# 

**CZ4041 Machine Learning**

Group 28

Kaggle: Plant Seedlings Classification

Rank 104 out of 833 (**Top 12%**)

| **Name** | **Matriculation Number** |
| --- | --- |
| Jeremy Book Kay Yip | U1822479G |
| Hussain Khozema Kheriwala | U1822071C |
| Wong Ying | U1821686A |
| Lim Yu Jie | U1822070A |

# 

**Table of Contents**

[**Roles & Contribution**](#_1zhtx67pr67f) **1**

[**Best Evaluation Score and Ranked Position**](#_8gt5mpg8hytt) **1**

[**Problem Statement**](#_ua6o1vore6q3) **1**

[**Dataset**](#_kbljpfb37432) **2**

[**Proposed Solution**](#_xhqyg3apmxy) **3**

[Preprocessing](#_oc8ki4hqklfw) 3

[Feature Engineering](#_qprep6xyqfcm) 6

[Methodologies](#_vaprpu7ft614) 7

[**Experiments**](#_z20k3gkdemc9) **12**

[Experiment 1.1: Random Forest Classifier (RFC)](#_zd7gmvccrrz0) 12

[Experiment 1.2: RFC + Hyperparameters Tuning](#_xbkmwlcmjaf4) 13

[Experiment 1.3: RFC + Feature Engineering](#_iw3214voxh2p) 13

[Experiment 2.1: Linear Support Vector Machine](#_bucbil563wzr) 14

[Experiment 2.2: Linear SVM + Feature Engineering](#_jnhqqd41jocs) 14

[Experiment 3.1: K-Nearest Neighbour (KNN)](#_r9tlhh24wmq) 14

[Experiment 3.2: KNN + Feature Engineering](#_v1cmu9pz352b) 15

[Experiment 4.1: Deep Neural Network (DNN)](#_g3j3uyjxry1n) 15

[Experiment 4.2: DNN + Feature Engineering](#_lf1cp65sgm5n) 16

[Experiment 5.1: Convolutional Neural Network (CNN)](#_nqvi7l8nh4bs) 16

[Experiment 5.2: CNN + Feature Engineering](#_t8dksdfeyg60) 16

[Experiment 6: EfficientNet](#_puxrzt5x93om) 17

[Experiment 7: Xception](#_klzjcp8g42ch) 18

[Experiment 8: Inception](#_c3ue2gnagebs) 18

[Experiment 9: Ensemble Model](#_jzj4n89u4d41) 19

[**Results**](#_uaue4xsb6wpe) **20**

[**Solution Novelty**](#_306hk4a7qwlt) **21**

[**Challenges Faced**](#_h6whxhmp15e0) **23**

[**Conclusion**](#_xdrnxxvlgnjz) **25**

[**References**](#_g9820smer67g) **26**

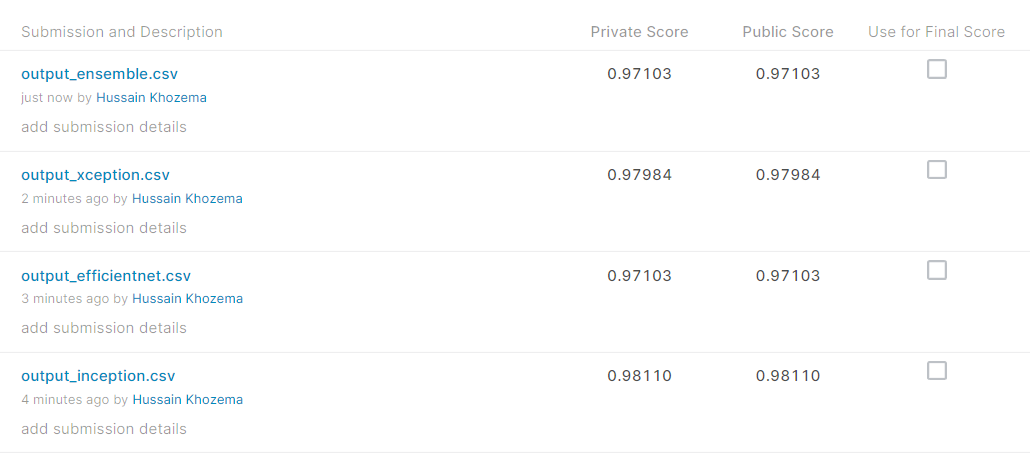
# 

# 

# Roles & Contribution

| **Name** | **Roles** |
| --- | --- |
| Jeremy Book Kay Yip | Transfer learning, Ensemble Model, Video presentation |
| Hussain Khozema Kheriwala |
| Wong Ying | Raw classifier, Raw classifiers with Feature Engineering, Video Presentation |
| Lim Yu Jie |

# Best Evaluation Score and Ranked Position



*Figure 1: Best Evaluation Score*

# Problem Statement

In the field of gardening, weed is a major problem for farmers. Weed competes with the crops for nutrients from the soil, hence the crops will have significantly lower amounts of essential nutrients to grow healthily. As such, weed has to be removed to ensure proper crop growth. However, it is difficult to differentiate between weed and seedlings, as all their leaves are green. Therefore, in this project, we will be exploring the potential of using machine learning to develop an image classification model to determine a plant's species from a photo. This would potentially help in the removal of weeds, which would return higher crop yields, by removing the hiding place for pests and reducing competition for nutrients with the crops.

# 

# Dataset

| **Plant Seedling Dataset** |
| --- |
| * 12 unique species of plants at various growth stages * 4750 training images and 794 test images   **Varying image resolutions**   * Image sizes varies from 154x154 to 1285x1285 pixels   **Coloured images**   * Images taken in different lightning * Have similar backgrounds   **High similarity between classes**   * Leaves of different species looks alike     *Figure 2: Distribution of species in dataset* |

# 

# Proposed Solution

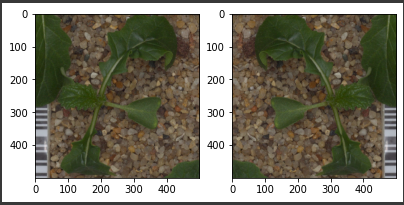
## Preprocessing

**Data Augmentation**

To increase the variation of the dataset, data augmentation has been used to enhance the images. In particular, the operations performed were flip, rotate and zoom. The following shows the specific data augmentations with their respective datagen functions.

