Image Classification using XGBoost Classifier and LightGBM Classifier

Soh Hong Yu  
Diploma in Applied Artificial Intelligence and Analytics  
School of Computing   
Singapore Polytechnic

***This technical paper aims to conduct a supervised learning approach to classify the type of flower. We aim to create a machine learning classifier that is able to identify the type of flower. We will be using 2 models (XGBoost and LightGBM) and doing a comparison between them to see which model is better at classifying the images. We are using VGG Imagenet as a feature extractor.***

***Keywords—image, machine learning, python, xgboost, vgg imagenet, lightgbm, comparison, flower, supervised learning***

# Introduction

Based on Globenewswire, the Floriculture Market is valued at US$ 49.8 billion in 2022 and by 2029 there is a surge in valuation of around US$ 80.5 billion [1]. With the floriculture industry growing rapidly, being able to successfully classify flowers is very important. Classifying flowers manually takes a lot of time and manpower which can be detrimental to a business. Furthermore, Gradient boosting has gained a lot of traction in recent years. This is primarily due to the improvement in performance offered compared to other algorithms. In this technical paper, we aim to compare the different gradient boosting methods and which model is better at classifying images.

# Related Works

In the past there have been similar works of classifying flowers.

An earlier investigation of this problem by Hiary, H., Saadeh, H., Saadeh, M. and Yaqub, M. [2]. They used deep learning techniques like convolutional neural networks (CNNs) and binary segmentation methods. And trained it against 3 different datasets, Oxford 102, Zou–Nagy, Oxford 17.

Another investigation was conducted to compare the difference between the different boosting methods [3]. They used the adult dataset and compared the results from each. The best model is found to be the LightGBM model, which obtained an AUC score of 0.861501 compared to XGBoost which obtained an AUC score of 0.861398. Although the difference between the AUC score is not very significant, the time taken to execute the models is very significant. LightGBM completed executing the code at 00:00:00.283759 compared to XGBoost which completed execution at 00:00:02.047220. This is a 7 times difference which means LightGBM is a better model for large datasets.

# Methodology

## Data Set

The data set, collected by Tung, K [4], has been made publicly available on the Harvard Dataverse. The data is from the Python distribution’s open-source flower images.The data contains 3 folders separating the images into the test, train and validation folders. In each folder, there are subfolders with the labels of the flowers. There are 5 different flowers, daisy, dandelion, roses, sunflowers and tulips. There are 130 files in the test folder and 3,684 files in the train folder. As there is no features in the images, an external library, VGG16 Imagenet, from Keras is used.

# Image Processing

Using the glob and os libraries from Python, we are able to loop through the different folders in the dataset. We use the cv2 library to read the images and recolour the images using RGB. The cv2 library will read the image by pixels and store it in a NumPy array. We reshape the images into 224 x 224 so that we can have a better comparison. After loading the images, the size of the total array is around 6MB and has a shape of (3684, 224, 224, 3). Each element in the image array is on an RGB colour scale, each component contains an unsigned integer number in the range 0 to 255, the range that a single 8-bit byte can offer. Because of the different colour channels, in theory, each pixel can be 16,777,216 different forms of colour.

Since the pixel intensity is always between 0 to 255 inclusively, we can solely divide each pixel by 255 to normalize the image as the minimum is a constant 0.

# Feature Engineering

As the models are unable to understand and predict based on the pixels of the images, we would need to use an additional library from Keras called VGG16. VGG16 has a model called Imagenet which is a pre-trained model to predict images [5]. Based on Fig 1, we will run the model and remove all the training layers of the VGG16 model. Afterwards, we use the VGG Model to predict the values using the image pixels. We reshape the features into (3684, 25088) that are extracted and store them back to X\_for\_training.

