Machine Learning Engineer Nanodegree

Capstone Project Report

Vibhav Chaturvedy
16th, October,2017

I. Definition

Project Overview

There are estimated to be nearly half a million species of plants in the world. Classification of species has been historically problamatic and often results in duplicate identifications, Automating plant recognition might have many applications,including:

- Species population tracking and preservation
- Plant-based medicinal research
- Crop and food supply management

In the recent few years the emergence of new organised data ,high computaional power and successful researchs in machine learning has given us the ability to use a computer to predict the species of a plant .

Problem Statement

The objective of this project is to use binary leaf images and extracted features, including shape, margin and texture, to accurately identify 99 species of plants. Leaves due to their volume, prevelance, unique characteristics, are an effective means of differentiating plant species. They also provide a fun introduction to applying techniques that involve image-based features.

The project consist of deploying a deep learning model trying to find accuracy for the given dataset to classify the given image of a leaf ,using for that objective a set of variables of different types (like boolean and categorical), divided into train and test datasets.

Evalutation metrics:

As it is required in the kaggle competition, the evaluation Metric will be the multi-class logarithmic loss i.e. logloss over the dataset. For each image, it is required to submit a set of predicted probabilities(one for every species).The formula is then,

$$logloss = -1N \sum i=1N \sum j=1 Myij \log(pij),$$

where N : no of images in test set
M : no of species labels
yij : is 1 if observation i is in class j otherwise 0
pij :is predicted probability that observation i belongs to class j.
Log loss metrics is been used to find the accuracy of the training set by using History methory of keras using theano backend.

To find accuracy of the testing process we have used accuracy_score method from sklearn.metrics package. The reason for chossing accuracy is that purpose of this project is to be as efficient as possible when predicting the probablities for each species.Hence, metrics like r2_score which determine how close the prediction is to the real value will not serve the purpose necessarily.

II. Analysis

Data Exploration

The dataset consists approximately 1,584 images of leaf specimens (16 samples each of 99 species) which have been converted to binary black leaves against white backgrounds. Three sets of features are also provided per image: a shape contiguous descriptor, an interior texture histogram, and a fine-scale margin histogram. For each feature, a 64-attribute vector is given per leaf sample.

Note that of the original 100 species, i have eliminated one on account of incomplete associated data in the original dataset.

File Description:

- **train.csv :** the training set which contains species id their name and their corresponding features.

	id	species	margin1	margin2	margin3	margin4	margin5	margin6	margin7	margin8	...	texture55	texture56	texture57	texture58
0	1	Acer_Opalus	0.007812	0.023438	0.023438	0.003906	0.011719	0.009766	0.027344	0.000000	...	0.007812	0.000000	0.002930	0.002930
1	2	Pterocarya_Stenoptera	0.005859	0.000000	0.031250	0.015625	0.025391	0.001953	0.019531	0.000000	...	0.000977	0.000000	0.000000	0.000977
2	3	Quercus_Hartwissiana	0.005859	0.009766	0.019531	0.007812	0.003906	0.005859	0.068359	0.000000	...	0.154300	0.000000	0.005859	0.000977
3	5	Tilia_Tomentosa	0.000000	0.003906	0.023438	0.005859	0.021484	0.019531	0.023438	0.000000	...	0.000000	0.000977	0.000000	0.000000
4	6	Quercus_Variabilis	0.005859	0.003906	0.048828	0.009766	0.013672	0.015625	0.005859	0.000000	...	0.096680	0.000000	0.021484	0.000000
5	8	Magnolia_Salicifolia	0.070312	0.093750	0.033203	0.001953	0.000000	0.152340	0.007812	0.000000	...	0.145510	0.000000	0.041992	0.000000
6	10	Quercus_Canariensis	0.021484	0.031250	0.017578	0.009766	0.001953	0.042969	0.039062	0.000000	...	0.085938	0.000000	0.040039	0.000000
7	11	Quercus_Rubra	0.000000	0.000000	0.037109	0.050781	0.003906	0.000000	0.003906	0.000000	...	0.038086	0.025391	0.009766	0.002930
8	14	Quercus_Brantii	0.005859	0.001953	0.033203	0.015625	0.001953	0.000000	0.023438	0.000000	...	0.000000	0.000000	0.008789	0.000000
9	15	Salix_Fragilis	0.000000	0.000000	0.009766	0.037109	0.072266	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.070312

- **test.csv :** the testing set which contains species id and their corresponding features based on which predictions are done and model accuracy is calculated.