Horizontal Flip

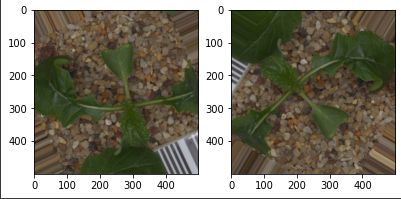
datagen = ImageDataGenerator(horizontal\_flip=True)

****

*Figure 3: Horizontal Flip effect*

Rotation

datagen = ImageDataGenerator(rotation\_range=120)

****

*Figure 4: Rotation effect*

Zoom

datagen = ImageDataGenerator(zoom\_range=[0.3, 1.5])

****

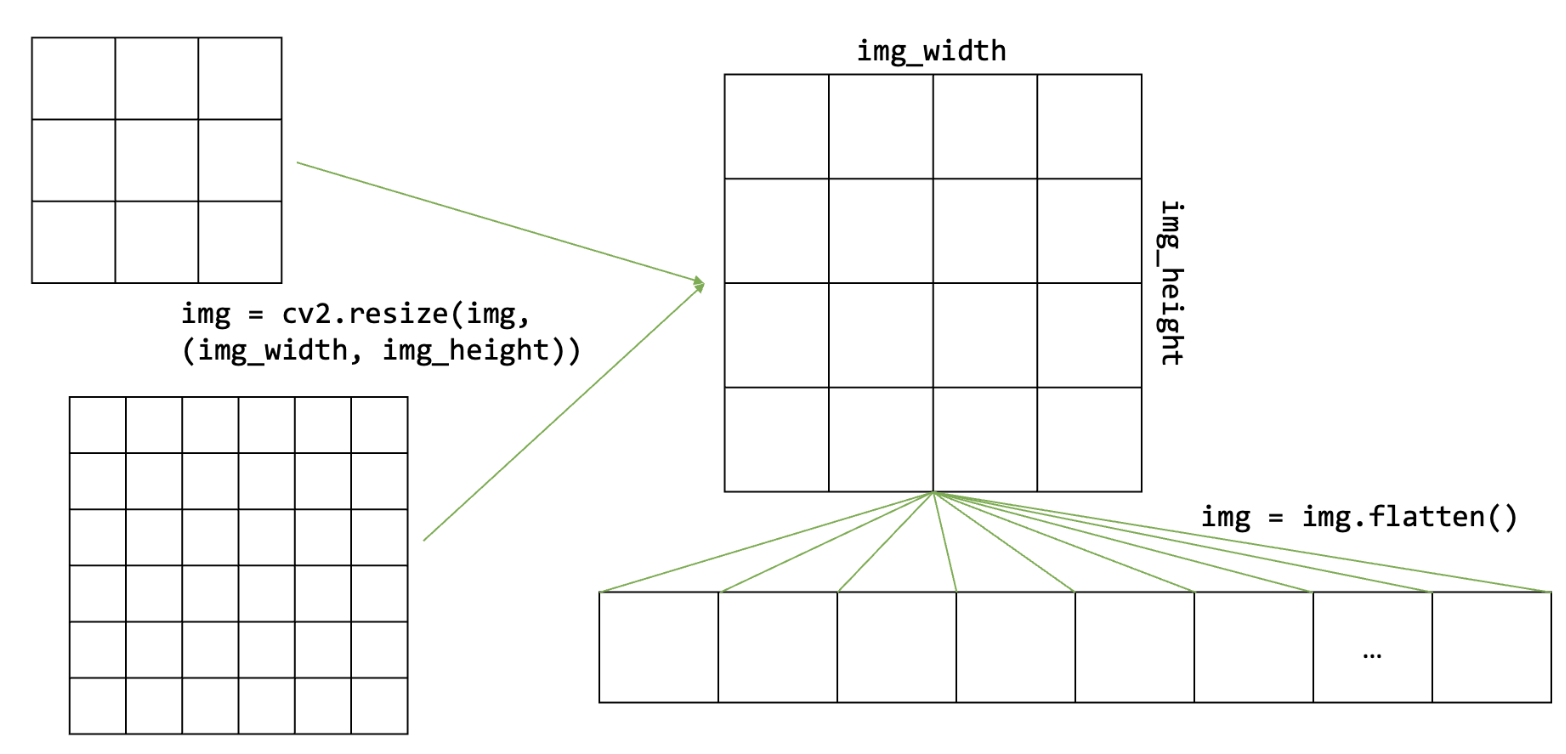
*Figure 5: Zoom effect*

**Resizing and Flattening**

To make all the images consistent before feeding them into the models, all the images have been resized to a fixed size using the function

img = cv2.resize(img, (img\_width, img\_height))

Some models only read 1-dimensional arrays, and in those cases, the images have been flattened with the function img = img.flatten().



*Figure 6: Resizing and Flattening of images*

**Coloured dataset processing**

The original images contain the surrounding of the seedlings, the soil and gravel. These noise factors confuse the model since it is common across all species. In order to aid the model in extracting features of the images for classification without losing features critical towards a good prediction, masking, segmenting and sharpening the images are explored.



*Figure 7: Original image representation of crop seedlings*

Firstly, we converge the original images (first image) to HSV (hue-saturation-value) to create a mask (second image) for the green areas of the images. The mask is then applied to the images to sieve out the areas to be kept (third image) or removed. Lastly, the image is sharpened to help the features stand out even better (fourth image).



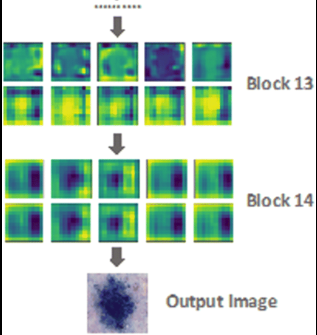
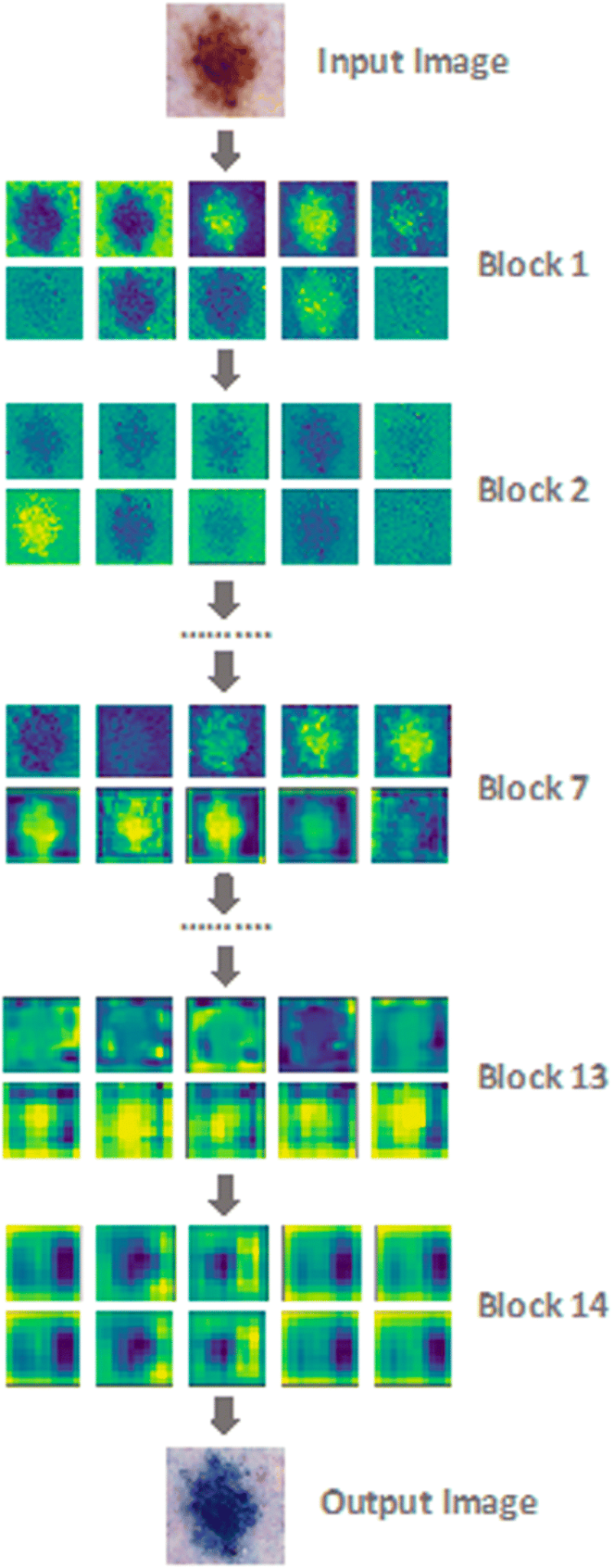
*Figure 8: Pipeline of filters on images*

## 

## Feature Engineering

Feature engineering refers to the process of using domain knowledge to select and transform the most relevant variables from raw data when creating a predictive model using machine learning or statistical modelling. We know from domain knowledge that the leaves of the seedlings are green, while the background soil and gravel is any colour other than green. This means that we can simply extract the green pixels from the image for training. This preprocessing has been previously discussed under coloured dataset preprocessing.