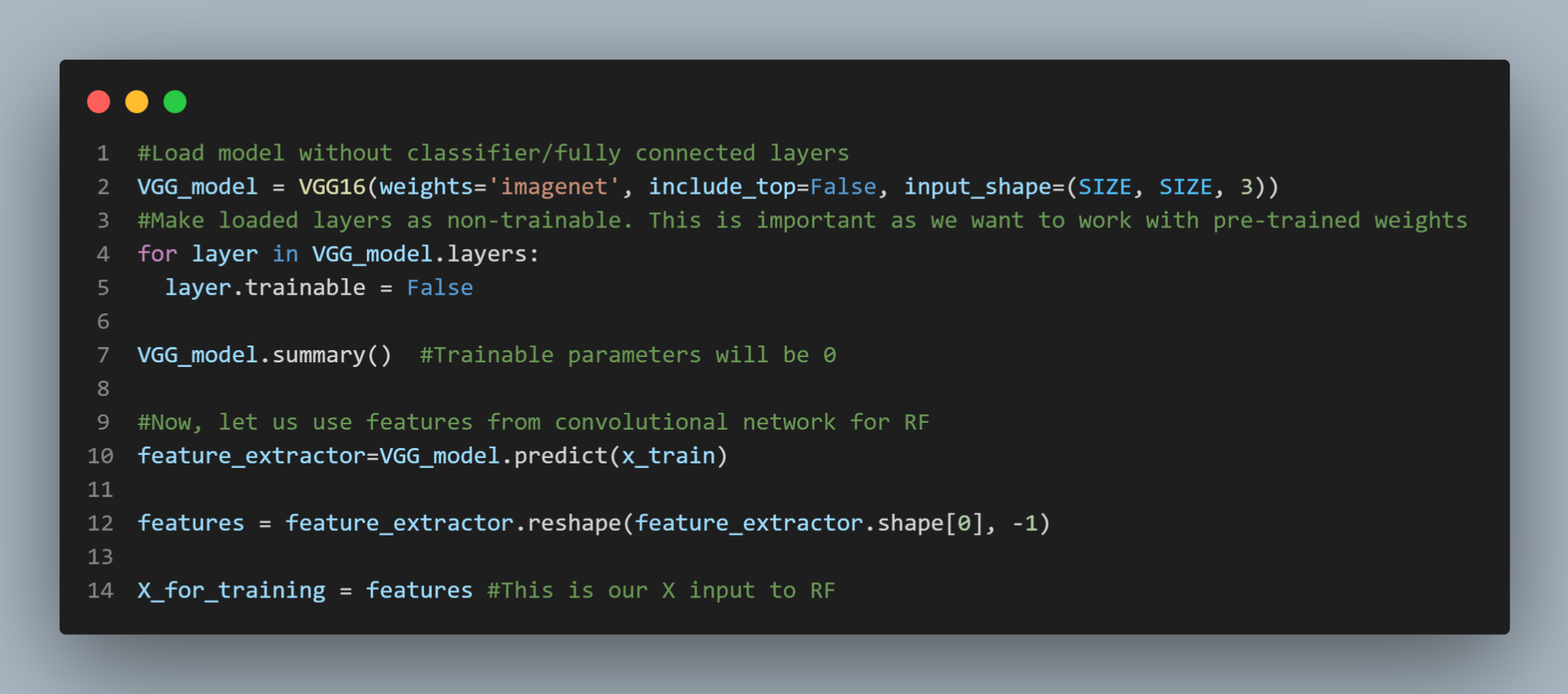


Fig. 1. Code implementation for feature engineering

The VGG16 Model has 16 layers that have weights. There are around 138 million parameters which is a pretty large network which means running the VGG16 takes a longer time [6]. Using Fig. 2., It works by “squeezing” data out by condensing the image and adding more depth to the image.

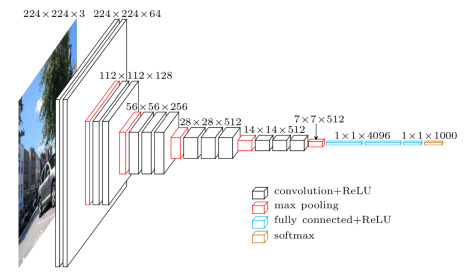
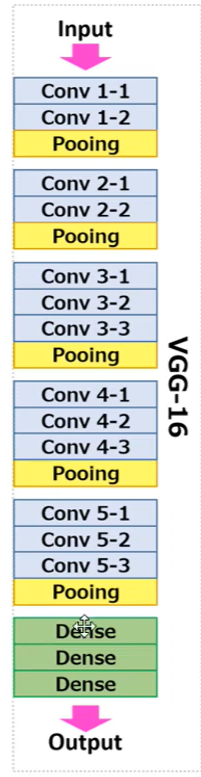


Fig. 2. VGG16 Architecture MapFig. 3. Simple VGG16 Architecture Map

We added the include\_top=False parameter, and we remove the dense layers to make the VGG16 model run faster [Fig. 3.].

# Gradient Boosting

First, let’s start off with what is boosting. Boosting combines multiple underlying models and uses a voting technique to determine the final prediction. Instead of training the models separately, boosting trains models together. Each new model will be trained to correct the error of the previous model and the output will be given a higher or lower weight depending on the prediction. The weight are averaged after running all the models to produce a final outcome.

