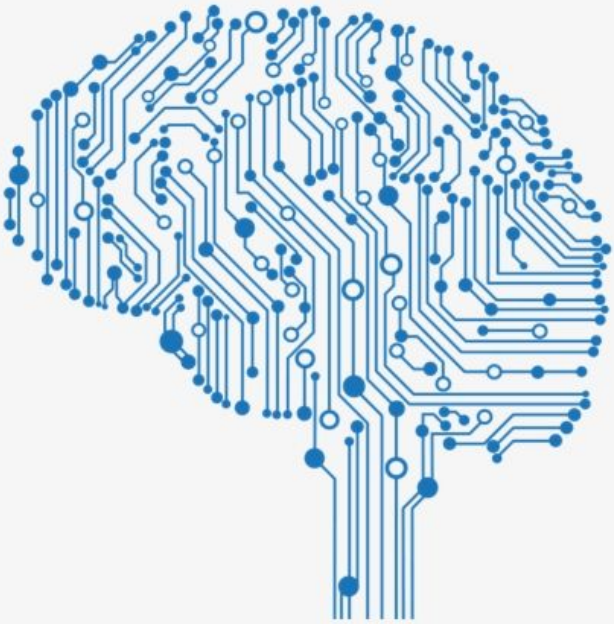
UE17CS303

Machine Learning Assignment



Gaurang Rao - PES1201701103

Anantharam R U - PES12017000088

Adam Rizk - PES1201802117

Problem Statement

To classify stellar objects into spectrometric classes based on photometric data.

Spectrophotometry is the quantitative measurement of the reflection or transmission properties of a material as a function of wavelength.

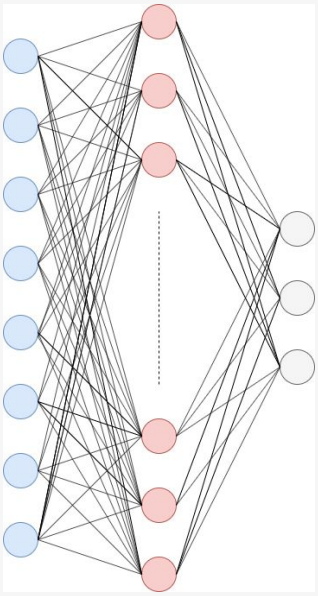
In astronomy, spectrophotometry is the measurement of the spectrum of a celestial object in which the flux scale (rate at which energy is transmitted through electromagnetic radiation) as a function of wavelength.

Our dataset consisted of observed photometric data for stellar objects, and their corresponding classes. The classes include 0, 1 and 2.

The dataset consisted of about 8,400 observations, and 24 features, out of which 8 were independent, excluding the class column.

Techniques Employed

a) Artificial Neural Network



An Artificial Neural Network tries to find optimal weight and bias values, to classify data given to the Neural Network.

We build an ANN with the following constraints:

1. Eight input neurons, corresponding to eight unique features.
2. Three output neurons, corresponding to the three classes.
3. A single hidden layer, with a variable number of neurons.

We initialise the bias values for the neurons in the model to zero.

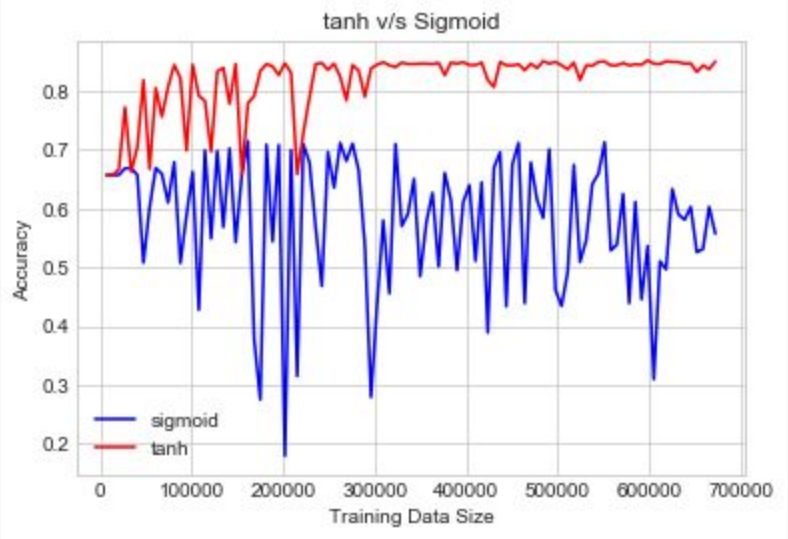
Since very low values or very high values of weights, when used with activation functions such as sigmoid or tanh, can cause the vanishing gradient problem, and could lead to longer training times, we employ the following technique to avoid this:

We use a normalisation factor of $2/\sqrt{size(l-1)+size(l)}$ that is multiplied to our initial random weights (drawn from a normal distribution), between layers $l-1$ and l.

We further employ stochastic gradient descent to fix the weights and biases at every iteration, and run a back-propagation algorithm to do so.

As the dataset consists only of a little over 8,000 observations, we randomly sampled 80% of the dataset for every iteration of the model, until the accuracy reached an acceptable value.

Sigmoid shows us fairly mediocre results, however tanh, shows promising accuracy values for our test dataset, reaching a peak of 85% at some point. We try different values of learning rates, and see that we get good results in the ranges of [0.01, 0.1].

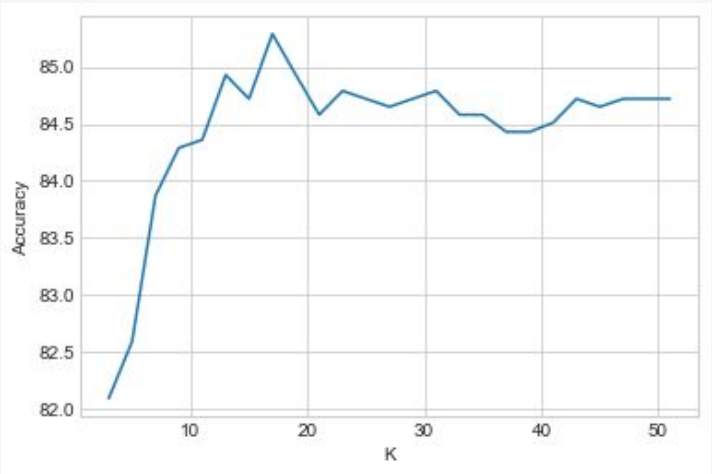


We calculate the precision and recall for our testing data within the code, by building a confusion matrix and updating the matrix at the end of every classification by the Neural Network.

b) K-Nearest-Neighbours

A K-Nearest-Neighbours classifier, considers the K-closest observed data points, and classifies it based on the most occuring class within these K points.

The model has no training time, and the only time taken is during classification of a new instance, as it generally requires finding the distance between the new instance and all other existing data points.



We test our model against several values of K, to find the optimal value for this dataset, based on the accuracy provided by the models.

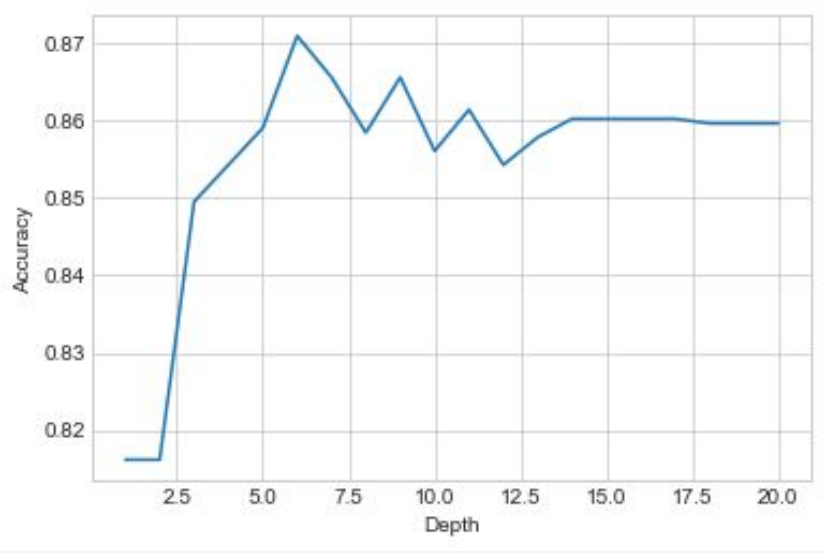
This model was written in C, and it too calculates precision and recall within the code itself.

