Cancer Detection

Springboard - DSC

Capstone Project 1

by Morgan Fry

May 2020

Introduction

Histopathologic cancer detection is the identification of cancer cells in tissue samples. In brief, tissue is stained with a contrast dye that reacts with cancerous cells and then are visually identified under a microscope. If machine image classification can improve on the performance of humans in this regard, it can lead to better patient outcomes by detecting cancer more accurately and earlier.

The data for this project is a subset of the PatchCamelyon (PCam) dataset, assembled for the Kaggle competition Histopathologic Cancer Detection. It is a set of images of lymph node sections, with binary labels indicating the presence of metastatic tissue. The PCam set contains some duplicate images to balance the numbers of positive and negative samples. The Kaggle set used here has the duplicates removed so as to deal with the original data.

Using some simple classification algorithms implemented in Python using the Scikit-Learn package, I was able to achieve about a 70% accuracy in classifying these images.

All code for this project can be found in the following Github repository: https://github.com/morganfry/springboard/tree/master/Capstone%20Project%201

Approach

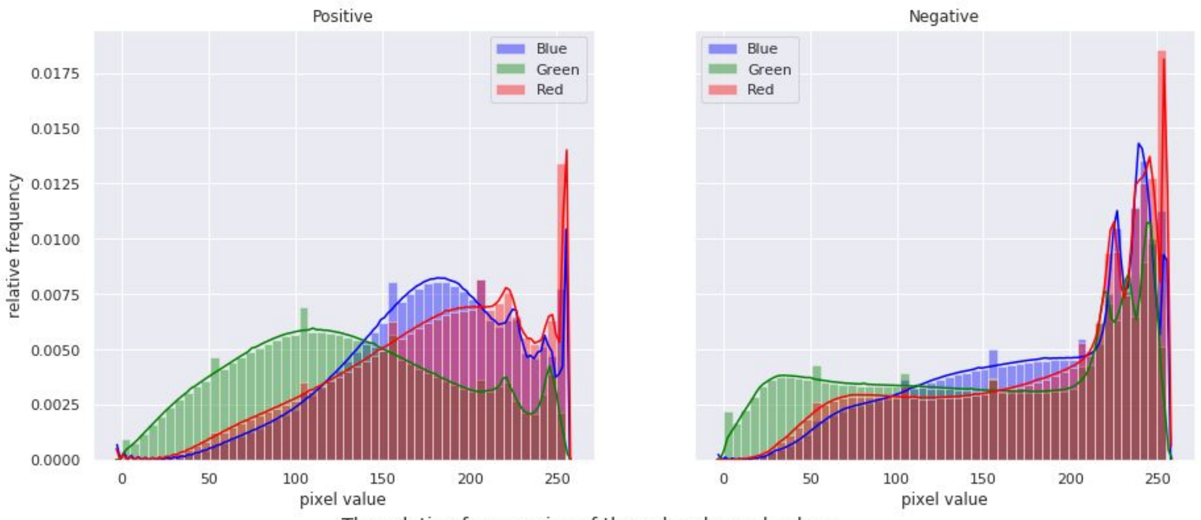
Data Acquisition and Wrangling

The images and labels were downloaded from the PCam Github repository. The images were loaded into a Python dictionary with their ID values as keys and 3-d Numpy arrays as values. Their labels were loaded into a similar dictionary, ID's as keys and labels (1 or 0) as values.