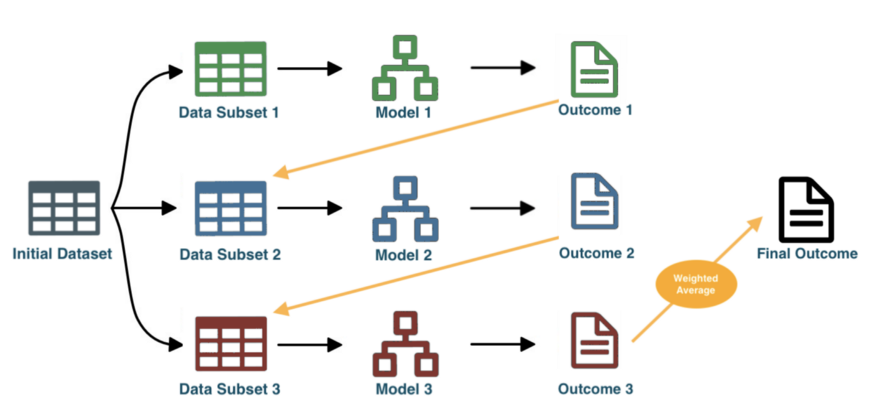


Fig. 3. Ensemble Learning example with the Boosting method, using weighted-Average strategy [7]

Back to Gradient Boosting, it is a boosting method which uses the gradient descent algorithm. Gradient descent is an algorithm which finds a local minimum or maximum of a function. From mathematics, we know to calculate the minimum point of a curved function, we need to take the differentiation of the function and make it equal to 0.

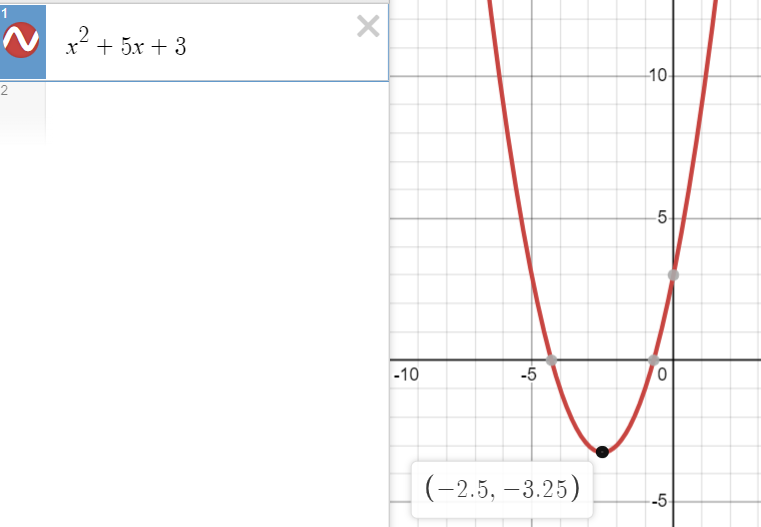


Fig. 4. Parabola graph example

For example take Figure 4, it is a parabola curve with the function of . To get the minimum point of the graph which is (-2.5, -3.25). We will take the differentiation of the function into the following equation . To find the minimum point, we let Therefore,

.

From here, we can clearly tell the minimum point of the function. Next, we need to check if the function is convex. This which means for any 2 points in the graph that are connected using a line segment lays on or above the curve and does not cross the graph. Mathematically, we check if the second derivative is larger than 0. Using the previous example, the second derivative is more than 0

Gradient Descent basically calculate the next point using the gradient at the current position and scales it by a learning rate and subtracts obtained value from the current position. In summary, Gradient Descent chooses a starting point and calculate the current gradient. It makes a scaled step in the opposite direction to the gradient and repeat until it met the maximum number of iterations and step size is smaller than the tolerance. [8] The easiest way to visualise this is to choose a point to put a ball and make the ball slide down to the minimum point of the function. Combining the ideas of Gradient Descent and Boosting will give you Gradient Boosting. [9]

1. XGBoost

XGBoost is also known as eXtreme Gradient Boosting. XGBoost is an algorithm that is similar to a decision tree, it uses the Classification and Regression Tree (CART) model which combines both the Classification Trees and Regression Trees. [7] The model combines the following kinds of boosting. Gradient Boosting, Stochastic Gradient Boosting as well as Regularized Gradient Boosting. The training proceeds by adding new trees to the prior tree so that it can predict the residuals of the prior trees and make the final prediction. XGBoost uses the Lasso (L1) and Ridge (L2) regularisation methods. This can be illstruatioed by Fig. 5.

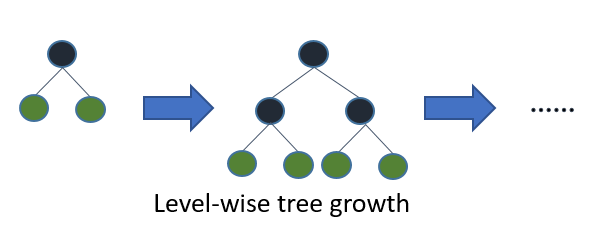


Fig. 5. XGBoost Algorithm Tree Growth

1. LightGBM

LightGBM is also known as Light Gradient Boosting Machine. LightGBMt is very similar to XGBoost by using decision trees for ranking and classification. In contrast to the level wise growth, LightGBM carries out a leaf-wise growth.