With the processed images, we also used a transfer learning model called Xception to perform feature extraction. Xception has 71 layers of neurons. We can load a pre-trained version of the network trained on more than a million images from the ImageNet database. The pre-trained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The Xception transfer learning model is therefore able to identify the parts of the original image that is a seedling and return the extracted features in numbers for inputs to model training.



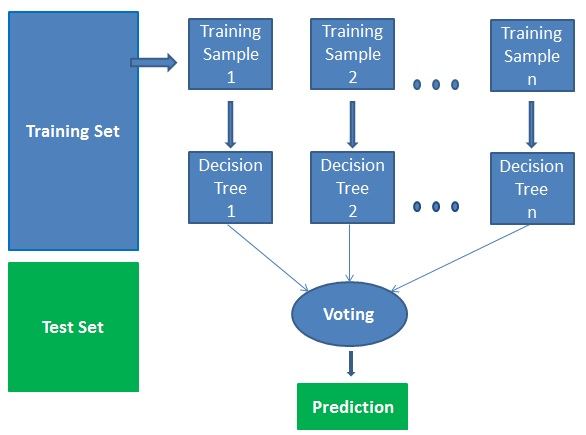
*Figure 9: Example of Feature Extraction Output*

## 

## Methodologies

**Random Forest Classifier (RFC)**

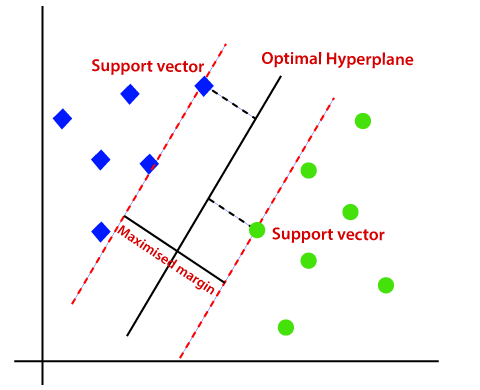
RFC is one of the most flexible and easiest algorithms to use. The "random forest" refers to an ensemble of decision trees, which are then merged based on voting to improve the overall result. It is commonly used for recommendation engines, image classification and feature selection. An advantage of RFC is its resistance to overfitting since it averages the prediction, which cancels out biases.



*Figure 10: Random Forest Classifier*

**Support Vector Machine (SVM)**

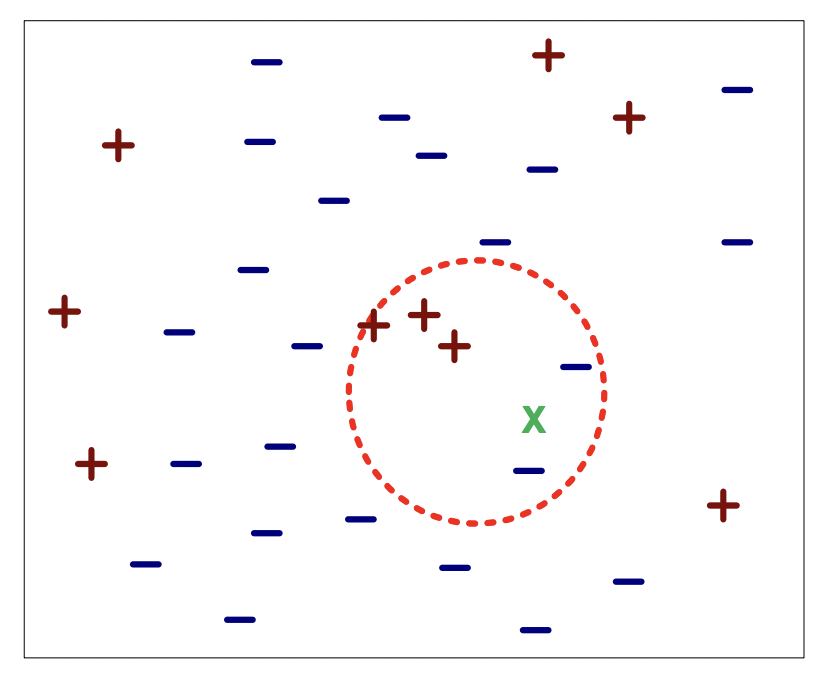
SVM is responsible for finding the decision boundary to separate different classes with maximised margin.Generally, SVM performs and generalises well and has proven itself to be a fast and efficient classifier. One advantage of SVM is its ability to handle high dimensional data.



