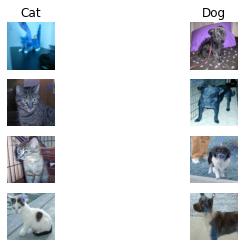
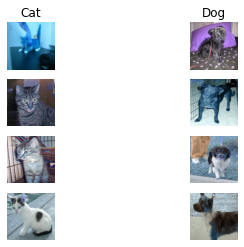
The objective is to build a classifier that tells us whether an image contains a dog or a cat. This is an example of a binary classification problem. For this, the dataset “Cat Dog images.zip” that was taken from Kaggle. The training folder contains 1002 images of cats and dogs. The filenames contained the label “Cat” or “Dog” followed by a serial number. The test folder contains 100 images of cats and dogs without any labelling. This means that we must build a model from scratch and perform Validation methods thoroughly.

**Part A: Pre-processing Phase:**

As part of pre-processing we will read the images, re-size them to 350\*350\*3 and normalize them at last. Below are the original training images.



We created the labels for the class of image from the filenames in the training dataset. '0' stands for cat and '1' stands for dog. However, there are no labels in the testing data and arbitrary labels (0) were assigned to it. These values would change once the model building is done. Below are the training images after normalization.



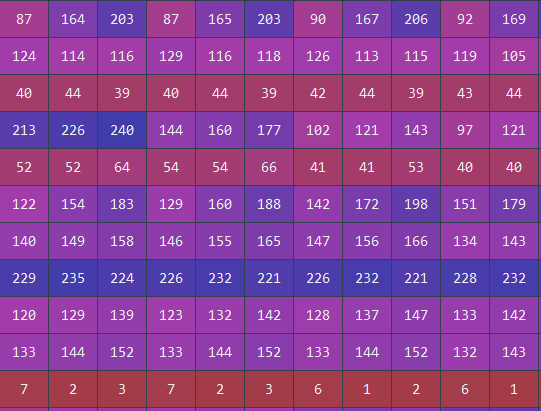
A standard normalizer was used for transforming the images. The formula is as follows:

Z norm= (Z – Z min)/(Z max – Z min) where Z is the variable value, Z min is the minimum value of Z and Z max is the maximum value of Z.

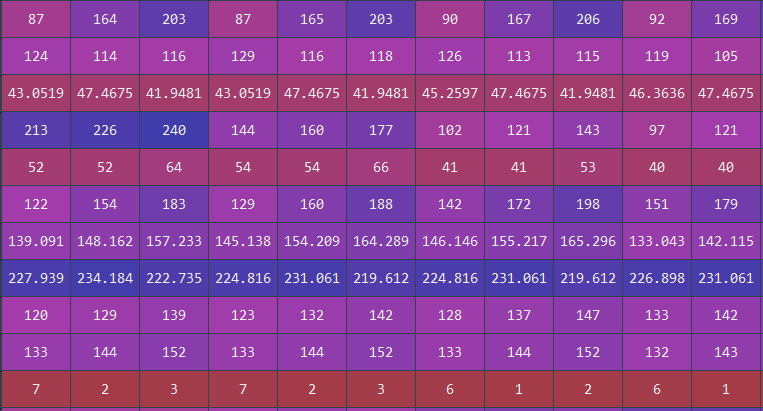
There is not much difference after normalization due to the high contrast in the images. Below are the testing images before and after normalization. The testing image seems to be normalized as it is and hence, we don’t see much difference.



Below is the numeric representation of the training image before normalization.

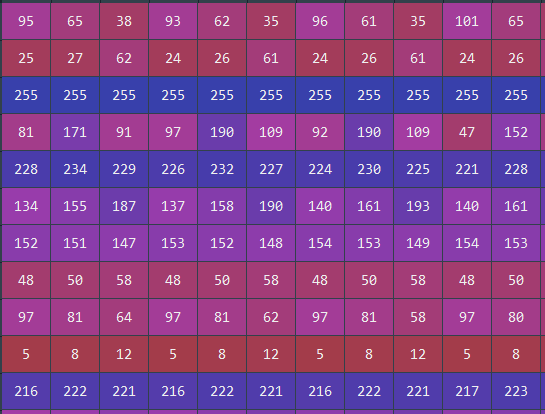


Below is the numeric representation of the training image after normalization.