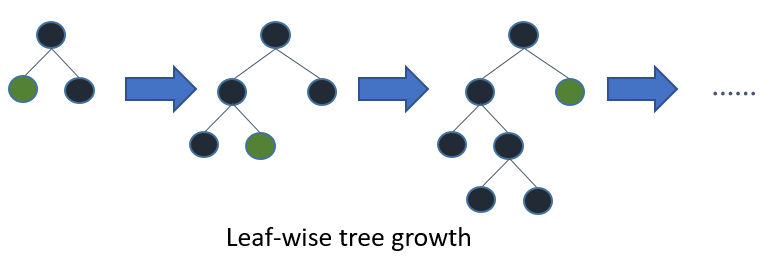


Fig. 6. LightGBM Algorithm Tree Growth

1. Difference between Models

LightGBM is faster than XGBoost to train. However, XGBoost is able to handle and prevent overfitting through the tree pruning using the Lasso and Ridge regularisation. The reason why LightGBM is faster is due to the method of tree growth. Leaf-wise tree growth is faster than Level-wise tree growth as Leaf-wise does not grow unnecessary trees while Level-wise grows all the possible trees.

1. Implementing Models

To implement the models, we import xgboost and lightgbm as xgb and lgb respectively. As we are solving a classification problem, we will be using xgb.XGBClassifier() and lgb.LGBMClassifier(). We will fit the X\_for\_training variables as well as the target column of y\_train into the models.

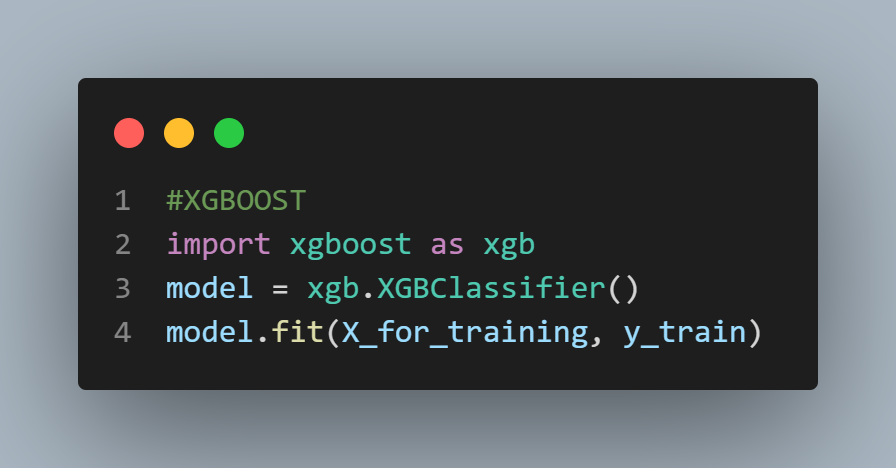


Fig. 7. Code implementation for training model using XGBoost



Fig. 8. Code implementation for training model using LightGBM

1. Results and Discussion

After training the model, we will be using the x\_test and extracting the features using VGG16 and using it to do prediction.