*Figure 11: Support Vector Machine*

**K-Nearest Neighbour (KNN)**

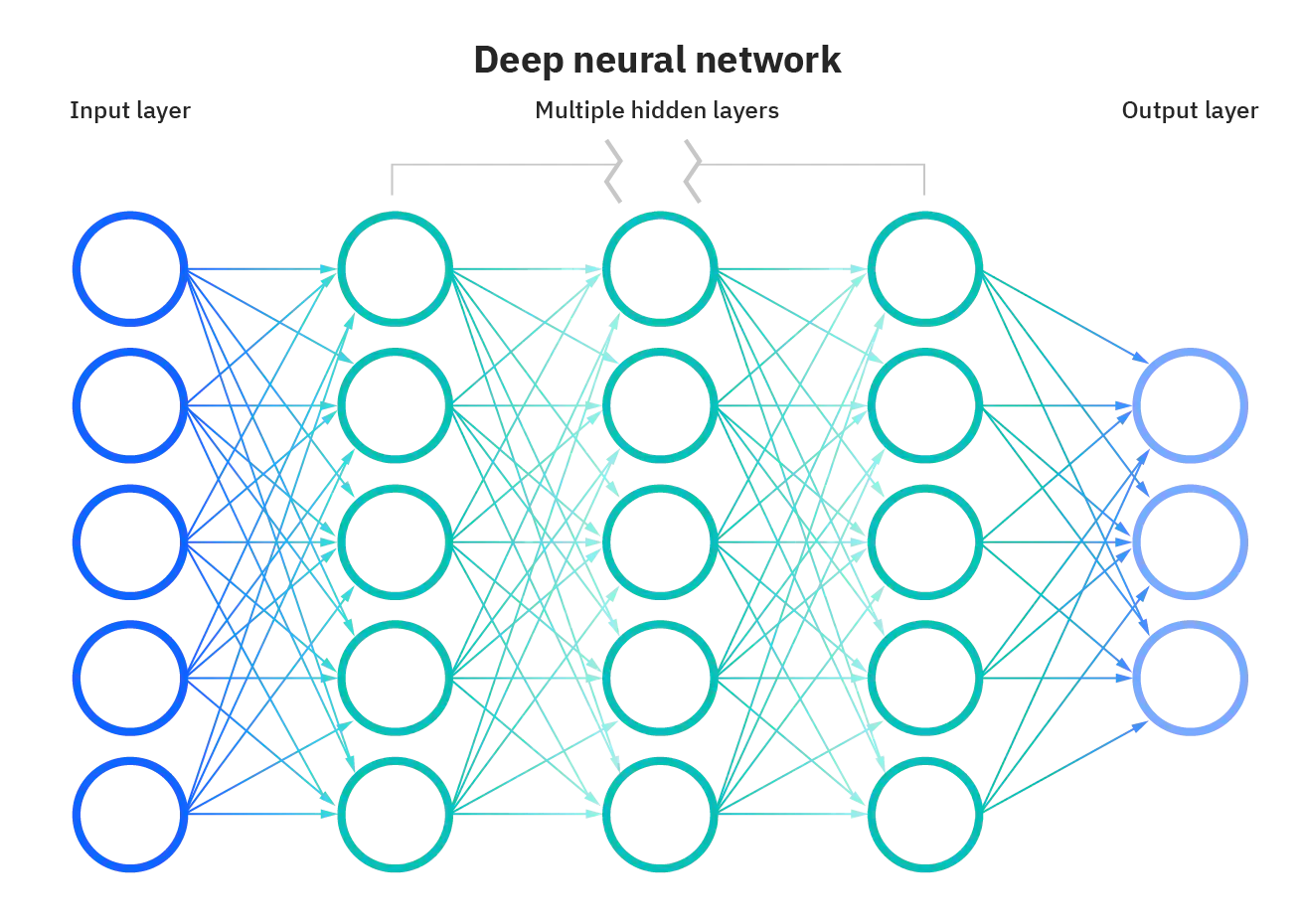
KNN is a supervised learning algorithm which is commonly used for classification. It considers K nearest neighbours to predict the class for the unlabeled data. For the classifier to make a prediction for a new data point, it first considers all the data in the training set. Then, it finds the new data point’s K-nearest neighbours from the feature space using Euclidean distance. The new data point takes the majority vote of class labels among the K-nearest neighbours.



*Figure 12: KNN example*

**Deep Neural Network (DNN)**

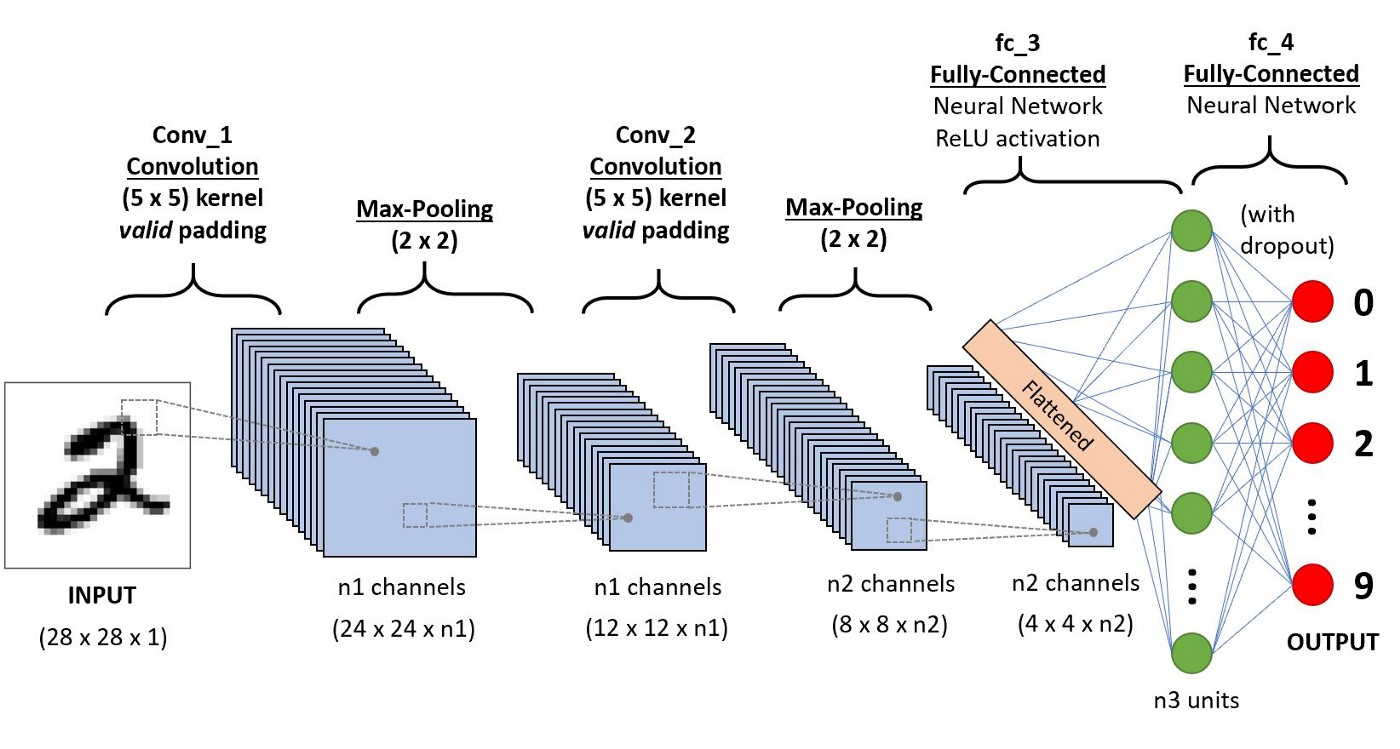
A neural network that consists of more than three layers is considered a deep learning algorithm. It is known for its strong ability to learn the input data’s increasingly abstract representations. By observing patterns in the data, a deep learning model can cluster inputs appropriately. It is used in technologies such as face and voice recognition, and in any field that deals with analysing and categorising sensor data such as still images or video and audio streams.