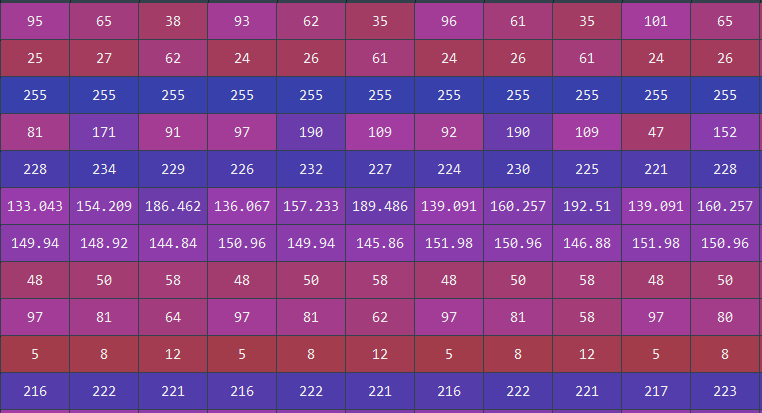


There are slight nuances in the training images pre and post normalization.

Below is the numeric representation of the testing image before normalization.



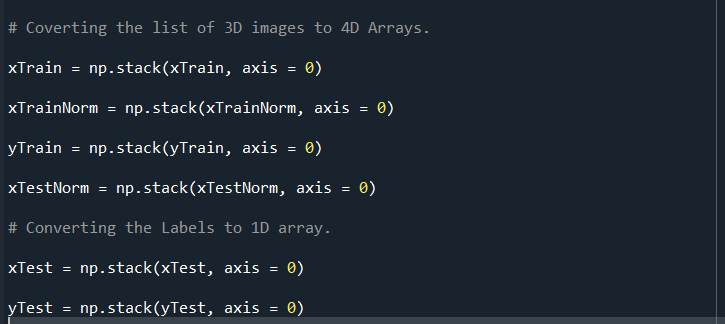
Below is the numeric representation of the testing image after normalization.

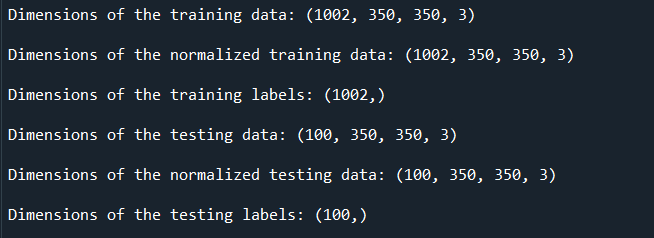


There are slight nuances in the testing images pre and post normalization.

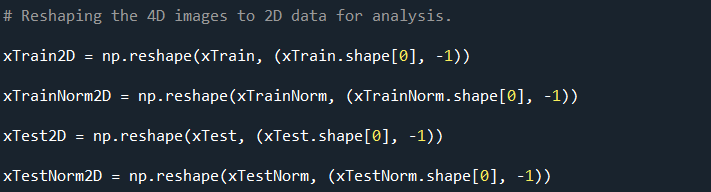
The original and normalized images are stored in separate lists.

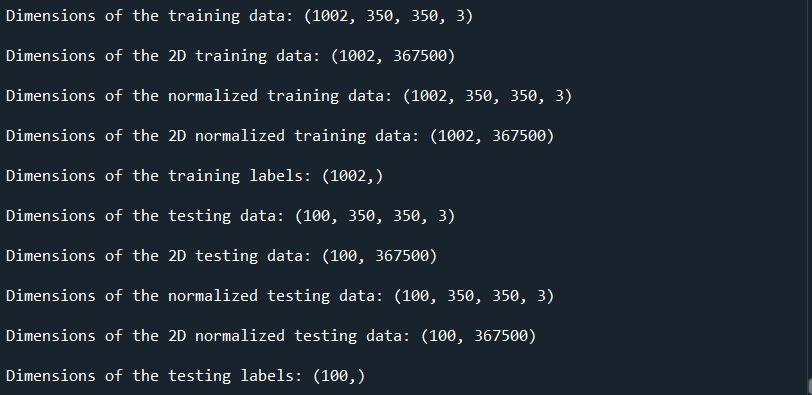
We need to convert the list of 3D images to 4D arrays.