	id	margin1	margin2	margin3	margin4	margin5	margin6	margin7	margin8	margin9	...	texture55	texture56	texture57	texture58	texture59	texture60
0	4	0.019531	0.009766	0.078125	0.011719	0.003906	0.015625	0.005859	0.000000	0.005859	...	0.006836	0.000000	0.015625	0.000977	0.015625	0.000000
1	7	0.007812	0.005859	0.064453	0.009766	0.003906	0.013672	0.007812	0.000000	0.033203	...	0.000000	0.000000	0.006836	0.001953	0.013672	0.000000
2	9	0.000000	0.000000	0.001953	0.021484	0.041016	0.000000	0.023438	0.000000	0.011719	...	0.128910	0.000000	0.000977	0.000000	0.000000	0.000000
3	12	0.000000	0.000000	0.009766	0.011719	0.017578	0.000000	0.003906	0.000000	0.003906	...	0.012695	0.015625	0.002930	0.036133	0.013672	0.000000
4	13	0.001953	0.000000	0.015625	0.009766	0.039062	0.000000	0.009766	0.000000	0.005859	...	0.000000	0.042969	0.016602	0.010742	0.041016	0.000000
5	16	0.021484	0.033203	0.021484	0.009766	0.015625	0.035156	0.039062	0.000000	0.003906	...	0.000000	0.000000	0.000000	0.000977	0.049805	0.000000
6	19	0.015625	0.025391	0.046875	0.009766	0.005859	0.027344	0.042969	0.000000	0.000000	...	0.001953	0.000000	0.000000	0.004883	0.030273	0.000000
7	23	0.007812	0.031250	0.011719	0.050781	0.000000	0.117190	0.003906	0.000000	0.011719	...	0.047852	0.000000	0.030273	0.000000	0.011719	0.000000
8	24	0.003906	0.007812	0.074219	0.017578	0.015625	0.003906	0.011719	0.000000	0.009766	...	0.102540	0.000000	0.023438	0.000000	0.000000	0.000000
9	28	0.000000	0.000000	0.005859	0.021484	0.054688	0.000000	0.015625	0.000000	0.011719	...	0.195310	0.039062	0.003906	0.007812	0.013672	0.001953

- **sample_submission.csv :** a sample submission file in correct format in which output is arranged in systematic format.

- **images :** the image files (each image is named with its corresponding id)

Data Fields:

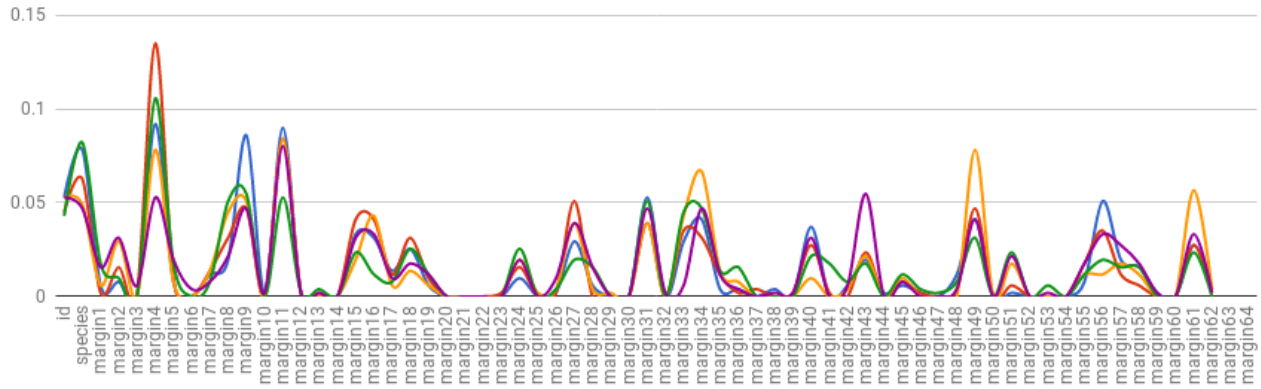
- **id** :an anonymous id unique to an image
- **margin_1, margin_2, margin_3, ..., margin_64 :** each of the 64 attribute vectors for the margin feature
- **shape_1, shape_2, shape_3, ..., shape_64 :**each of the 64 attribute vectors for the shape feature
- **texture_1, texture_2, texture_3, ..., texture_64 :** each of the 64 attribute vectors for the texture feature.