Data Storytelling and Inferential Statistics

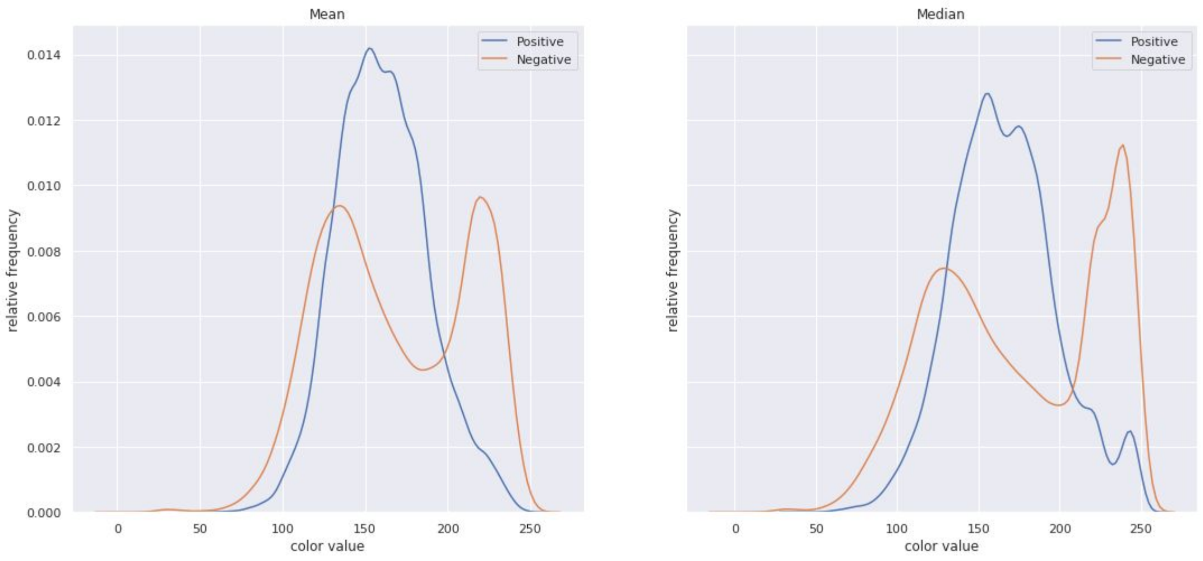
During the preliminary data exploration, a few key observations were made:

- The dataset is made up of 40.5% positive and 59.5% negative images. This imbalance will have some implications for model training.
- The values of the color channels were examined, and there was a noticeable difference between the plots of the positive and the negative sets as seen in the following:

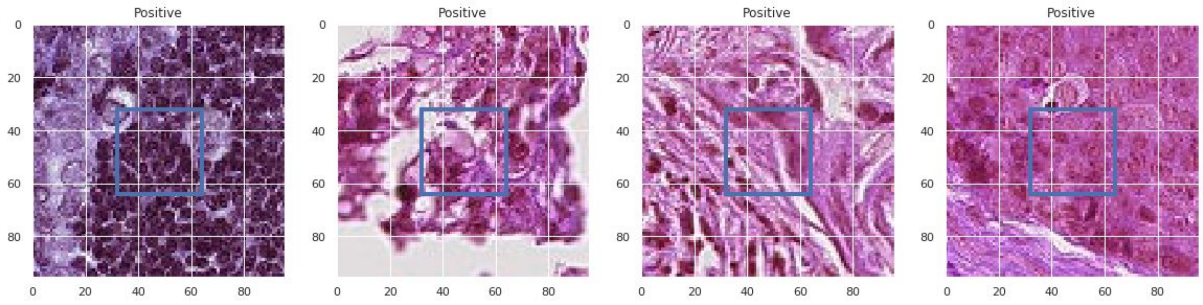


The relative frequencies of the color channel values.