There are 1002 training images and each of them are of the dimensions 350 x 350 with 3 pixels for RGB. Hence, the training images (pre and post normalization) are of the size 1002 x 350 x 350 x 3. The same applies for the testing images as well. Model building cannot be done for 4D data and hence, we must transform the images to 2D.





There are 1002 training images with each image dimension = 350 x 350 x 3. Hence, the training images (pre and post normalization) are of the size 1002 x 367500 which is the product of the 3 dimensions (350\*350\*3).

The same applies for the testing images as well.

Pre-processing steps such as loading, displaying, resizing, normalization is completed.

Next step is dimension reduction. Principal component analysis (PCA) is one of the dimension reduction techniques. It is derived from the eigenvalues and eigenvectors of correlation matrix of a dataset. The eigenvector Matrix is multiplied with the input data to get the principal components. The concept of “Single Vector Decomposition (SVD)” of matrices is applied here.

Before that, we need to check if PCA is needed or not by using Bartlett's test of sphericity.

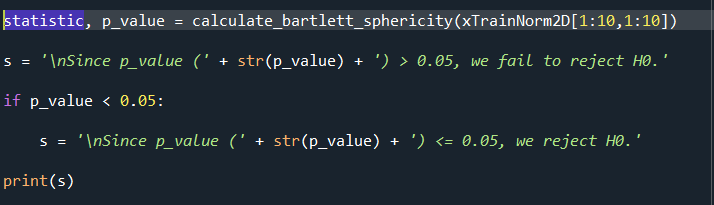
**Bartlett's test of sphericity:**

H0 = Correlation Matrix is an identity matrix which means variables are uncorrelated.

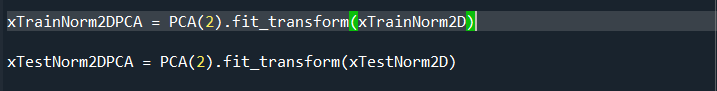
HA = Correlation Matrix is not an identity matrix which means variables are correlated and suitable for factor analysis.

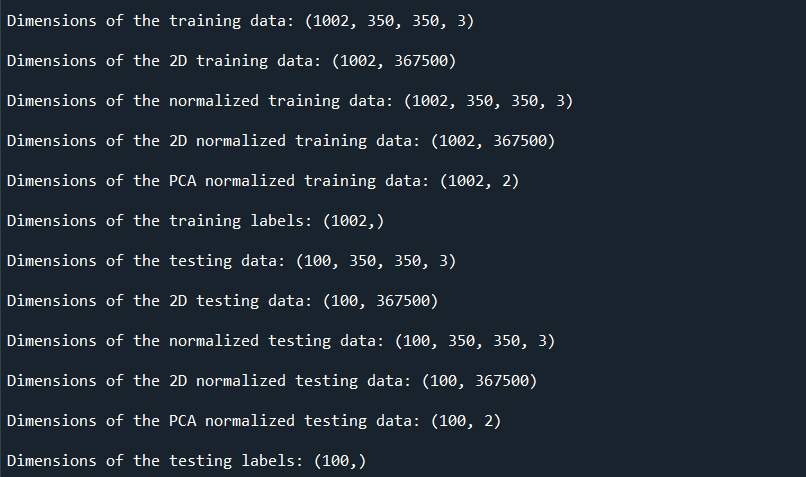
PCA is a special case of factor analysis.

Since there are too many observations in the normalized training data, we take the first ten rows and columns for the Bartlett's test.



Since p\_value (2.1035549803098057e-47) <= 0.05, we reject H0. This means that there is no sufficient evidence to show that the Correlation Matrix is an identity matrix. In PCA we take the first two components since they hold the highest variance.