Exploratory Visualization:

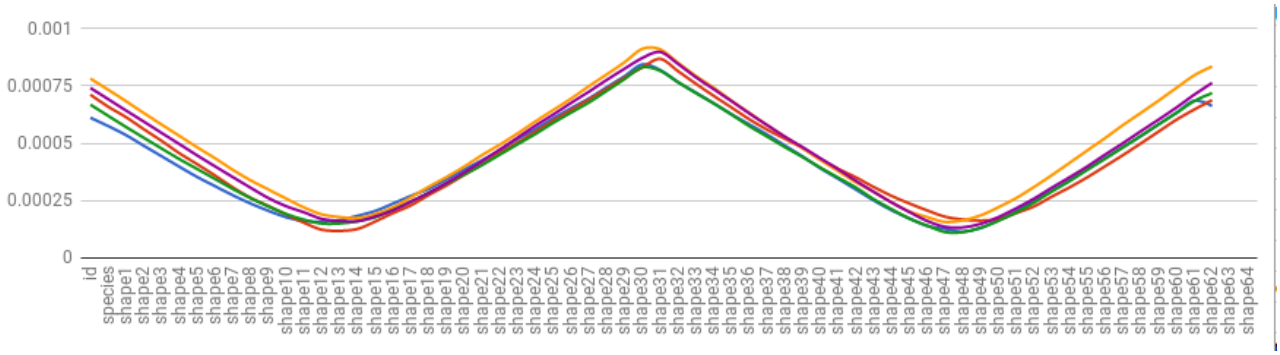
Lets compare two species among given 99:

Eucalyptus_Urnigera *:* We took data of different five id's with same species name (265, 310, 393, 660, 772).

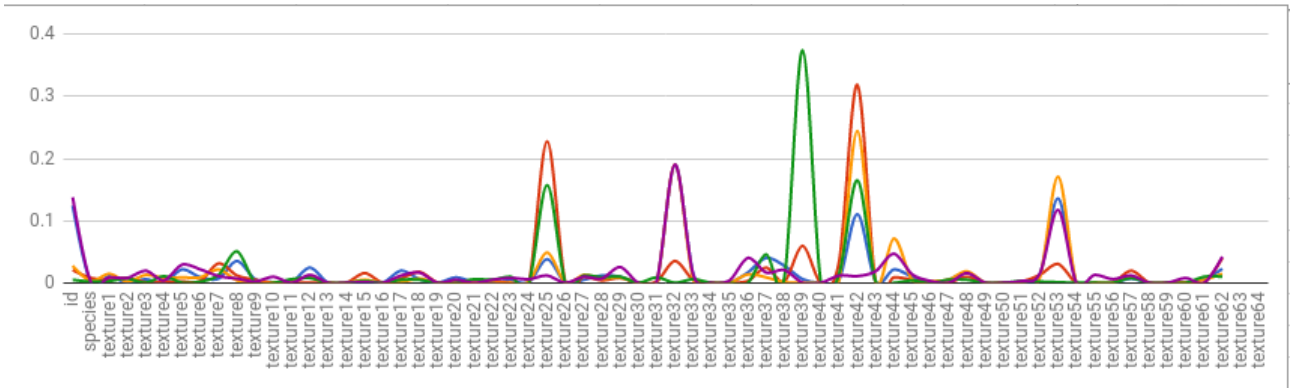
Margin graph:



Shape graph:



Texture:



Images:


265.jpg


310.jpg


393.jpg


660.jpg

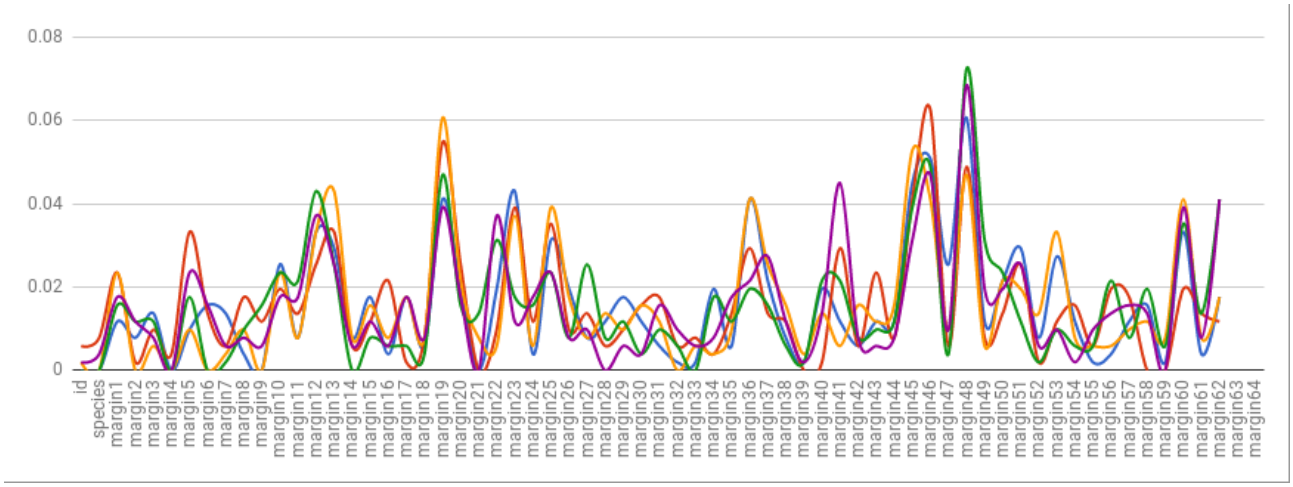

772.jpg

We can see from above three graphs that all the five id's show similar trend as if they resembles the same species. To further check our prediction we extracted images of given id's from our image dataset. And we can see that all these five images look similar to each other.

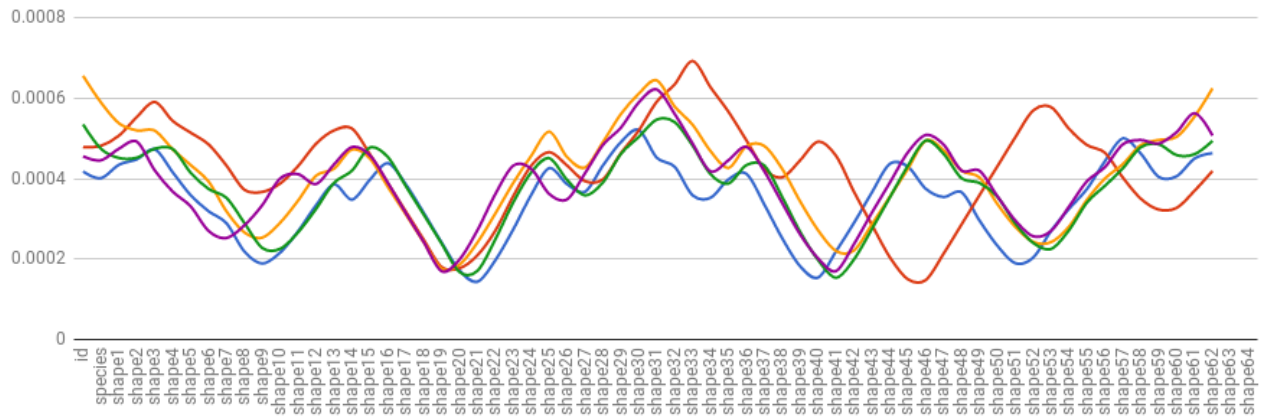
Let check this for another type of species:

Castanea_Sativa:

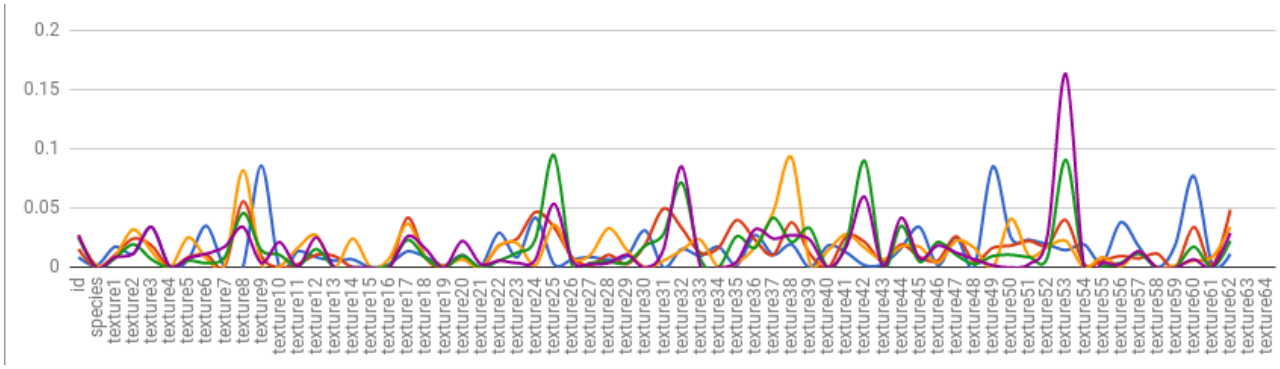
Margin graph:



Shape graph:



Texture:



Images:

From above images we can see that five of them look similar as if belong to same species especially when it comes to shape and margin.