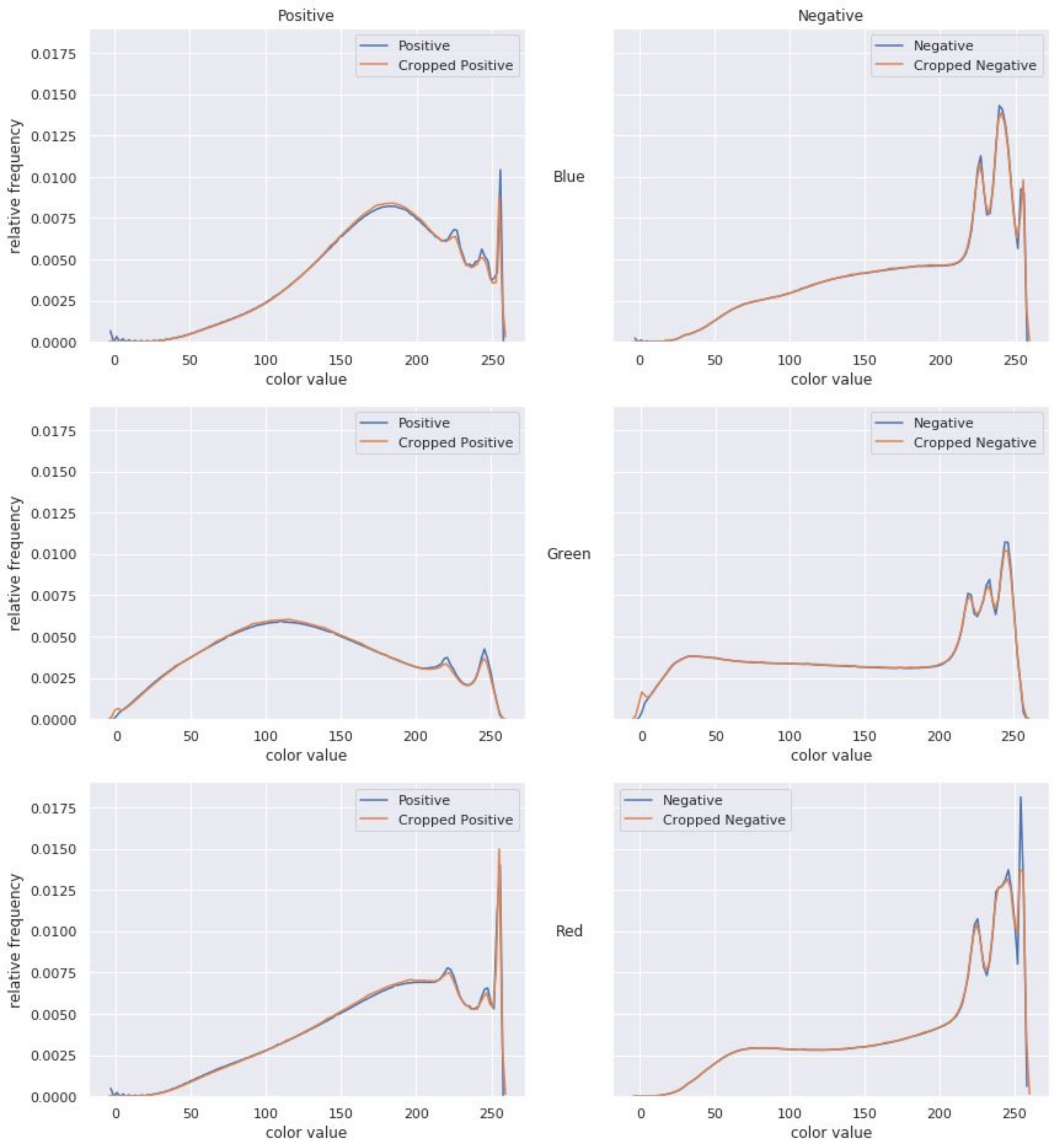
- There was also a distinct difference in the distribution of the aggregate values of the images (example below of the mean and median values for the image sets):



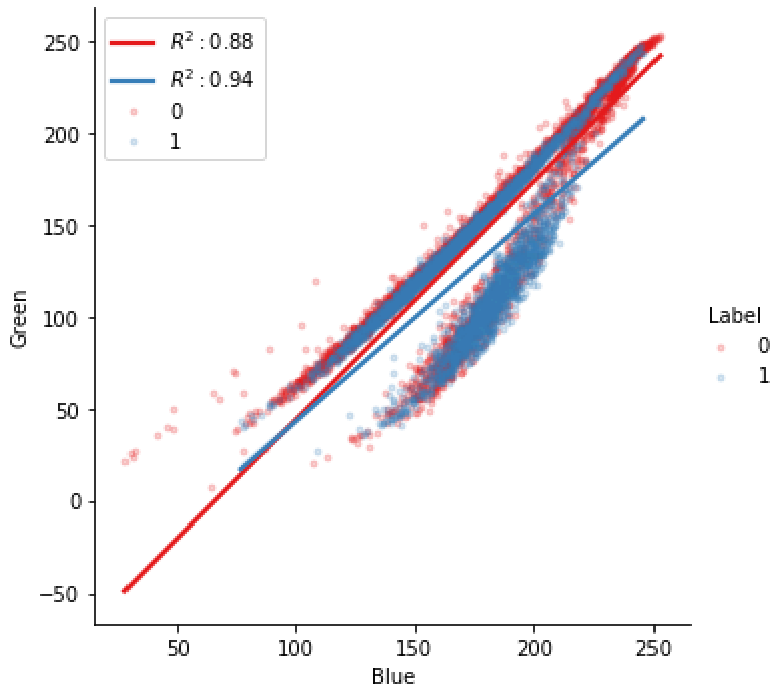
- It was also noted that in the documentation for this dataset, a positive label denotes that an image has >= 1 pixel of cancer cell in the center 32x32 pixel square of the image:



- These center areas were examined to establish whether the images could be cropped to reduce the size of the dataset needed for processing. As we can see below, the distributions of the pixel values are substantially the same for both cropped and full images. Considering this similarity, and as an image label of ‘1’ means the relevant feature is within the proposed cropped area, I worked with cropped images for modeling.