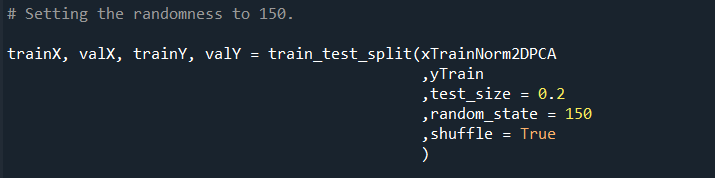




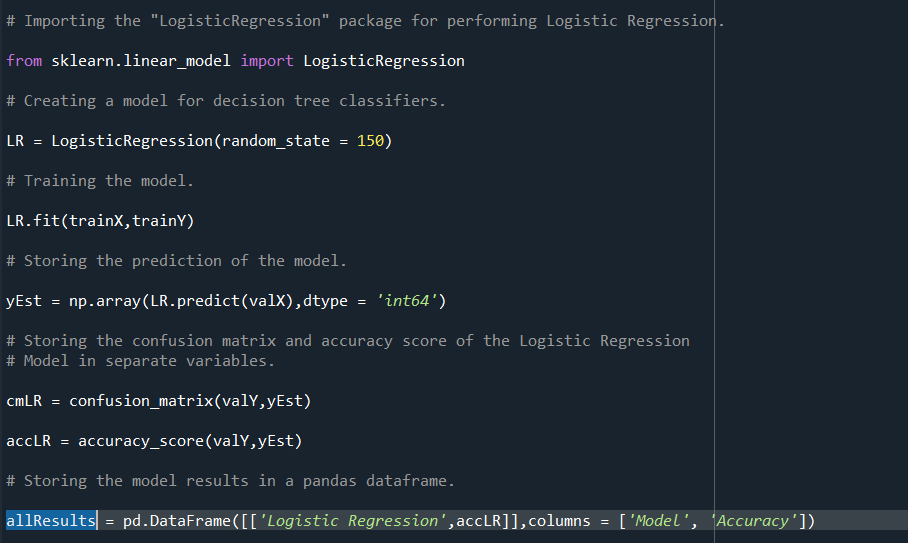
This concludes the pre-processing with the PCA. Next, we go for model building.

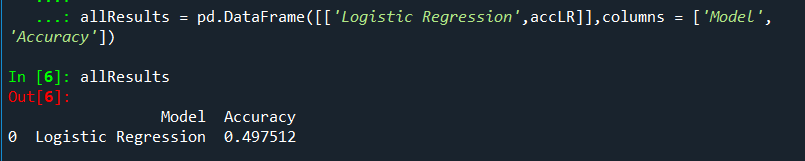
**Part B: Training phase:**

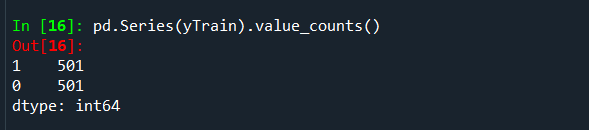
Since this is a binary classification problem, we could use algorithms such as Logistic Regression (LR), Decision Tree Classifier (DTC), Random Forests (RF) and Stochastic Gradient Boosting Models (GBM). Since the test dataset does not contain any prediction values, we need to split the training dataset into train and validation datasets. Split of data -> 80% training data means 0.8\*number of rows in new dataset.



Since all the values are numeric except for the target column, we can apply logistic regression models and check the accuracy.