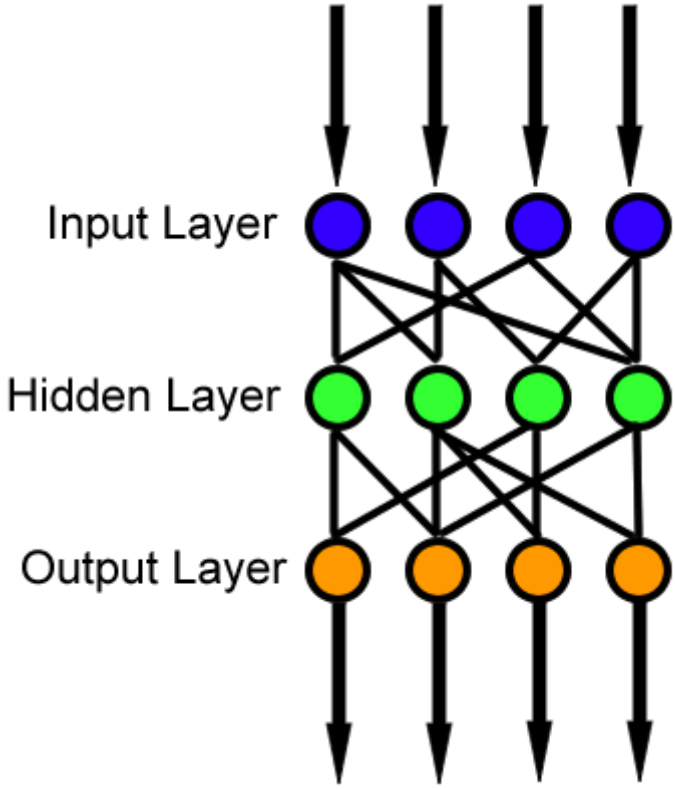
Algorithms and Techniques:

For this problem, many machine learning algorithm like RandomForestClassifier, Logistic Regression could be used,but i decided to build up a feed forward neural network to solve

this problem as i am a beginner in deep learning field and for capstone,i wanted to take up a project from which i can learn more about deep learning.

Feed Forward Neural Network:

A feedforward neural network is an artificial neural network wherein connections between the units do *not* form a cycle. The feedforward neural network was the first and simplest type of artificial neural network devised.In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network.



StratifiedShuffeSplit Cross Validation:

In **stratified** k-fold **cross-validation**, the folds are selected so that the mean response value is approximately equal in all the folds. In the case of a dichotomous classification, this means that each fold contains roughly the same proportions of the two types of class labels.This cross-validation object is a merge of StratifiedKFold and ShuffleSplit, which returns stratified randomized folds. The folds are made by preserving the percentage of samples for each class.
Note: like the ShuffleSplit strategy, stratified random splits do not guarantee that all folds will be different, although this is still very likely for sizeable datasets.
In this project 20% of training data has been used for cross validation process.

Benchmark:

The main models that will be used as bechmark are those published by the community over the discussion forum of the competition. The main references that will be used in a first stage will be:

https://www.kaggle.com/vibhav8/neural-network-through-keras

https://www.kaggle.com/vibhav8/feature-extraction-from-images

In the first kernel, they have implemented a three layered neural network by using keras . They have plotted an error versus no of iterations graph to depict the decrement of error with increase in number of iterations.

In the second kernal, they have used many machine learning models like RandomForestClassifier and calculated their accuracy and log loss score to compare each other.

In relation to the performance of the model (log loss) the benchmark will be those provided for the kaggle platform.

III. Methodology

Data processing:

Label Encoding: Our training dataset contains column named 'species' containing the name of different species in string format. Hence, it need to be encoded in integer form for further processes. Sklearn provides function LabelEncoder for this purpose.

```
## Importing sklearn libraries
from sklearn.preprocessing import LabelEncoder
```

Standard Scalar: The StandardScaler assumes our data is normally distributed within each feature and will scale them such that the distribution is now centred around 0, with a standard deviation of 1. After splitting dataset into label and features i.e. X_train and y_train, it is necessary to apply feature scaling to x_train data to get better performance of the model.

The mean and standard deviation are calculated for the feature and then the feature is scaled based on:

$xi–mean(x)stdev(x)$

Standard Scaler funtion in sklearn helps in feature scaling:

```
from sklearn.preprocessing import StandardScaler
```

```
X_train_enc = StandardScaler().fit(X_train).transform(X_train)
X_test_enc =StandardScaler().fit(X_test).transform(X_test)
```

Stratified Shuffle Split: Provides train/test indices to split data in train/test sets.This cross-validation object is a merge of StratifiedKFold and ShuffleSplit, which returns stratified randomized folds. The folds are made by preserving the percentage of samples for each class.
20 % of the training data was reserved as a validation data and splitted into y_test and x_test. Hence in whole data was split into four parts – x_train, y_train, x_test, y_test. This step is necessary as it makes the model more generalized and hence further preventing it from overfitting.

```
##Stratified Train/Test Split¶
sss = StratifiedShuffleSplit(labels, 10, test_size=0.2, random_state=23)
```

```
for train_index, test_index in sss:
    X_train, X_test = train.values[train_index], train.values[test_index]
    y_train, y_test = labels[train_index], labels[test_index]
```

One Hot Encoding: In one-hot encoding only one bit of the state vector is asserted for any given state. All other state bits are zero. Thus if there are *n* states then *n* state flip-flops are required. As only one bit remains logic high and rest are logic low, it is called as One-hot encoding.