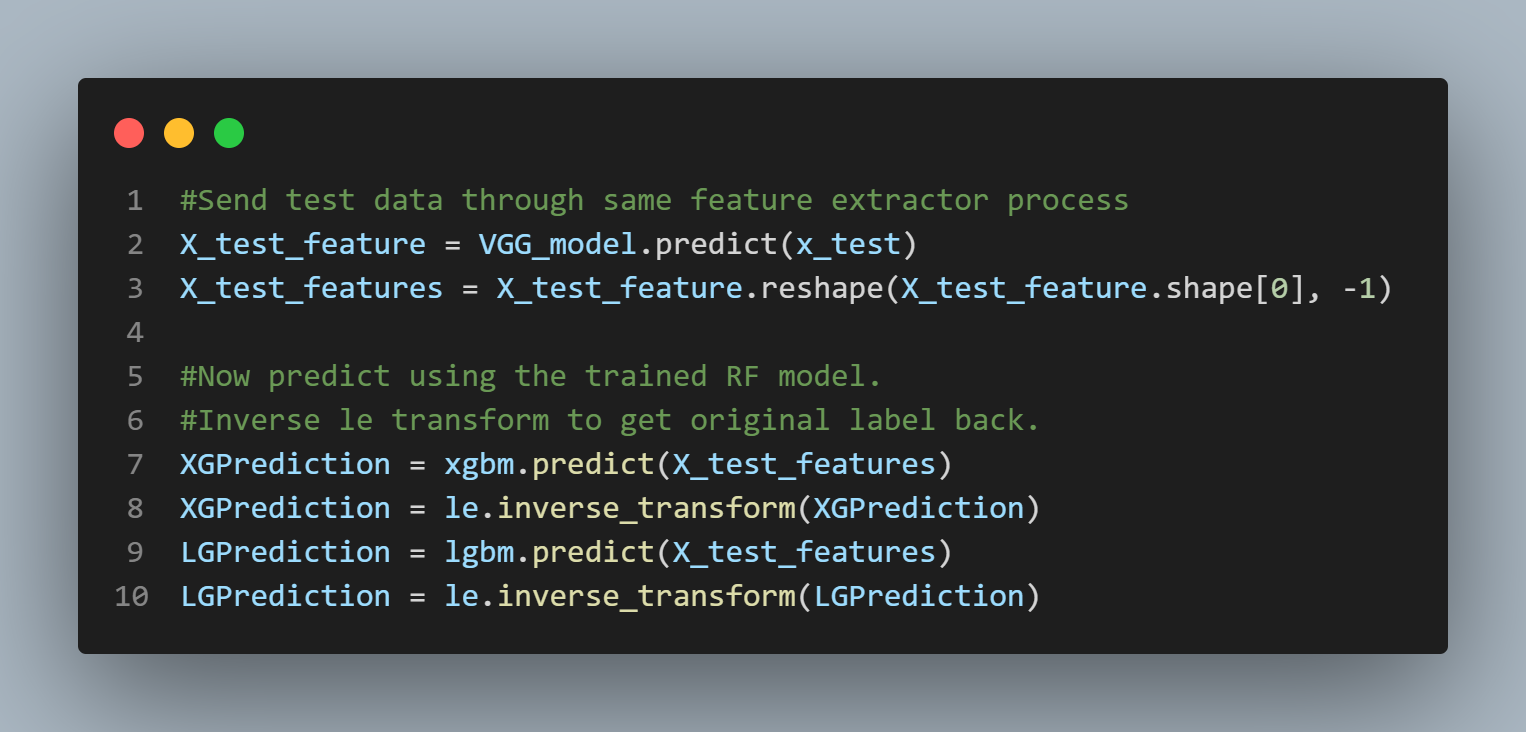


Fig. 9. Code implementation for model prediction

We use the Label Encoder to inverse transform of the labels into the actual names of the flower. Next, after we completed prediction.