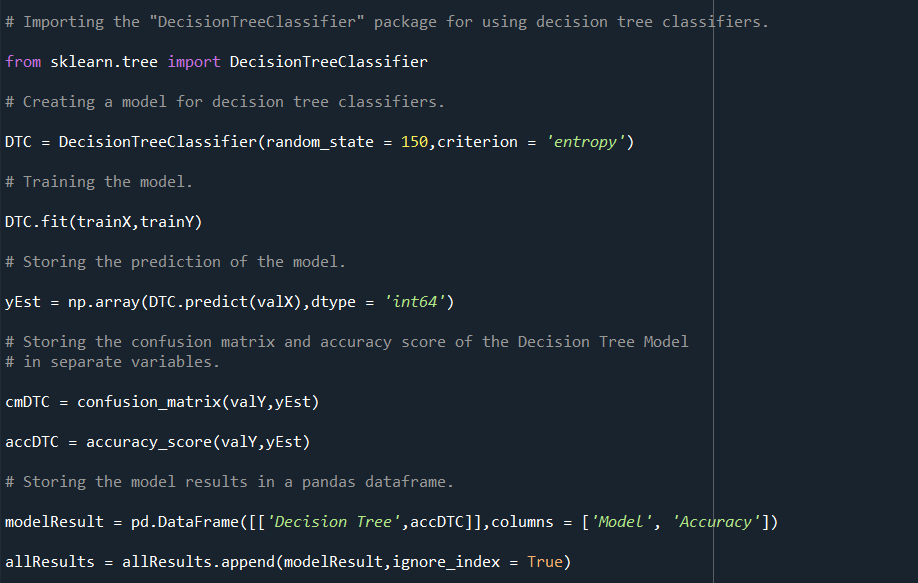


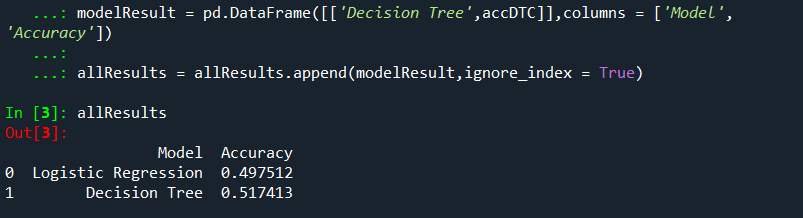




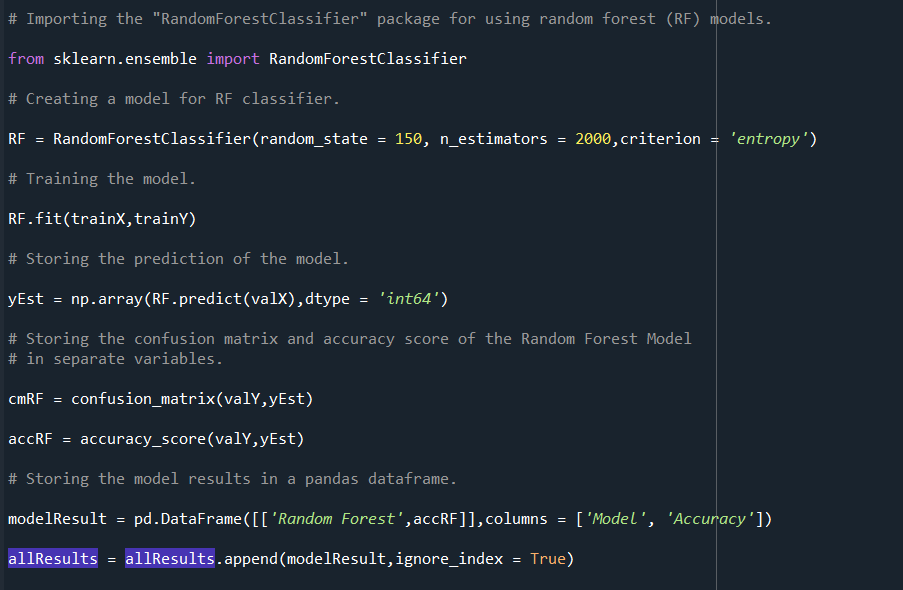
The baseline accuracy is 50%. Our model's accuracy must be higher than this. Logistic Regression has an accuracy of 49.75% which is slightly lesser than the baseline accuracy.

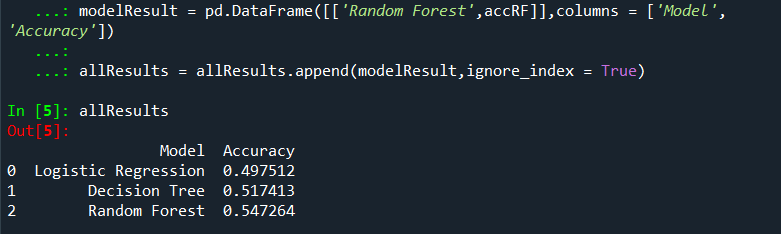
Next, we check the accuracy of a Decision Tree classifier whose splitting criterion is the entropy.





