracy
3	500	0.5	0.71292
5	500	0.5	0.71292
20	500	0.5	0.71292
10	500	0.5	0.71292
10	100	0.6	0.71292

Confusion Matrix for optimum model:

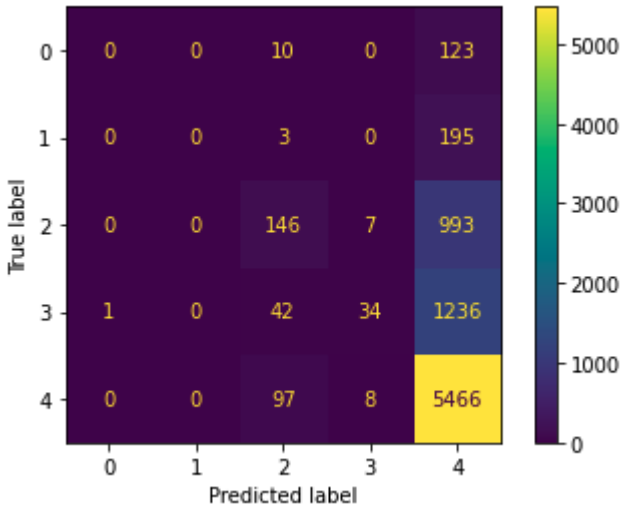


Assessing the SVM model simply by its accuracy would conclude in it being the 2nd most accurate model, consistently scoring 0.71292. However looking at the Confusion Matrix, the SVM is seen to simply classify every label as positive. Therefore the so-called accuracy of 0.71292 is in fact the proportion of labels in the validation data set that are positive. This model, in my view, is inappropriate as it fails to classify anything other than the positive label.

Logistic Regression

K-Fold CV	Pixel Size	Label Confidence	Number of Trees
4	500	0.5	0.70247
10	500	0.5	0.70247
10	500	0.6	0.70342
10	100	0.6	0.70057

Confusion Matrix for optimum model:



Of the three models, by accuracy the logistic regression model is the lowest scoring, with a high score of 0.70342.

However it does predict labels other than positive, so in my view it is a more successful model than the SVM model. Also the confusion matrix, whilst it does show a high number of misclassifications, with a tendency to classify positive over the other labels, it shows a more realistic distribution of results of a model that is learning (i.e. not overfitting as much as the logistic regression.

Experiment Evaluation

From reviewing the above results the random forest model performed the best over a range of parameters. The most accurate model had an accuracy of 0.72243 or 72.243%. This was achieved through:

- K-fold = 10
- Pixel size = 500,500
- Label Confidence minimum cut off of 0.5
- Number of trees = 400.

This was done using the features provided in the code template.

Further:

- K fold selection had very little impact between 4 and 10.
- Pixel reduction appears to influence the quality of the model, therefore pixels were left at (500,500). Lower pixel selection (e.g. 100,100) negatively impacted model accuracy.
- Label confidence heavily impacted the model accuracy, 0.5 label confidence as a minimum value appears to have yielded the best results.
- The number of trees for a Random Forest had a large impact with 350-400 being the optimal range of values.

Conclusion

The Random Forest models proved to be the most accurate across a range of parameters, scoring a highest accuracy of 72.243%. The SVM model was an inappropriate model over experimentation as it only classified photos as positive. Finally, the Logistic Regression model still performed well, getting a top classification of 0.70342

The main challenge with the input data was that it was heavily skewed towards positive data. And further, the positive labelled data had a higher label confidence then the less numerous, other data labels. This was inherently biasing the model and ideal in future studies, a more balanced data set would be obtained. Also, further improvements could likely be made with greater feature extraction and further EDA.

Despite these challenges an overall top accuracy of 0.72243 (or 72.243%) model accuracy was, in my view, a good outcome.

References

Brownlee, J. (2021). How to Configure k-Fold Cross-Validation. Retrieved 30 November 2021, from https://machinelearningmastery.com/how-to-configure-k-fold-cross-validation/

Płoński, P. (2021). How many trees in the Random Forest?. Retrieved 30 November 2021, from https://mljar.com/blog/how-many-trees-in-random-forest/

sklearn.linear_model.SGDClassifier. (2021). Retrieved 4 December 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html

sklearn.ensemble.GradientBoostingClassifier. (2021). Retrieved 4 December 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html