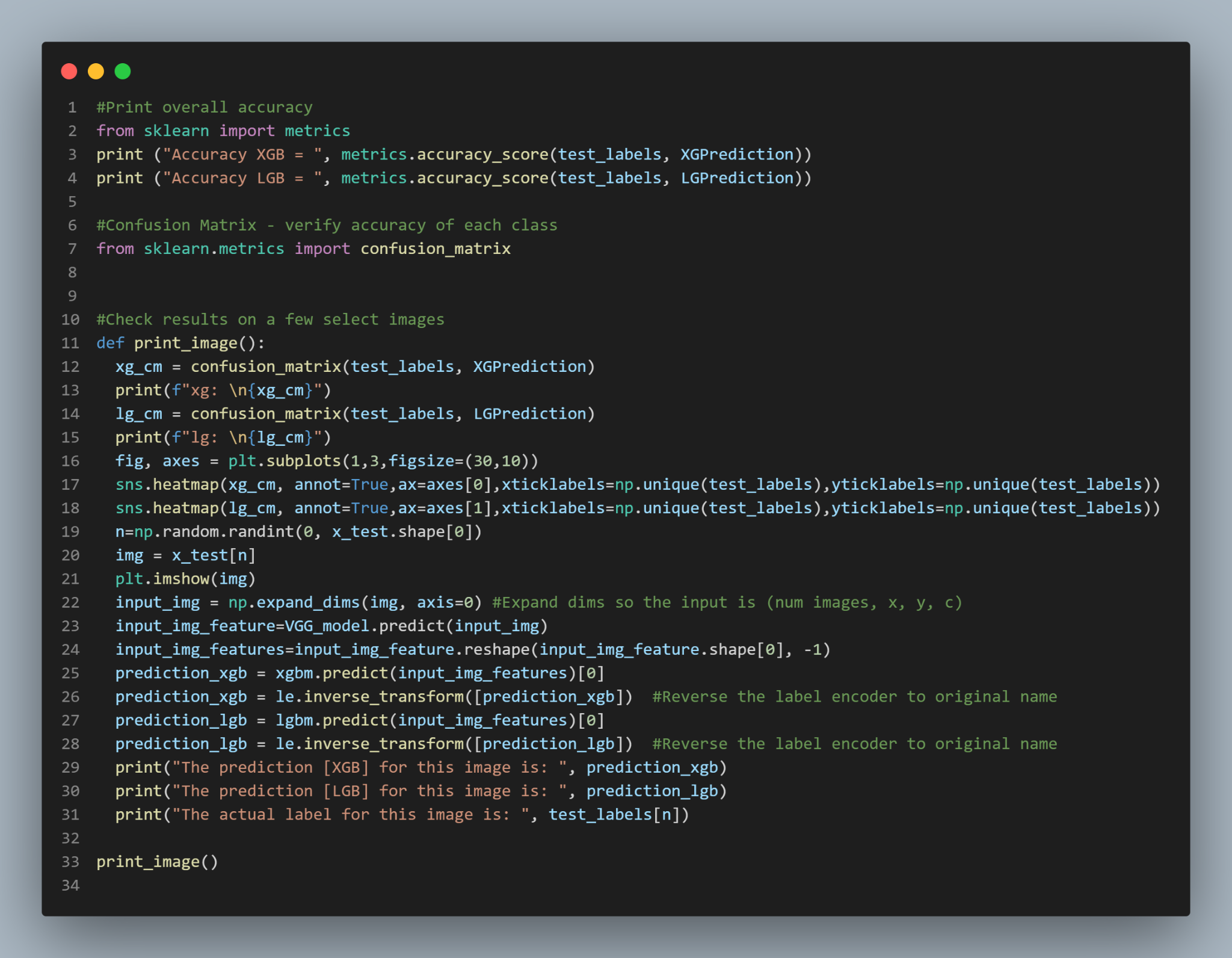


Fig. 10.Code implementation for model accuracy

We use sklearn’s metrics accuracy score to check the for the accuracy. We also used the classification\_report from sklearn’s metrics.

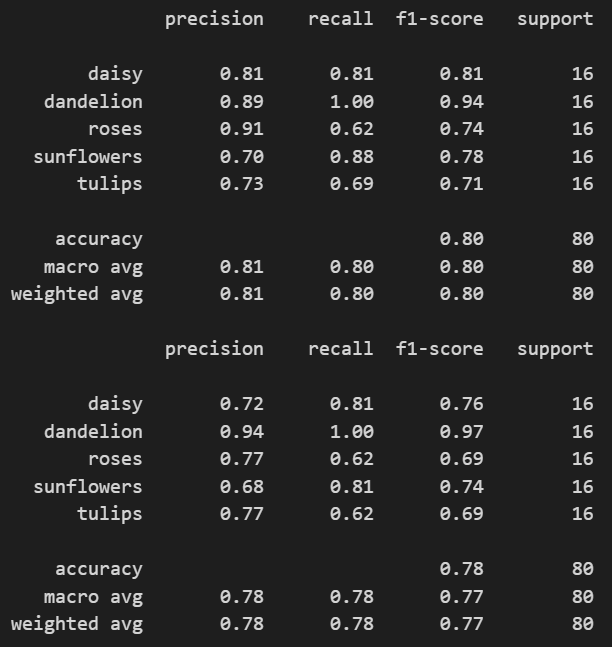


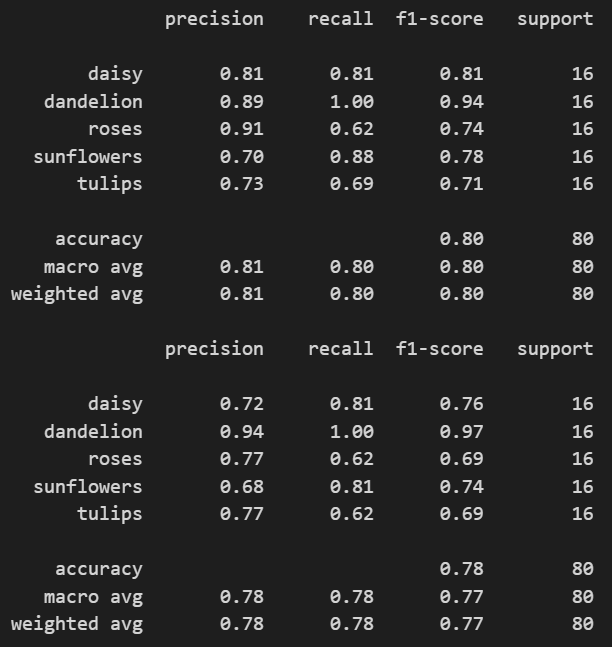
Fig. 11. Classification report of XGBoost

Fig. 12. Classification report of LightGBM

Based on the classification report, this shows that the accuracy of XGBoost is higher than LightGBM. This means that XGBoost is a better classifier compared to LightGBM. By comparing the Confusion Matrix, most of the predicted classes are the same and the False Negative of LightGBM is higher than XGBoost.