- It was also noted that there was a difference in the regression lines between positive and negative sets for each of the color groups, for instance:



- I observed that the difference between the positive and negative groups across the blue/red and blue/green planes is greater than across the red/green plane. An OLS Regression of the blue channel vs green and red channels combined shows a R^2 of 0.818 in the positive set and 0.93 in the negative set.

Baseline Modeling

For the baseline model, logistic regression was chosen, as implemented by Scikit-Learn. I used a train/test split, stratified to reflect the imbalance in the dataset.

When considering what metrics are important for this case, there are a few factors to consider:

- Both false negatives and false positives matter. More false positives among a population in which the positive condition is rare means that even with a high accuracy a positive result can still mean a low chance of the condition tested for. More false negatives are even more important as that can mean sick people do not get treatment they need.
- There are about 50% more negative than positive samples in this dataset, so as you see above there is much better recall for negative samples than positive.

Given these factors, F1 score for the positive class and area under the ROC curve (AUC) were decided on as the most important performance metrics.

The initial results for the baseline classifier were as follows:

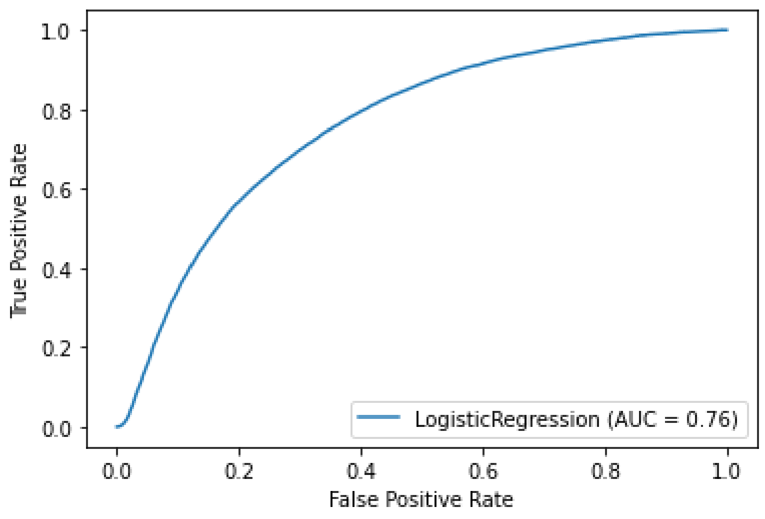
```
Classification Report (training set)
             precision    recall  f1-score   support

          0       0.72      0.83      0.77     98181
          1       0.68      0.52      0.59     66837
```

```
Classification Report (test set)
             precision    recall  f1-score   support

          0       0.72      0.83      0.77     32727
          1       0.67      0.52      0.59     22280
```

ROC curve (test set)

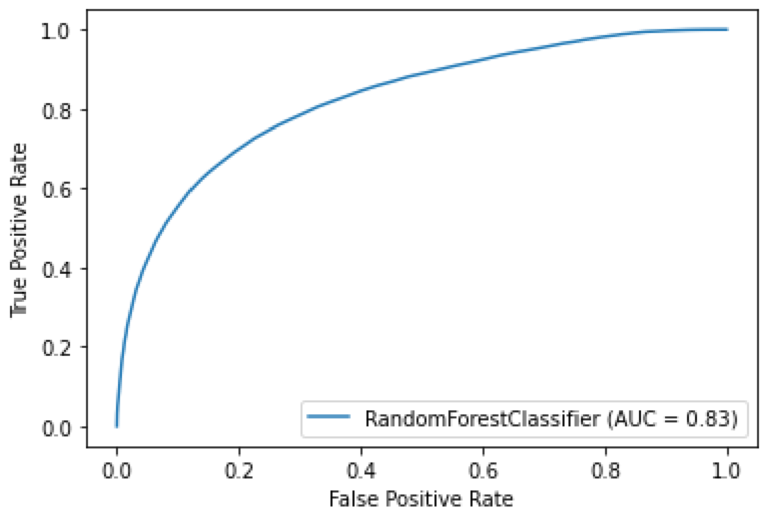


We can see from the similarity of the results on the train and test sets that there is no significant overfitting.

On the assumption that one of the strategies more commonly used for image classification might yield better results, I also tried both a Random Forest Classifier and MLP neural nets, both with default parameters. The Random Forest Classifier achieved better results on the test set than the Logistic Regression Classifier on the same train/test split:

```
          precision    recall  f1-score   support

          0       0.76      0.88      0.82     32727
          1       0.77      0.59      0.67     22280
```



Scikit-Learn’s MLPClassifier proved too time-consuming on the full dataset (on a Google Cloud instance with 16 CPU cores it ran for 24 hours with no results), so further work with that model was abandoned. The Random Forest Classifier was used for further modeling.