This model peaks at an accuracy of ~86% at a K value of 17.

c) Decision Trees :

Decision tree is a powerful and popular tool for classification and prediction. It is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

We have taken the training dataset to be the standard 80% of the original dataset (8400 rows) and the test data to be the remaining 20%. The datasets were **randomly sampled**.



Since the dataset is continuous, we have split the nodes based on the condition '$\leq$', and we have taken the split value to be the mean of every two consecutive rows. We have found the best split using **entropy**.

$$E = \sum - p \log(p)$$

Initially the model was trained without giving a constraint on the depth of the model . That is, it was trained on all the rows in the training dataset and gave an accuracy of about 85-86 %.

Then we added the depth constraint on the model and repeatedly calculated the accuracy of the model for different depth values (it is to be noted that this was done keeping the training data and test data fixed). Doing this, the model gave an accuracy of 87% when the depth was 6 as shown in the graph.

We have also added code to find the precision and recall for each class using the model.

d) Random Forest:

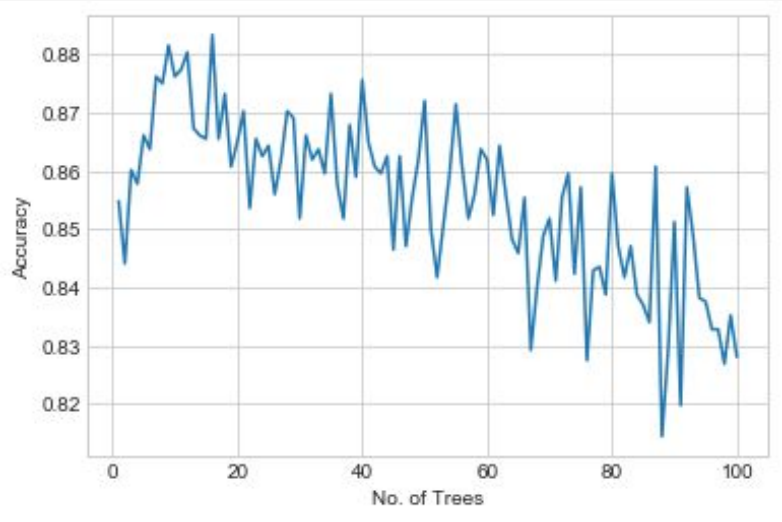
Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees. **In our case the final output is the class that is the mode of the classes predicted by each individual tree.**

Again, we have used the standard 80% training and 20% training dataset and they were **randomly sampled**.

Initially, we manually checked for different number of trees and got a maximum accuracy of 87% .

We then repeatedly executed the model and found the accuracies with the values of number of trees ranging from 1-100. By doing this we managed to achieve an accuracy of 88 % as shown in the graph. On one execution we even managed to get an accuracy of 88.9 ~ 89 %.

We have also added code to find the precision and recall for each class using the model.



Results & Conclusions

	Accuracy
ANN	85%
KNN	85%
Decision Trees	87%
Random Forest	89%

We don't mention the recall and precision here as they are calculated individually for each class within the code, for every model.

From the observed values, class 1 almost always had high precision and recall values, and class 0 always showed the lowest values for precision and recall.

From the above results, our Random Forest model seems to give us the best accuracy, and hence turns out to be the best option as a classifier for this dataset.