*Figure 13: DNN architecture*

**Convolutional Neural Network (CNN)**

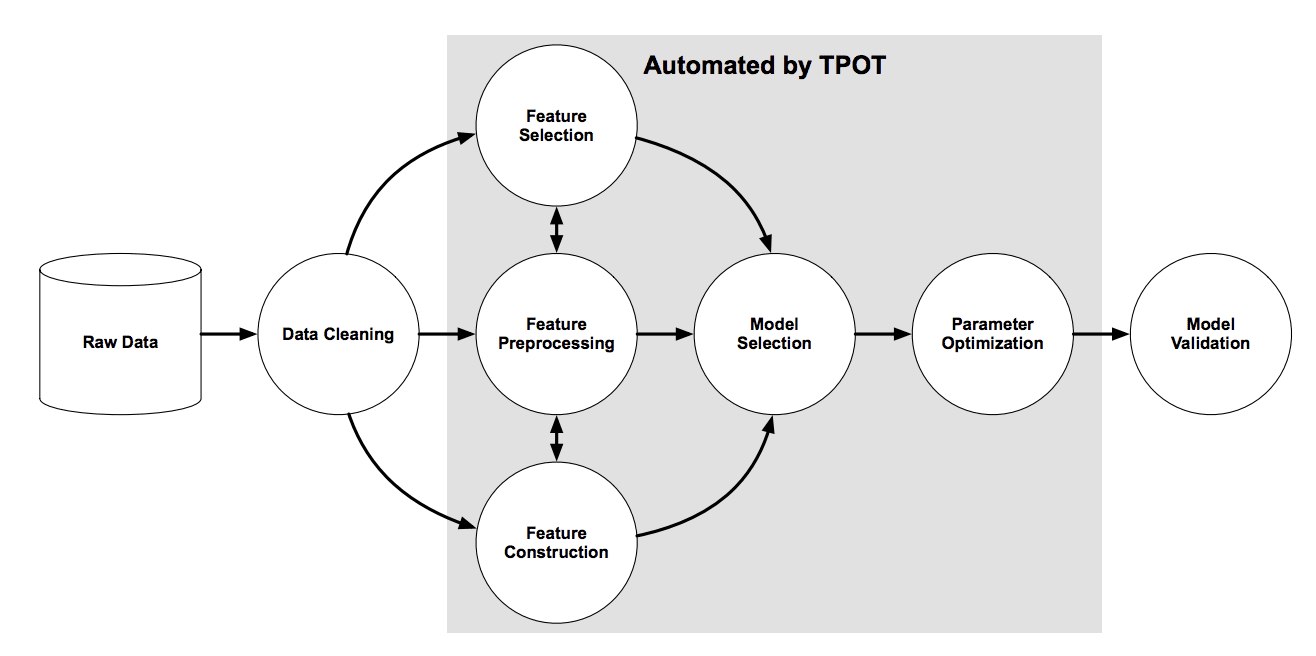
CNN is a class of DNN which is popularly used to analyse visual imagery. It is able to successfully capture spatial dependencies in an image with relevant filters. Another advantage is its ability to reduce the number of parameters involved and reusing of weights, making it a better fitting to the image dataset.

****

*Figure 14: Convolutional Neural Network model*

**Hyperparameters Tuning**

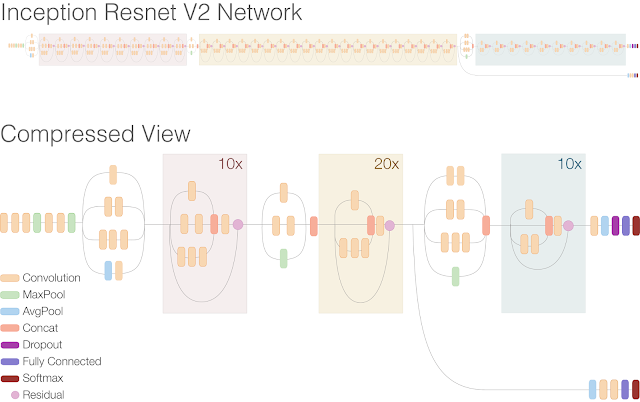
Hyperparameters control the overall behaviour of a machine learning model, hence it is crucial that the optimal combination of hyperparameters is found to help minimise the loss function. Therefore, automated machine learning (AutoML) is explored. AutoML allows for automatic discovery of well-performing models for predictive modelling tasks with very little user involvement. The library used is TPOT, it makes use of the popular Scikit-Learn machine learning library for data transforms and machine learning algorithms and uses a Genetic Programming stochastic global search procedure to efficiently discover a top-performing model pipeline for a given dataset.



*Figure 15: TPOT pipeline*

**Inception-ResNet-v2**

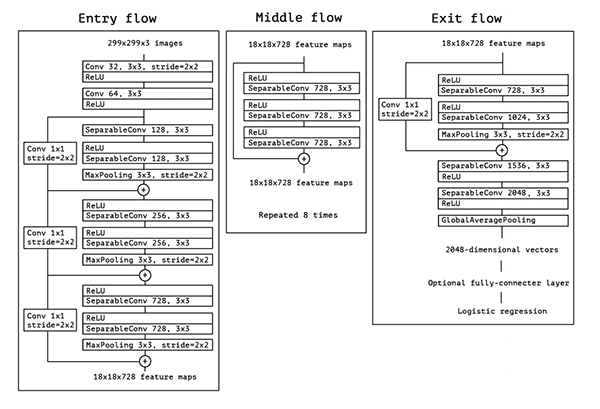
Inception-ResNet-v2 is a variation of the earlier Inception V3 model which borrows some ideas from Microsoft's ResNet papers. It is trained on more than a million images from the ImageNet database. The network is 164 layers deep and can classify images into 1000 object categories. Therefore, the network has learned rich feature representations for a wide range of images.

****

*Figure 16: Schematic diagram of Inception-ResNet-v2*

**Xception**

Xception is an extension of the Inception architecture which replaces the standard Inception modules with depth wise separable convolutions. It is 71 layers deep and trained on more than a million images from the ImageNet database, similar to Inception-ResNet-v2. In this model, the data first goes through the entry flow, then it goes through the middle flow (repeating itself 8 times in this middle flow), and finally through the exit flow.

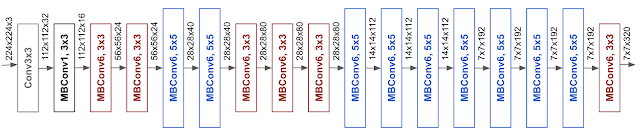
****

*Figure 17: Xception Architecture*

In this project, we have used xception for both feature engineering, as discussed earlier, and model training.

**Efficient Net**

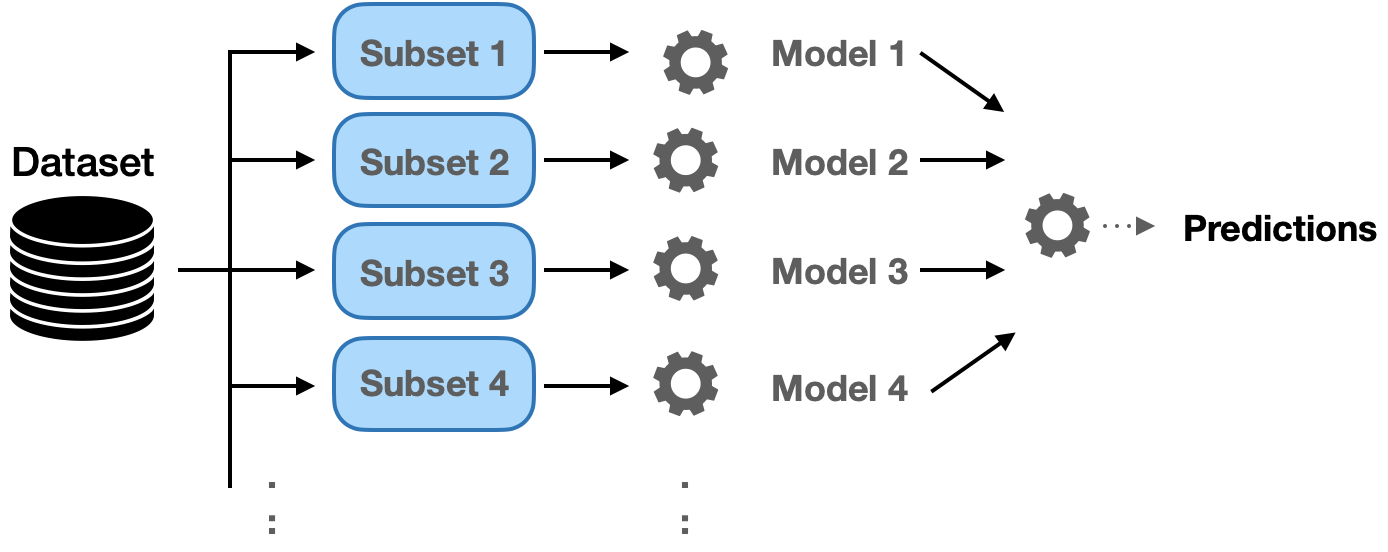
EfficientNet is a CNN and scaling method that uniformly scales all dimensions of depth/width/resolution using a compound coefficient. It is among the most efficient models that reaches State-of-the-Art accuracy on both imagenet and common image classification transfer learning tasks. It relies on AutoML and compound scaling to achieve superior performance without compromising resource efficiency.