It is faster than other encoding techniques. Speed is independent of number of states, and depends only on the number of transitions into a particular state.In our model (y_train) which was already labeled encoded has been further hot encoded before training the model. In keras , one hot technique can be done by using function to_categorical.

```
## Keras Libraries for Neural Networks
from keras.utils.np_utils import to_categorical
```

```
## We will be working with categorical crossentropy function
## It is required to further convert the labels into "one-hot" representation
y_cat_train = to_categorical(y_train)
y_cat_train.shape
```

Implementation:

Data Preprosessing:

- I was given two datasets in the form of csv format namely train.csv and test.csv.

- Firstly species in the form of string in training data was labeled encoded to convert them into 'int' format. Further processes were done on test data like saving test ids and columns for submission.This was done under ' encode ' function as shown below.

```
# Swiss army knife function to organize the data
def encode(train, test):
    le = LabelEncoder().fit(train.species)
    labels = le.transform(train.species)           # encode species strings
    classes = list(le.classes_)                    # save column names for submission
    test_ids = test.id                             # save test ids for submission
    train = train.drop(['species', 'id'], axis=1)
    test = test.drop(['id'], axis=1)
    return train, labels, test, test_ids, classes
train, labels, test, test_ids, classes = encode(train, test)
train.head(1)
```

- Now cross validation is done on training set by using StratifiedShuffle Split technique. 20% of total training data was used for testing or as validation set. Now the data was split into four parts namely x_train, y_train, x_test , y_test.
- Feature scaling of data by using sklearn Standard Scalar technique is done on x_train and x_test.
- Further y_train set is one hot encoded by using keras to_categorical technique.

Model training and prediction:

- Firstly a three layered neural network(feed forward neural network) is developed.
- Model selected was keras Sequential model.
- Model consisted of three layers namely Input layer, one hidden layer and an output layer.
- Between two successive layers a dropout layer was introduced to prevent the model from overfitting.
- Activation layer used after input and hidden layer used ' ReLu' activation function.
- After output layer we have used ' softmax' activation function.
- Dimension of an individual input in input layer is '192'.
- Number of nodes selected for hidden and input layer was decided by experimenting the model with different number of nodes and the combination giving the highest accuracy was choosen.

```
## Developing a layered model for Neural Networks
## Input dimensions should be equal to the number of features
## We used softmax layer to predict a uniform probabilistic distribution of outcomes
model = Sequential()
model.add(Dense(770,input_dim=192))
model.add(Dropout(0.2))
model.add(Activation('relu'))
model.add(Dense(770))
model.add(Dropout(0.3))
model.add(Activation('relu'))
model.add(Dense(99))
model.add(Activation('softmax'))
```

- Model was then compiled using multi class log loss funtion to find error.
- Further fitting of model is done taking batch size as 128 and number of epochs 50.

- Using accuracy score method test accuracy was found.

```
# finding test accuracy percentage
from sklearn.metrics import accuracy_score
acc = accuracy_score(y_test, y_pred)
print("Accuracy: {:.4%}".format(acc))
```

Refinement:

Initially the data was not cross validated. The training accuracy came out to be very less (20%). After cross validating the training data and putting 20% of the data as validation set, training data accuracy came out to be surprisingly high i.e. about 97% but it seems like the model was overfitting as testing accuracy was very less i.e. 35%.To improve the testing accuracy model was tested with varying parameters like epochs , number of nodes in input and hidden layer. Initially sigmoid activation function was used between the layers but 'Relu' function performed much better. For this type of data, it was observed that taking less number of nodes in input and hidden layer made the model more accurate than taking large number of nodes hence it came out that taking following values for parameters resulted into maximum test accuracy:

number of input layer node: 770
number of hidden layer nodes: 770
number of epochs: 50

IV. RESULTS

Model Evaluation and Validation:

1) Feed Forward Neural Network:

Without cross_validation:

- Accuracy: 0.35734
- Log_loss: 0.45654

With cross_validation:

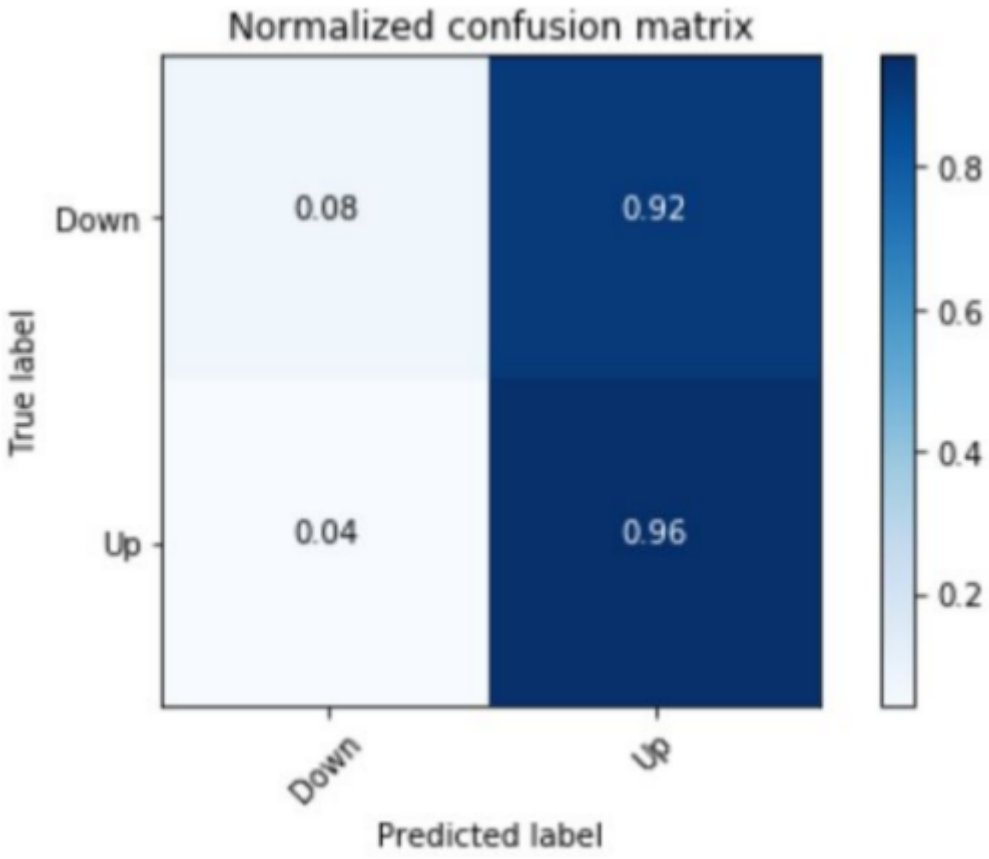
- Accuracy: 0.63435
- Log_loss: 0.0034

Although using classifiers like RandomForest Classifier we can get much higher accuracy and lower log_loss.

2) RandomForestClassifier:

- Accuracy: 0.9875
- Log_loss: 0.0014

which is much higher than that achieved by my model.But as i already said that i wanted to take up a deep learning project so i choose feed forward neural network model and tried to achieve the maximum accuracy.
With accuracy in hand now its time to check out for robustness. I shall approach it using a confusion matrix to see how well it predicts the species. Given below is the normalized confusion matrix:



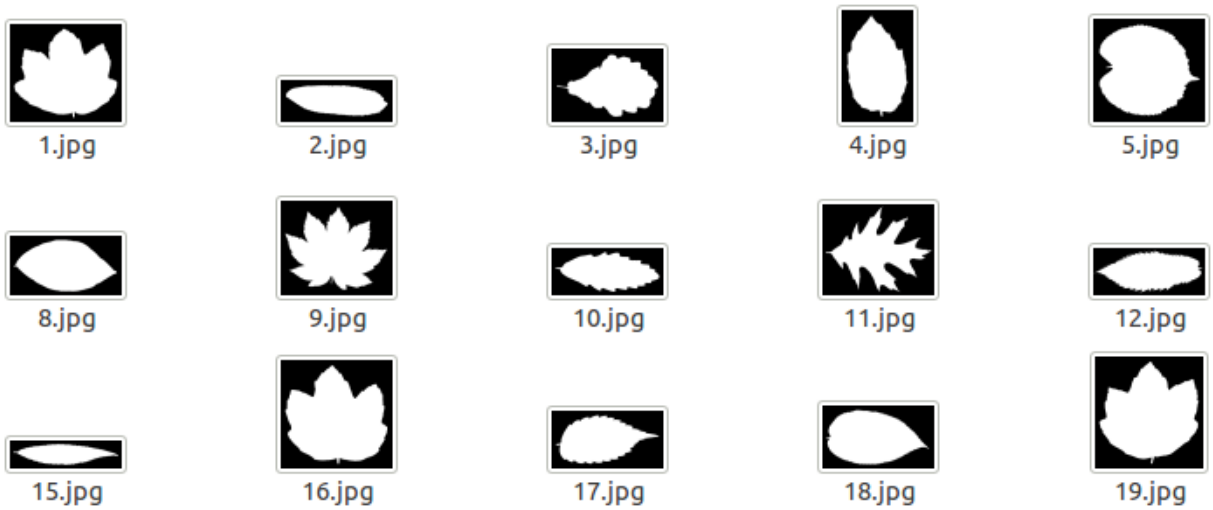
This leads to the realization that the model has generalized for a class and hence is not robust. This is something to work upon later on and hence can be included in the improvement section.