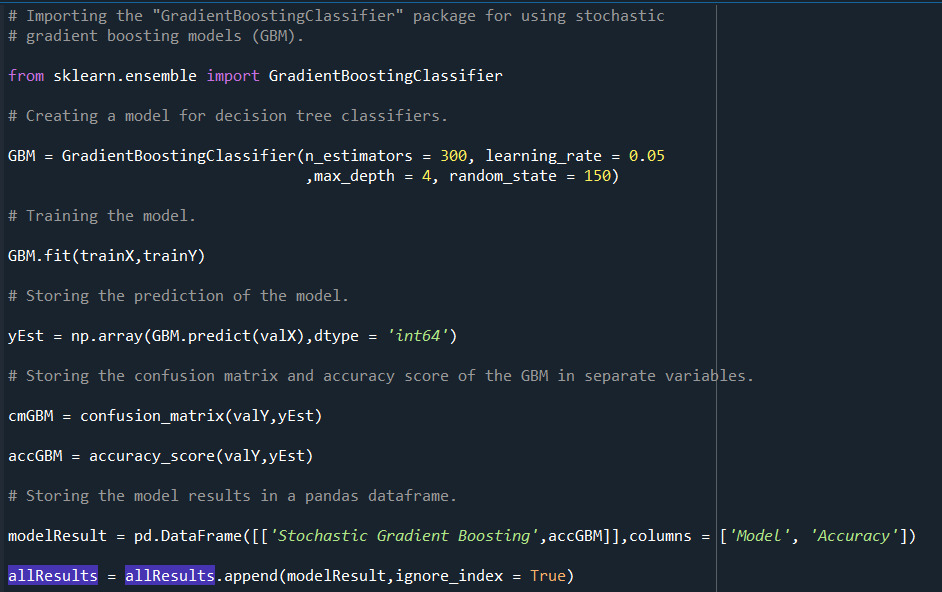
Decision Tree Classifier has an accuracy of 51.74% which is slightly higher than the baseline accuracy. Let’s extend the Decision Tree Classifier to a Random Forest Model with 2000 estimators and the same splitting criterion.

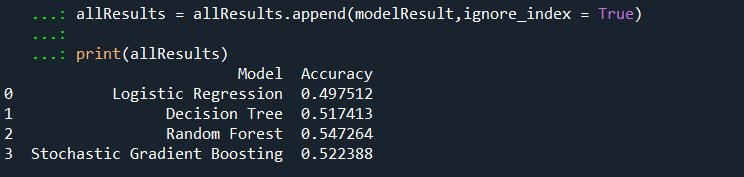




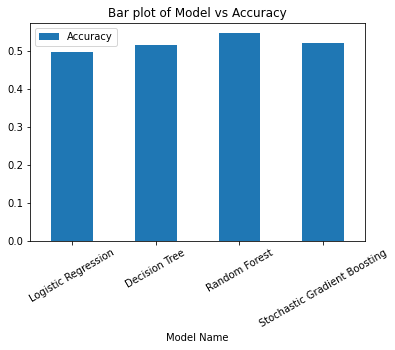
Random Forest Model has an accuracy of 54.72% which is slightly higher than the Decision Tree Classifier but much better than the baseline accuracy.

Lastly, let’s compare the accuracy of a Stochastic Gradient Boosting model (GBM) with the other models. GBM is build using 300 estimators, a learning rate of 0.05 and a maximum depth of 4. Lower the learning rate, higher the accuracy.

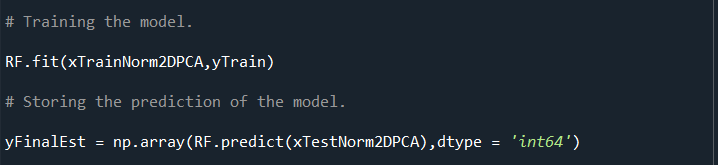


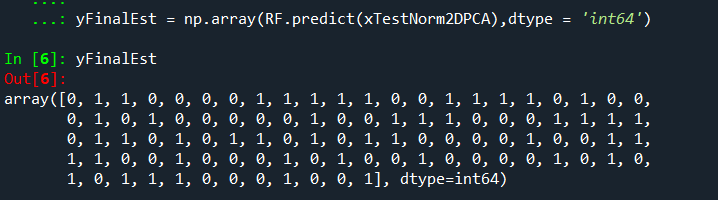


GBM has an accuracy of 52.238% which is slightly higher than the Decision Tree Classifier but lesser than the RF’s accuracy.