*Figure 18: EfficientNet-B0 architecture*

**Ensemble Model**

Ensemble models is a machine learning approach to combine multiple other models, also known as base estimators, in the prediction process. Overall, it is able to overcome the challenges of high variance, low accuracy and features noise and bias when using a single model. Since all models have their own limitations, by aggregating them, the overall accuracy could get boosted. Some ensemble techniques include max voting, weighted average, bagging, boosting, stacking and blending.



*Figure 19: Architecture of Ensemble Model*

# 

# Experiments

We used several models to predict the species of the seedlings:

1. Random Forest Classifier (RFC)
2. Linear Support Vector Machine (SVM)
3. K-Nearest Neighbours (KNN)
4. Deep Neural Network (DNN) (Fully-Connected)
5. Convolutional Neural Network (CNN)
6. Xception
7. Inception
8. Efficient Net
9. Ensemble Model

For raw models (without Xception model for feature extraction), all images were resized to 45 by 45 pixels and preprocessed. The preprocessed output is directly fed into the models for training and testing.

We also tried using the Xception transfer learning model for feature engineering, as described earlier under the “Feature Engineering'' section. All images are resized to 299 by 299 pixels and preprocessed before being fed to the Xception model for feature extraction. The output features are then used as inputs for training models 1 to 5 above. The following experiment results show that using Xception for feature extraction performs better in predicting plant seedling species.

For models / experiments 1 to 5, the train-validation split is as follows:

| **Training Size** | 3800 | **Validation Size** | 950 |
| --- | --- | --- | --- |

For models / experiments 6 to 9, train-validation split is as follows:

| **Training Size** | 4279 | **Validation Size** | 471 |
| --- | --- | --- | --- |

## 

## Experiment 1.1: Random Forest Classifier (RFC)

**Experiment details:**

| **Hyperparamter** | **Value** |
| --- | --- |
| n\_estimators | 500 |

**Results:**

| **Private Val Accuracy** | **0.67368** | **Public Kaggle Score** | **0.67128** |
| --- | --- | --- | --- |

## Experiment 1.2: RFC + Hyperparameters Tuning

**Experiment details:**

TPOT classifier has been explored to help return the best hyperparameters. The hyperparameters tuned include n\_estimator, max\_features, max\_depth, min\_samples\_split, min\_sample\_leaf and criterion.

The result achieved was not significantly better than that of the raw RFC model despite taking several hours to search for the best hyperparameters. Due to the poor performance and time constraint, subsequent models were not further tuned with TPOT.

| **(Best) Hyperparamter** | **Value** |
| --- | --- |
| n\_estimator | 1600 |
| max\_features | sqrt |
| max\_depth | 670 |
| min\_samples\_split | 5 |
| min\_sample\_leaf | 1 |
| criterion | gini |
| generations | 5 |
| population\_size | 24 |
| offspring\_size | 12 |
| verbosity | 2 |
| early\_stop | 12 |
| cv | 4 |

**Results:**

| **Private Val Accuracy** | **0.66736** | **Public Kaggle Score** | **0.69395** |
| --- | --- | --- | --- |

## Experiment 1.3: RFC + Feature Engineering

**Experiment details:**

The Random Forest Classifier now trains on the features selected by the Xception model, with the same hyperparameters as in Experiment 1.1. There was an accuracy improvement of **18.766%** on the test set.

**Results:**

| **Private Val Accuracy** | **0.86210** | **Public Kaggle Score** | **0.85894** |
| --- | --- | --- | --- |

## Experiment 2.1: Linear Support Vector Machine

**Experiment details:**

| **Hyperparamter** | **Value** |
| --- | --- |
| kernel | linear |
| C | 1 |
| decision\_function\_shape | ovo |
| random\_state | seed=4041 |

**Results:**

| **Private Val Accuracy** | **0.57368** | **Public Kaggle Score** | **0.60327** |
| --- | --- | --- | --- |

## 

## Experiment 2.2: Linear SVM + Feature Engineering

**Experiment details:**

The SVM now trains on the features selected by the Xception model, with the same hyperparameters as above (Experiment 2.1). There was an accuracy improvement of **29.093%** on the test set.

**Results:**

| **Private Val Accuracy** | **0.90947** | **Public Kaggle Score** | **0.89420** |
| --- | --- | --- | --- |

## 

## 

## Experiment 3.1: K-Nearest Neighbour (KNN)

**Experiment details:**

| **Hyperparamter** | **Value** |
| --- | --- |
| n\_neighbours | 15 |

**Results:**

| **Private Val Accuracy** | **0.42105** | **Public Kaggle Score** | **0.36964** |
| --- | --- | --- | --- |

## 

## Experiment 3.2: KNN + Feature Engineering

**Experiment details:**

The KNN Classifier now trains on the features selected by the Xception model, with the same hyperparameter as above (Experiment 3.1). There was a significant accuracy improvement of **46.663%** on the test set.

| **Hyperparamter** | **Value** |
| --- | --- |
| n\_neighbours | 15 |

**Results:**

| **Private Val Accuracy** | **0.85473** | **Public Kaggle Score** | **0.83627** |
| --- | --- | --- | --- |

## 

## Experiment 4.1: Deep Neural Network (DNN)

**Experiment details:**

| **Hyperparamter** | **Value** |
| --- | --- |
| optimizer | Nadam |
| learning\_rate | 0.0004 |
| beta\_1 | 0.9 |
| beta\_2 | 0.999 |
| loss | sparse\_categorical\_crossentropy |
| metrics | accuracy |
| epochs | 50 |
| callbacks | EarlyStopping(monitor='val\_loss', patience=5) |

**Results:**

| **Private Val Accuracy** | **0.37368** | **Public Kaggle Score** | **0.34508** |
| --- | --- | --- | --- |

## 

## Experiment 4.2: DNN + Feature Engineering

**Experiment details:**

The Deep Neural Network Classifier now trains on the features selected by the Xception model, with the same hyperparameters as above (Experiment 4.1). DNN was the worst performing model with a low accuracy score of 0.34508. However, with feature extraction using the Xception model, DNN is now one of the best performing models with an improvement of **55.668%**.

**Results:**

| **Private Val Accuracy** | **0.90842** | **Public Kaggle Score** | **0.90176** |
| --- | --- | --- | --- |

## 

## Experiment 5.1: Convolutional Neural Network (CNN)

**Experiment details:**

| **Hyperparamter** | **Value** |
| --- | --- |
| hidden\_layer\_sizes | (300, 200,) |
| random\_state | seed=4041 |
| max\_iter | 1000 |

**Results:**

| **Private Val Accuracy** | **0.60210** | **Public Kaggle Score** | **0.62090** |
| --- | --- | --- | --- |

## 

## Experiment 5.2: CNN + Feature Engineering

**Experiment details:**

The CNN Classifier now trains on the features selected by the Xception model, with the same hyperparameter as above (Experiment 5.1). There was an accuracy improvement of **28.338%** on the test set.