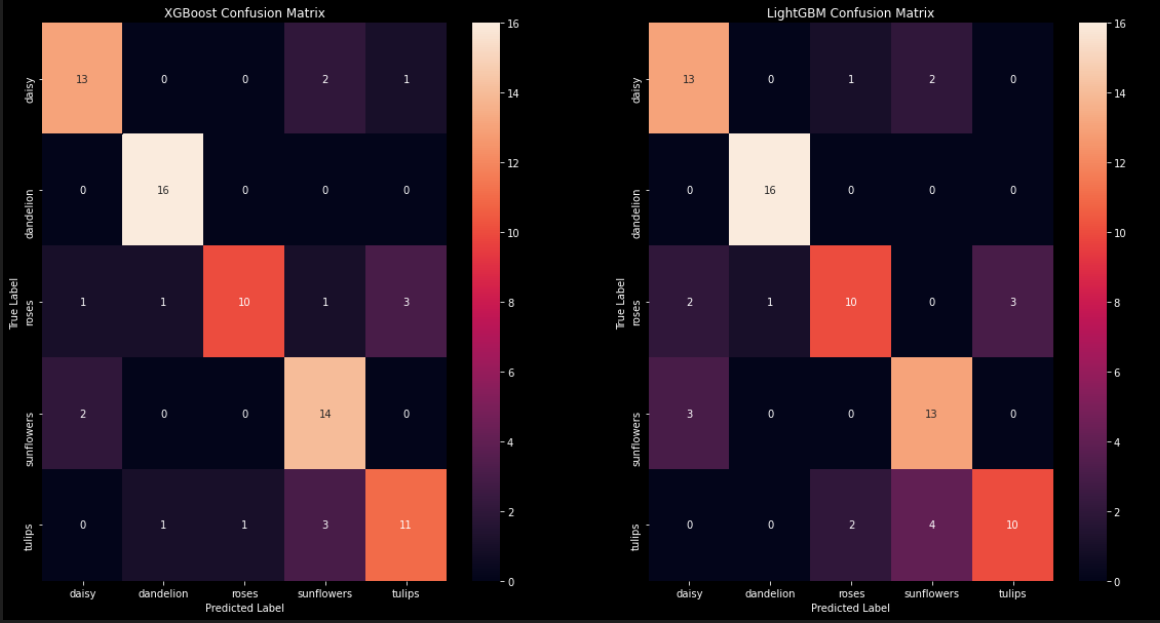


Fig. 13. XGBoost Confusion Matrix

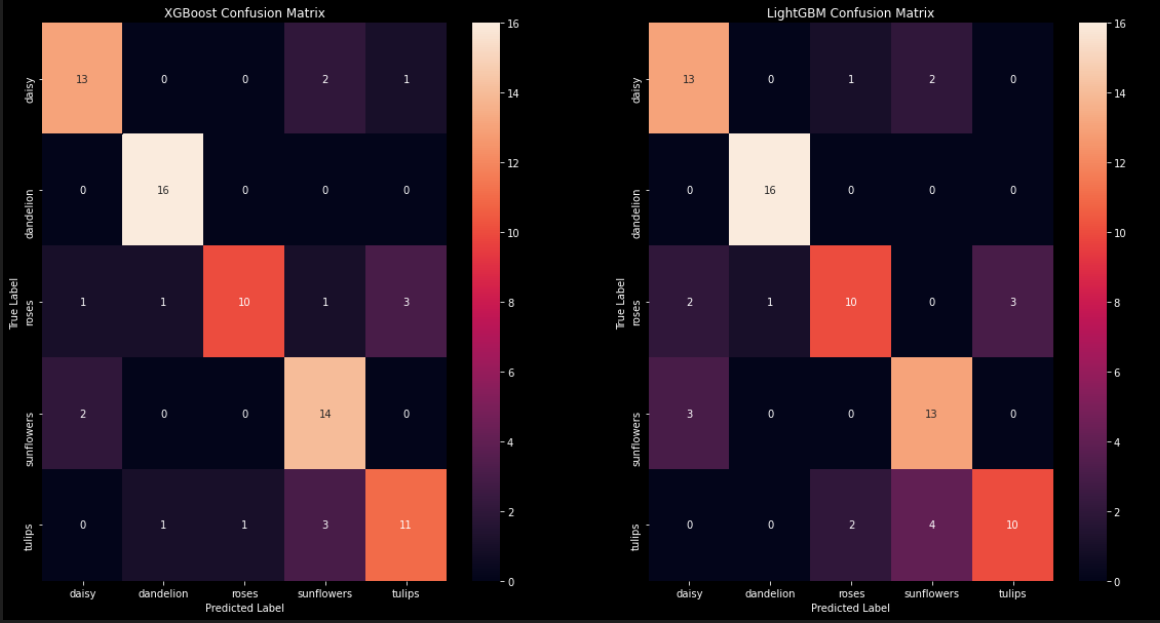


Fig. 14. LightGBM Confusion Matrix

From this results, we can see that XGBoost is a better model than LightGBM. However, the training time can be a factor of concern. Based on Fig 15, XGBoost took around 9 minutes and 31 seconds while LightGBM took around 7 minutes and 20 seconds. This difference is due to the tree growth method mentioned above.