Random Forest Model has the best accuracy out of all the models. Hence, it will be used for predicting the testing data.



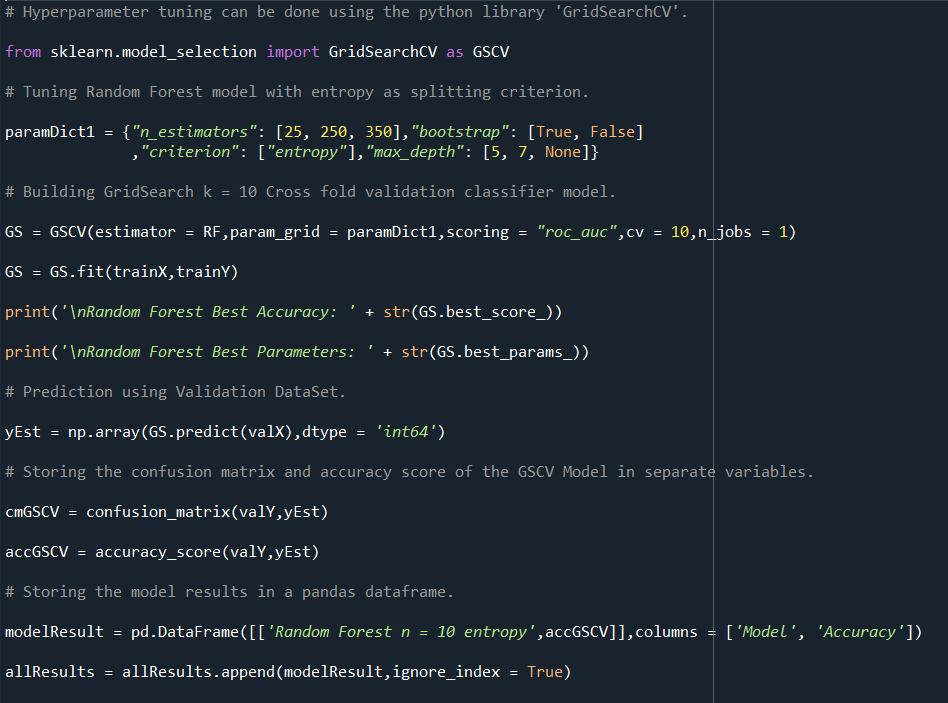


‘0’ stands for cat and ‘1’ stands for dog. Once the prediction is down, we go for hyperparameter tuning of the best model.

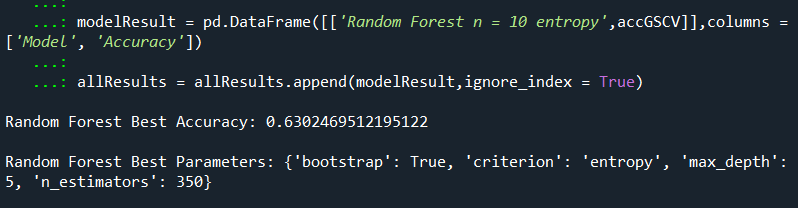
**Part C: Optimization phase:**

Hyperparameter tuning can be done using the python library 'GridSearchCV'.

Two RF models are to be tuned: one with Gini Indexing as splitting criterion and the other with standard entropy.