**Results:**

| **Private Val Accuracy** | **0.90947** | **Public Kaggle Score** | **0.90428** |
| --- | --- | --- | --- |

## 

## Experiment 6: EfficientNet

**Experiment details:**

Before training, all images were resized to 300 by 300 pixels and followed by data augmentation.

| **Hyperparamter** | **Value** |
| --- | --- |
| number of layers after base model | 2 |
| number of hidden layer units | 1024 |
| hidden layer activation function | relu |
| number of output layer units | 12 |
| output layer activation function | softmax |
| regularization method | dropout(0.5) |
| optimizer | Adam |
| learning\_rate | 0.0005 |
| loss | CategoricalCrossentropy |
| metrics | accuracy |
| epochs | 100 |
| callbacks | EarlyStopping  ReduceLROnPlateau  ModelCheckPoint |

**Results:**

| **Private Val Accuracy** | **0.97474** | **Public Kaggle Score** | **0.97103** |
| --- | --- | --- | --- |

**Training Accuracy and Loss:**

|  |  |
| --- | --- |

## Experiment 7: Xception

**Experiment details:**

The pre-train model has been swapped to Xception from the previous experiment, with the same augmentation and hyperparameter as above (Experiment 6). The score improved slightly by 0.00881% on the test set.

**Results:**

| **Private Val Accuracy** | **0.97198** | **Public Kaggle Score** | **0.97984** |
| --- | --- | --- | --- |

**Training Accuracy and Loss:**

|  |  |
| --- | --- |

## Experiment 8: Inception

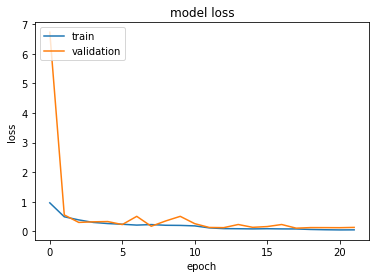
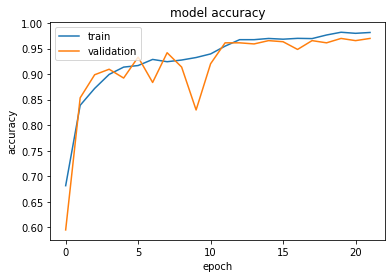
**Experiment details:**

The pre-train model has been swapped to Inception from the previous experiment, with the same augmentation and hyperparameter as above (Experiment 7). The score improved slightly by 0.00126% on the test set.

**Results:**

| **Private Val Accuracy** | **0.96983** | **Public Kaggle Score** | **0.98110** |
| --- | --- | --- | --- |

**Training Accuracy and Loss:**

****

## Experiment 9: Ensemble Model

**Experiment details:**

Ensemble model has used the previous 3 models from Experiments 6, 7 and 8 as the base classifiers.

| **Hyperparamter** | **Value** |
| --- | --- |
| Base model | EfficientNet  Xception  Inception |
| number of layers after base model | 2 |
| number of hidden layer units | 10 |
| hidden layer activation function | relu |
| number of output layer units | 12 |
| output layer activation function | softmax |
| optimizer | Adam |
| learning\_rate | 0.001 |
| loss | CategoricalCrossentropy |
| metrics | accuracy |
| epochs | 100 |
| callbacks | EarlyStopping  ReduceLROnPlateau  ModelCheckPoint |

**Results:**

| **Private Val Accuracy** | **0.99158** | **Public Kaggle Score** | **0.97103** |
| --- | --- | --- | --- |

# 

# Results

| **Model** | **Test Accuracy** | **Ranking** | **Percentile** |
| --- | --- | --- | --- |
| RandomForestClassifier | 0.67128 | 746 | 89.56% |
| RandomForestClassifier + TPOT | 0.69395 | 737 | 88.48% |
| RandomForestClassifier + Xception | 0.85894 | 627 | 75.27% |
| Linear SVM | 0.60327 | 757 | 90.88% |
| Linear SVM + Xception | 0.89420 | 584 | 70.11% |
| KNearestNeighbour | 0.36964 | 781 | 93.76% |
| KNearestNeighbour + Xception | 0.83627 | 666 | 79.95% |
| DeepNN | 0.34508 | 783 | 94.00% |
| DeepNN + Xception | 0.90176 | 568 | 68.19% |
| ConvolutionalNN | 0.62090 | 754 | 90.52% |
| ConvolutionalNN + Xception | 0.90428 | 565 | 67.83% |
| Xception | 0.97984 | 125 | 15.00% |
| EfficientNet | 0.97103 | 262 | 31.45% |
| **Inception** | **0.98110** | **104** | **12.48%** |
| Ensemble (Models 6 to 8) | 0.97103 | 125 | 15.00% |

# 

# Solution Novelty

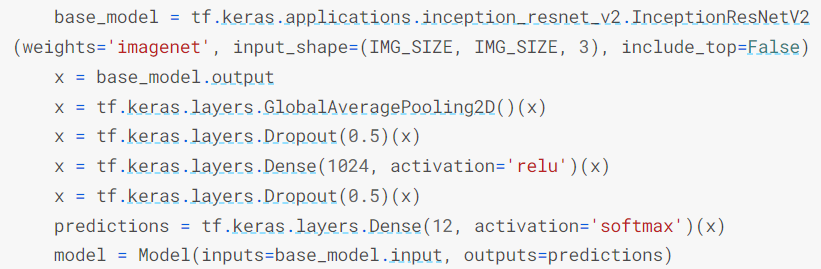
Several novel techniques have been implemented with the attempt to improve the classification accuracy. The Inception model performance has improved to 98.1% due to the combination of the novel techniques which include:

1. **Transfer learning**

With the use of popular pre-trained models namely Xception, InceptionV2, and EffiicientNetB2 which have already been trained on large benchmark dataset, we were able to start from patterns that have been learned to solve our problem. We have unfreezed a few of the top layers of a frozen model base and jointly trained both the newly-added classifier layers and the last layers of the base model. This allowed us to "fine-tune" the higher-order feature representations in the base model in order to make them more relevant for the plant seedling classification.

1. **Dropout**

Dropout layers have been integrated into the model architecture to help reduce overfitting and improve the generalisation of models.



*Figure 20: Dropout Layers of Inception Model*

1. **Keras CallBack (ReduceLROnPlateau and EarlyStopping)**

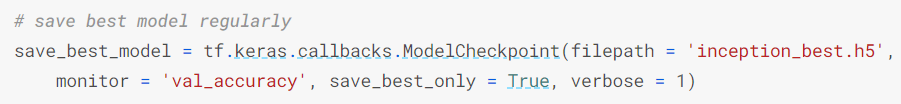
Keras library provides functions to perform during training. During training, two CallBack functions were used namely ReduceLROnPlateau and EarlyStopping. ReduceLRonPlateau reduces the learning rate by a defined factor when the validation accuracy stops decreasing. On the other hand, EarlyStopping will terminate the training when the validation accuracy stops decreasing, which prevents overfitting of the model.



*Figure 21: CallBack function of Inception Model*

1. **Saving best model**

Rather than using the model at the last epoch, the model with the best validation accuracy was saved by using the Keras CallBack function library.