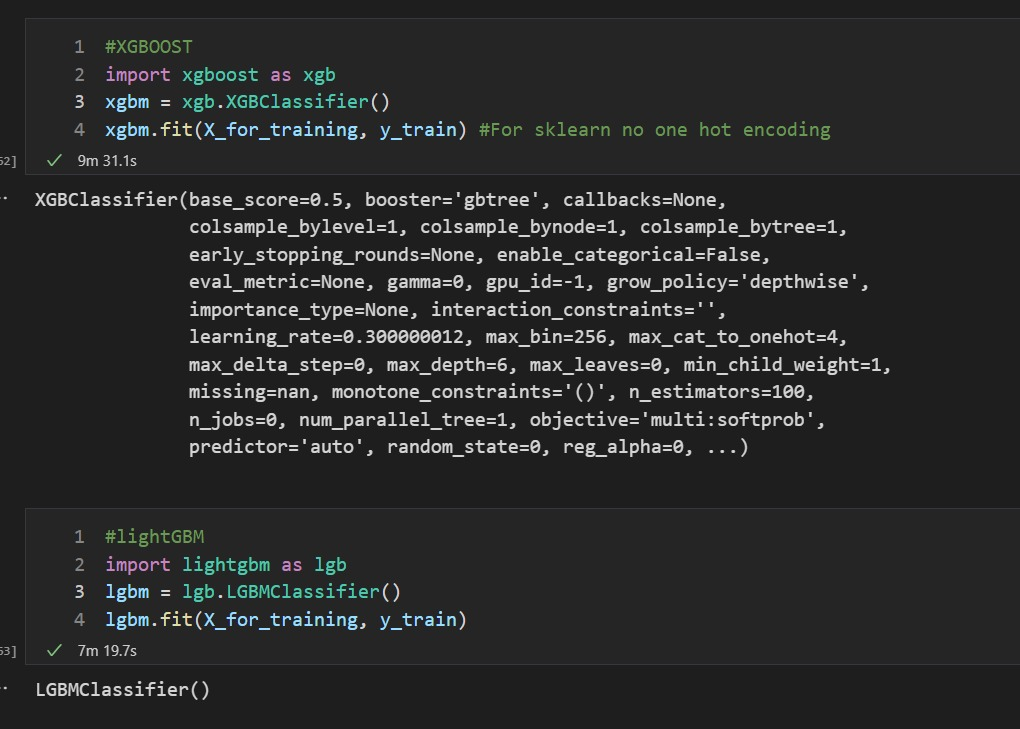


Fig. 15. XGBoost and LightGBM time comparision

1. Conclusion

In this technical paper, we have investigated the best gradient boosting method for image classification which is XGBoost. Level wise tree growth allows for a better prediction of the image compared to Leave wise tree growth. However, there are more improvements that can be made to the models like Hyperparameters Tuning.

##### Acknowledgment

I thank Dr Wilson and Dr Peter, who implemented this technical paper section into the assignment.

##### References

1. Ltd., F., 2022. Floriculture Market is expected to surge to a valuation of around US$ 80.5 Billion by 2029 - Comprehensive Research Report by FMI. [online] GlobeNewswire News Room. Available at: <https://www.globenewswire.com/news-release/2022/01/24/2371911/0/en/Floriculture-Market-is-expected-to-surge-to-a-valuation-of-around-US-80-5-Billion-by-2029-Comprehensive-Research-Report-by-FMI.html> [Accessed 11 May 2022].
2. Hiary, H., Saadeh, H., Saadeh, M. and Yaqub, M. 2018, Flower classification using deep convolutional neural networks. IET Comput. Vis., 12: 855-862. Available at: <https://doi.org/10.1049/iet-cvi.2017.0155> [Accessed 11 May 2022].
3. pranjalk7, 2017. Light GBM vs XGBOOST: Which algorithm takes the crown. [online] Analytics Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/> [Accessed 12 May 2022].
4. Tung, K, 2020, "Flowers Dataset", . Available at: <https://doi.org/10.7910/DVN/1ECTVN, Harvard Dataverse, V8, UNF:6:z6JGwpi2tftxFU+tbVH/3g== [fileUNF]> [Accessed 12 May 2022].
5. WekaDeeplearning4j, 2022. IMAGENET 1000 Class List - WekaDeeplearning4j. [online] Deeplearning.cms.waikato.ac.nz. Available at: <https://deeplearning.cms.waikato.ac.nz/user-guide/class-maps/IMAGENET/> [Accessed 12 May 2022].
6. Rohit Thakur, 2019. Step by step VGG16 implementation in Keras for beginners. [online] Medium. Available at: <https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c> [Accessed 16 May 2022].
7. Martins, D., 2021. XGBoost: A Complete Guide to Fine-Tune and Optimize your Model. [online] Medium. Available at: <https://towardsdatascience.com/xgboost-fine-tune-and-optimize-your-model-23d996fab663> [Accessed 16 May 2022].
8. Kwiatkowski, R., 2021. Gradient Descent Algorithm — a deep dive. [online] Medium. Available at: <https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21> [Accessed 16 May 2022].
9. Saha, S., 2022. XGBoost vs LightGBM: How Are They Different - neptune.ai. [online] neptune.ai. Available at: <https://neptune.ai/blog/xgboost-vs-lightgbm> [Accessed 16 May 2022].