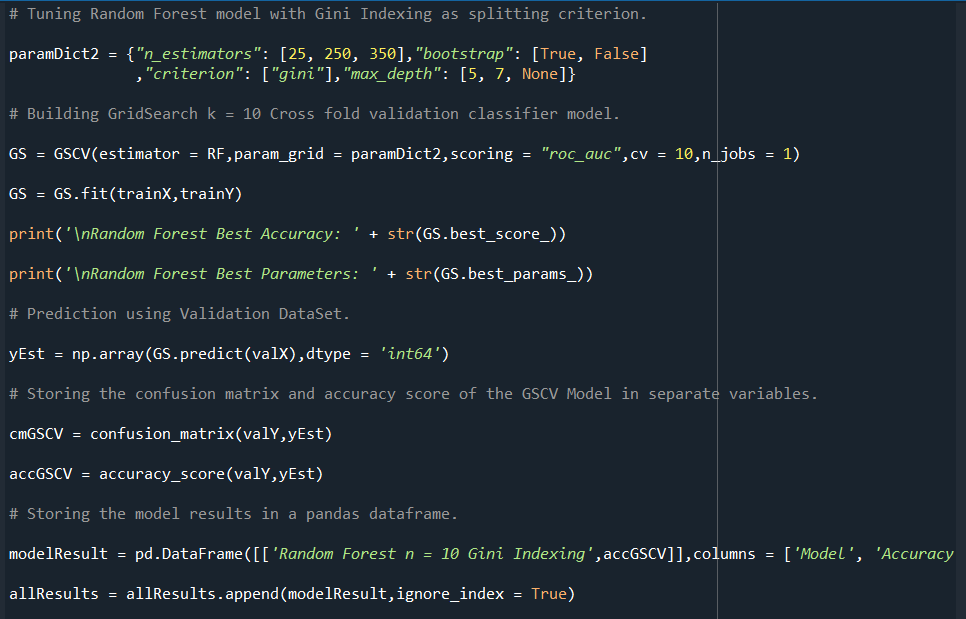


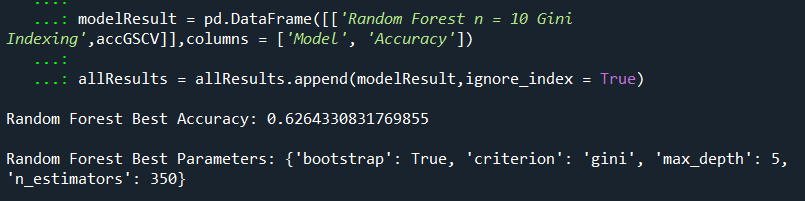
RF Model1 (entropy) uses 25, 250 and 350 estimators at least with optional bootstrapping and optional max depth ranging from 5 and 7. The best model would be chosen using the Operating Area under the Curve region (AUC).



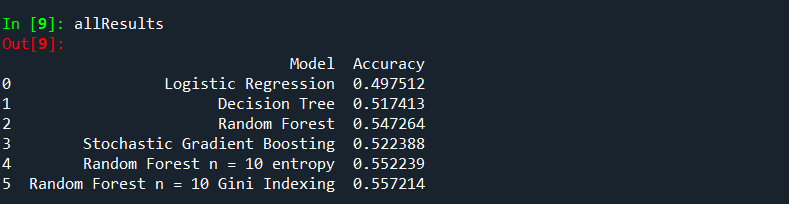
The ‘GridSearchCV’ says that it is better to have a RF model with bootstrapping, max depth = 5 and 350 estimators when entropy is used as splitting criteria.

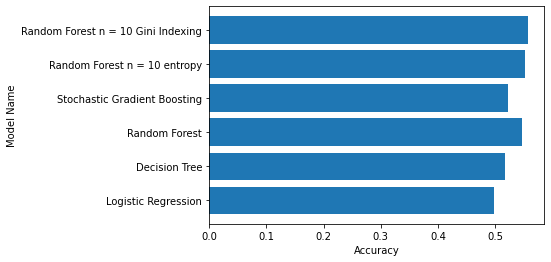
RF Model2 (Gini Indexing) uses the same Grid parameters as that of Model1.



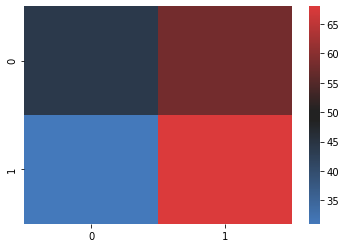


The ‘GridSearchCV’ says that it is better to have a RF model with bootstrapping, max depth = 5 and 350 estimators when Gini Indexing is used as splitting criteria.





We can see a slight increase in accuracy after hyper parameter tuning. We can conclude that the accuracy for this image classification problem would be better if Gini Indexing is used on a Random Forest model. This can be confirmed by the heat map of the confusion matrix of RF Model2.



The True Negative (TN) value is the highest when compared to the other values. The False Positive (FP) value is the second highest followed by the True Positive (TP) and the False Negative (NP) value is the lowest among all.