*Figure 22: Saving best model of Inception Model*

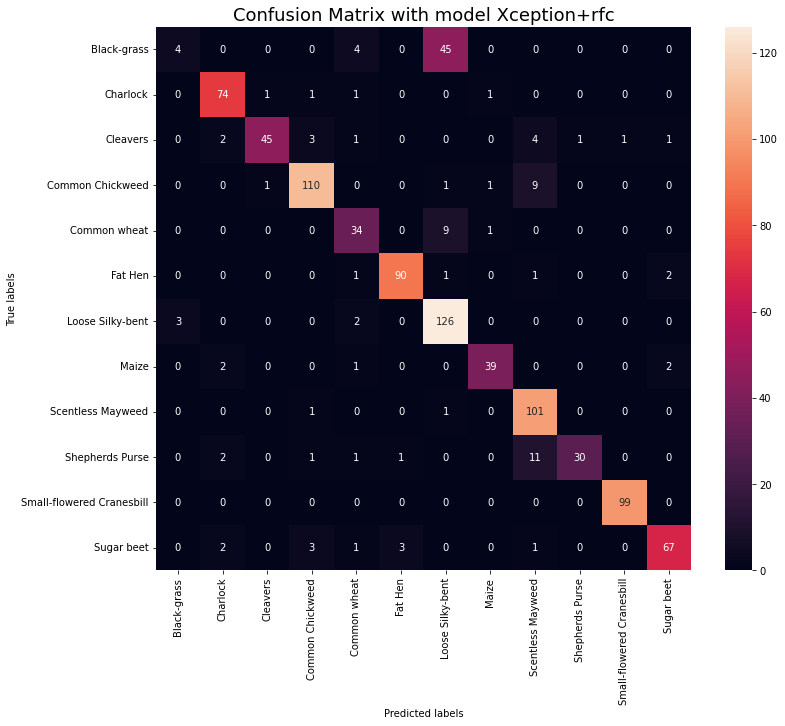
1. **Test data augmentation**

Test-time augmentation has helped to create multiple augmented copies of each image in the test set, having the model make a prediction for each, then returning an ensemble of those predictions. This gives the model the best opportunity for correctly classifying a given image, which in return increases the accuracy.

# 

# Challenges Faced

1. **Misclassification for specific classes**

****

*Figure 23: Confusion Matrix of model RFC with Feature Engineering*

It can be observed from the confusion matrix that 2 particular species are commonly misclassified, which are black-grass and loose silky-bent. Hence, this could be one of the main hurdles to achieving a higher accuracy. We attempted to reduce this false negative error of incorrectly predicting loose silky-bent. However, the prediction of other species were negatively affected and outweighed the gain in predicting Black-grass more accurately. Further work has to be done to consider how to improve the true positives for Black-grass without affecting the prediction of other species.

1. **Time and Resources Constraint**

With a large dataset and heavy preprocessing, the time taken to run and train some models takes up several hours and even days, even with GPU processing. This heavily restricted us on the number of experiments we could conduct within the short period of this project.

1. **Limited memory/Hardware constraints**

In general, machine learning is a mathematical and probabilistic model which requires tons of computations. This is especially so when we used the ensemble model to combine 3 transfer learning models in the prediction process. Additionally, images are represented as 2-dimensional arrays, which require O(n2) space. We were unable to increase the resolution of the images for processing as we will exceed the memory limit, although larger resolutions potentially give better prediction accuracies as they contain more data. We attempted to increase the virtual memory size by allocating space from the hard drive (SSD). However, only the CPU can access the hard drive memory while the GPU is unable to, thus compromising on processing speed, resulting in constraint 2.

1. **Fine-tuning of hyperparameters**

Despite the attempt of using TPOT to help finetune the RFC parameters, the result achieved did not reflect any improvement. Overall, finetuning of models requires trial and error, which is time-consuming. Overturning of the parameters could also result in overfitting of the model. Other forms of genetic algorithms could be explored in an attempt to find the optimal hyperparameters.

1. **Unexpected Performance of Ensemble Model**

Ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. However, it is not true that it will always perform better and there are many potential reasons for this.

* We used a concatenation ensemble for our experiment, which receives different inputs, whatever their dimensions are, and concatenates them on a given axis. A drawback, of attaching information side by side, can be dimensionality explosion. More the networks in the ensemble or greatest are the dense dimensionalities and bigger is the output of the concatenation. This operation can be dispersive, not allowing the final part of the network to learn important information or resulting in overfitting and test accuracy reduction.
* The stand-alone Inception model has an accuracy of 98.11%. It is much harder to achieve small increments in the test accuracy as most factors have already been considered. Including any more factors could only add more noise which worsens the model rather than improve it. Combining additional models which perform worse than Inception may therefore contribute to the lower overall performance of the ensemble model than the stand-alone Inception model.
* The way which the train dataset is split into the train set and validation set could be another factor. There may be distinct differences in the samples of the train set and the validation set which leads to substantial bias, resulting in the poor accuracy of the ensemble model.

# 

# Conclusion

Throughout this project, the team had the opportunity to see the great work done by many other teams on Kaggle. This has given us the chance to use Kaggle as a platform to learn and grow our machine learning skills. On top of that, we were able to apply knowledge we have learnt over the course to help us improve our results for the experiments done. There have been many hurdles to overcome and lessons learnt along the way.

**Complete Machine Learning Workflow**

The team has gained insight on the entire machine learning workflow, from data pre-processing, building datasets, model training, refinement to evaluation.

**Explored Various Models**

We explored an array of machine learning models, ranging from basic classifiers which require only a few lines of code, to an ensemble model which requires many blocks of code. It was an eye opener to know how machine learning has grown and developed over the years.

**Importance of Feature Engineering**

With the use of feature engineering, the team was able to realise significant accuracy improvements in certain models, especially for Deep Neural Network which saw an improvement of 55.668%. From this, our team learned about the great benefits of feature engineering. Effective feature engineering implies higher efficiency of the model and easier algorithms that fit the data. It also makes it easier for algorithms to detect patterns in the data. When feature engineering activities are done correctly, the resulting dataset is optimal and contains all of the important factors that affect our target problem.

**Soft Skills**

We have practised soft skills which include project management, documentations and teamwork. As a team, we respected one another’s opinion and often provided updates to ensure a steady progression of the team.

Overall, the team is glad to have explored various machine learning models and put the theories taught within our syllabus into good use. There was a sense of satisfaction within us as we saw improvement from one experiment to another from the suggestions we gave each other.

# References

1. <https://www.mathworks.com/help/deeplearning/ref/xception.html>
2. <https://www.datacamp.com/community/tutorials/random-forests-classifier-python>
3. <https://builtin.com/data-science/random-forest-algorithm>
4. <https://machinelearningmastery.com/tpot-for-automated-machine-learning-in-python/>
5. <https://www.techtarget.com/searchenterpriseai/definition/automated-machine-learning-AutoML>
6. <https://iq.opengenus.org/xception-model/>
7. <https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html>
8. <https://paperswithcode.com/method/efficientnet>
9. <https://www.analyticsvidhya.com/blog/2021/10/a-beginners-guide-to-feature-engineering-everything-you-need-to-know/>
10. <https://towardsdatascience.com/what-is-feature-engineering-importance-tools-and-techniques-for-machine-learning-2080b0269f10>