Justification:

As per the benchmark section, i had hoped to achieve or surpass the accuracy of 75.56%. However, the vast difference in dataset made me realize that it was a very tough task. Hence, my goal shifted to that of surpassing the conventional human prediction on basis of trends which is about 45-50% accurate. My result 63.75% accuracy very well lie above this standard and hence, I can justify the progress and final output of my approach. However. I realized it is not robust and that the project has a long way to go and can be mad bettter in several different ways.

V. Conclusion

Free-Form Visualization:



As data i was provided with two csv files (train.csv and test.csv) and a set of images with their corresponding id's. The features given were shape ,texture and margin of each leaf. Each feature has 64-attributes. Providing images with some of pre-defined features clearly indicates that the images could be used to extract more features out of it. But as an experimentation i tried to depend wholly on pre defined features without using images to extract more features. With only these given features i tried to train my neural network model and surprisingly it gave pretty good results which can be further improved by using images also. But as my capstone project requirement i decided not to use image dataset. This was a special feature i found about this project.

The above images shows that leaves are drawn in black background which makes the task of feature extraction easier.

Reflection:

The process for this project can be summarized using the following steps:

1. Identify an interesting problem and find a public dataset. (which i easily got from kaggle)
2. Download and preprocess the data.
3. Identify a benchmark for the classifier.
4. Research various algorithms and techniques for solving the problem
5. Implement a model and train it using the data.
6. Refine the model until a good set of parameters are found.

I found step 4 and 6 the most challenging. Already in the beginning i found out that i can gain an accuracy above 90% by using classifiers like RandomForestClassifier then too i decided to do this project by using neural network. Although this was a challenging task but it gave me much knowledge about how deep learning models work. I had to familiarize myself with the dataset, search the internet for research papers describing how the researchers have approached the problem in the past and understanding the different techniques they used, finally i had to implement a combination of those techniques. Finding good set of parameters was also a tedious task .Selecting number of nodes ,selecting number of epochs etc. I read many research papers and blogs ,some gave different formuals for finding out these set of parameters ,some said its just the work of experimentation. The most interesting aspect of the project for me was understanding the “feed forward neural network” model specifically how it tunes respectives weights on each node to get out with the maximum accuracy.

Improvement:

I saw many people who took part in this leaf classification challenege in kaggle were able to attain accuracy above 80%. Hence even after submission and acceptance of the project i will continue to work on it. The major changes required in my opinion are:

1. One aspect of the implementation that could be improved would be using the image dataset also for further feature extraction.As images consist of picture of a leaf with black background, i think it would not be too difficult to extract features out of it. I am sure this is going increase the model accuracy and performance.
2. Secondly i would like to implement this project on a CNN (Convolutional Neural Network) model. Being advantageous over feed forward NN in many aspects i hope in some areas where i faced some difficulties with my model they could be easily implemented if i had used CNN's.
3. Look into matter of robustness as the model eventually seems to generalize towards the majority class.

Libraries and Packages:

1. Scikit-learn: http://scikit-learn.org/stable/user_guide.html
2. Keras : https://keras.io/getting-started/sequential-model-guide/
3. Matplotlib : https://matplotlib.org/users/pyplot_tutorial.htm

Acknowledgement:

[1] J.-X. Du, X.-F. Wang, and G.-J. Zhang, “Leaf shape based plant species recognition,” Applied Mathematics and Computation, vol. 185, 2007.

[2] H. Fu and Z. Chi, “Combined thresholding and neural network approach for vein pattern extraction from leaf images,” IEE Proceedings-Vision, Image and Signal Processing
vol. 153, no. 6, December 2006

[3] https://www.kaggle.com/jeffd23/10-classifier-showdown-in-scikit-learn

[4] http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html