Extended Modeling

One of the issues noticed in the baseline model was the difference between performance of the positive and negative groups, with e.g. recall of 0.88 vs 0.59. It is reasonable to suspect this is a result of the imbalance in the dataset (with 40.5% positive and 59.5% negative samples). So to improve on the results of the baseline models I implemented and compared two different strategies for balancing the data, random undersampling and image augmentation.

For the undersampling, I used the a Random Under Sampler from Imbalanced Learn[1] to implement a pipeline of the Random Forest Classifier using several different random undersamplings of the majority class, and used them to construct a majority vote classifier. This yielded an improvement:

	precision	recall	f1-score	support
0	0.80	0.81	0.81	32727
1	0.72	0.71	0.71	22280

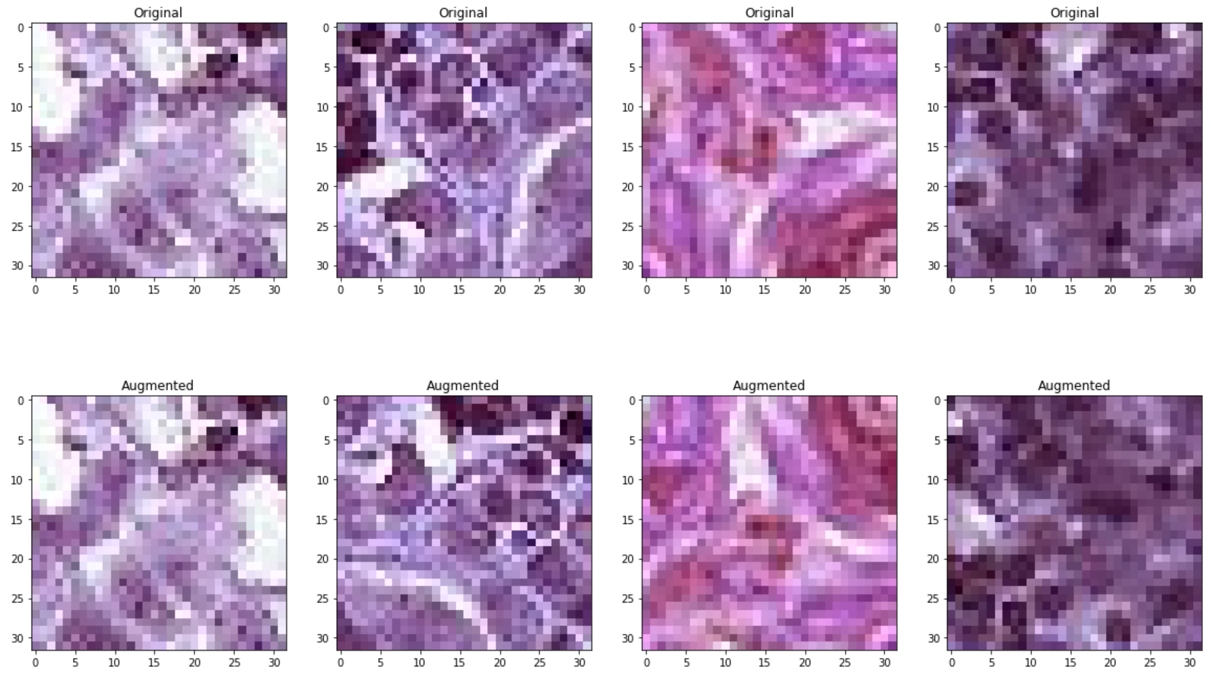
To see if augmentation of the minority portion of the training set would yield an improvement over undersampling of the majority set, I implemented image augmentation using the package Imgaug[2]. A random sample of the minority set was taken, the images changed using a randomized combination of transformations, and these added to the dataset. It is crucial that any transformations applied to the images do not destroy the

[1] Imblearn
[2] Imgaug

features needed for accurate classification. In a real-world scenario, this would be a point to consult with people with domain knowledge. For instance, doctors or technicians that normally would be examining these slides visually may be able to help identify what sort of transformations are useful and which hinder classification . To play it safe, I only used the following transformations:

- Rotation by 90, 180, or 270 degrees
- Flipping horizontally or vertically

For example:



No change	Rotation 90 deg	Rotation 90 deg	Rotation 270 deg

The Random Forest classifier performed similarly when trained on the augmented set to how it performed when trained on the undersampled set. The same test set was used to evaluate each model here.

	precision	recall	f1-score	support
0	0.80	0.80	0.80	32727
1	0.71	0.70	0.71	22280

Hyperparameter Tuning

To try to improve the f1 score and AUC, I implemented hyperparameter tuning of the Random Forest classifier. The relevant parameters examined were:

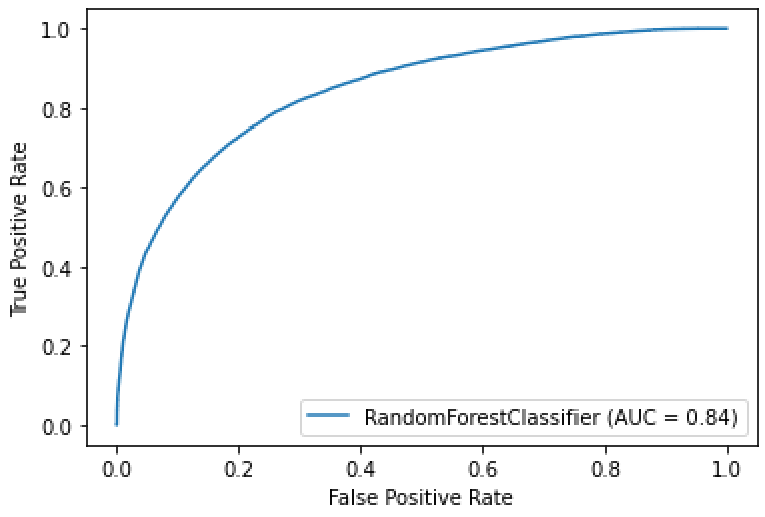
- n_estimators
- min_samples_split
- min_samples_leaf
- max_features
- criterion

A Random Search with cross-validation was used to try 30 different combinations of parameters and the best performing parameters were chosen. These were:

- n_estimators: 1466 (default 200)
- min_samples_split': 5 (default 2)
- min_samples_leaf: 4 (default 1)
- max_features': 'sqrt' (default sqrt)
- criterion: 'entropy' (default gini)

Small improvements in recall and AUC were achieved:

	precision	recall	f1-score	support
0	0.83	0.77	0.80	32727
1	0.69	0.76	0.72	22280



Findings

To summarize the performance of the classification models implemented, see the following table:

Model	Class	Precision	Recall	F1 Score	AUC	Support
Baseline Logistic Regression	0	0.72	0.83	0.77	0.76	32727
	1	0.67	0.52	0.59		22280
Baseline Random Forest	0	0.76	0.88	0.82	0.83	32727
	1	0.77	0.59	0.67		22280
Undersampling Ensemble	0	0.80	0.81	0.81	0.76	32727
	1	0.72	0.71	0.71		22280
Image Augmentation	0	0.80	0.81	0.79	0.83	32727
	1	0.72	0.70	0.72		22280
Hyperparameter Tuning with Image Augmentation	0	0.83	0.77	0.80	0.84	32727
	1	0.69	0.76	0.72		22280

We can conclude that balancing the dataset either by undersampling or by image augmentation yields similar significant improvements.

Conclusions and Future Work

Modest success was achieved but the model may not be not effective enough to be of practical use. Real world use of a model for identifying cancer might be to replace human visual detection, or to save time by narrowing a set of samples that a human has to examine. So far, the performance achieved would not help a clinician's decision making, there are too many errors. For the Kaggle competition that this dataset was taken from, a successful model was defined as having 0.95 or greater AUC. The Random Forest Classifier trained on the augmented image set currently achieves 0.84.

Future work should include:

- Hyperparameter tuning of the classification algorithms.
- New classification model. Convolutional Neural Nets, for example, are known to work on this class of problems and training can be parallelized on a GPU to overcome the computational demands encountered with the MLP classifier in this project.

Recommendations for the Client

- Use a balanced dataset for future modeling
- Focus future modeling work on a deep learning model.

- When a model is developed that does work with the prescribed degree of accuracy, it may be able to be used in practice.
- Meanwhile, investigate if it is possible to implement a hybrid pipeline that would allow healthcare personnel to pre-classify cases so that only those that cannot be clearly classified are screened by specialists.

Consulted Resources

1. Scikit-Learn
2. Imbalanced-Learn
3. OpenCV
4. Imgaug
5. PCam
6. Kaggle