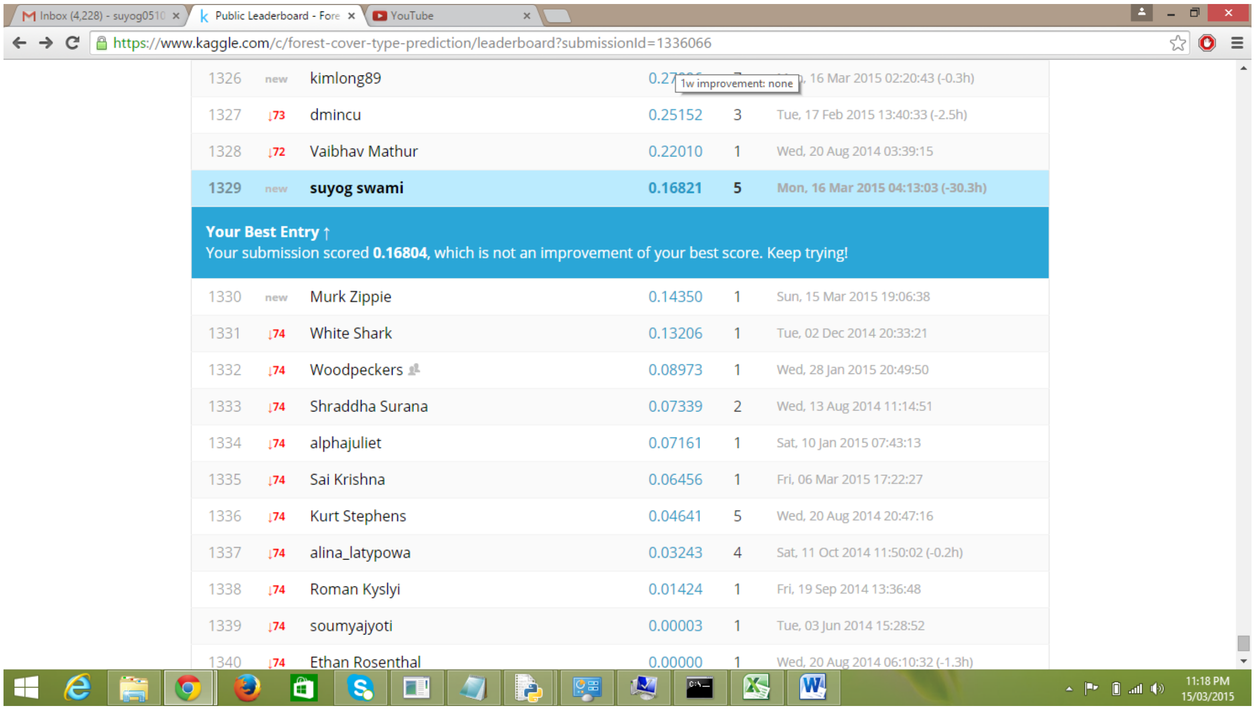
# CSE 4334/5334 – Data Mining Spring 2015 – Programming Assignment 2

Kaggle Username: suyog swami

Kaggle Rank: 1329

Kaggle Score: .16821



https://www.kaggle.com/c/forest-cover-type-prediction/leaderboard?submissionId=1336066

| | | | | | |
|---|---|---|---|---|---|
| 1326 | new | kimlong89 | 0.27[illegible] | [illegible] | [illegible], 16 Mar 2015 02:20:43 (-0.3h) |
| 1327 | ↓73 | dmincu | 0.25152 | 3 | Tue, 17 Feb 2015 13:40:33 (-2.5h) |
| 1328 | ↓72 | Vaibhav Mathur | 0.22010 | 1 | Wed, 20 Aug 2014 03:39:15 |
| **1329** | new | **suyog swami** | **0.16821** | **5** | **Mon, 16 Mar 2015 04:13:03 (-30.3h)** |
| 1330 | new | Murk Zippie | 0.14350 | 1 | Sun, 15 Mar 2015 19:06:38 |
| 1331 | ↓74 | White Shark | 0.13206 | 1 | Tue, 02 Dec 2014 20:33:21 |
| 1332 | ↓74 | Woodpeckers | 0.08973 | 1 | Wed, 28 Jan 2015 20:49:50 |
| 1333 | ↓74 | Shraddha Surana | 0.07339 | 2 | Wed, 13 Aug 2014 11:14:51 |
| 1334 | ↓74 | alphajuliet | 0.07161 | 1 | Sat, 10 Jan 2015 07:43:13 |
| 1335 | ↓74 | Sai Krishna | 0.06456 | 1 | Fri, 06 Mar 2015 17:22:27 |
| 1336 | ↓74 | Kurt Stephens | 0.04641 | 5 | Wed, 20 Aug 2014 20:47:16 |
| 1337 | ↓74 | alina_latypowa | 0.03243 | 4 | Sat, 11 Oct 2014 11:50:02 (-0.2h) |
| 1338 | ↓74 | Roman Kyslyi | 0.01424 | 1 | Fri, 19 Sep 2014 13:36:48 |
| 1339 | ↓74 | soumyajyoti | 0.00003 | 1 | Tue, 03 Jun 2014 15:28:52 |
| 1340 | ↓74 | Ethan Rosenthal | 0.00000 | 1 | Wed, 20 Aug 2014 06:10:32 (-1.3h) |

1w improvement: none

**Your Best Entry ↑**
Your submission scored **0.16804**, which is not an improvement of your best score. Keep trying!

Design and Implementation:

Classifier: Decision tree - ID3.

IDE : Spyder 2.3.1

Python version: 3.4.2

Details: In this programming assignment I have used decision tree ID3 classifier to predict the forest Cover Type results. In this program there is a single python code file containing 9 functions as follows:

1. main(): where I accept the two csv files and give calls to the createTree and predict fumction. The function is also used to get the attributes and data of both the files. The operation to write into the result.csv file is also done in this function.
2. createTree(): In this function the main recursive functionality of the program is placed. It contains function calls to five other functions. A) mostCommonAttriVal() B) chooseBestAttri() C)

getColVal() and D) getCurrentVal E) createTree()[This works recursively]. Also the stop condition for the recursion is check in the beginning of the createTree() function.

3. mostCommonAttriVal(): This function is used to calculate the most common value for the attribute. The value of this function is used to check the error condition if the data set is empty or there is only one attribute in the dataset.
4. chooseBestAtrribute(): this function is used to decide which is the best attribute to perform the create three operation in the specific run. The information gain and entropy are calculated from this function which is explained further.
5. getColVal(): This function is used to get the column values for the best attribute decided in the chooseBestAttribute function.
6. getCurrentVal(): This function is used to get the dataset from the training dataset which doesn’t include records of the best attribute. This dataset is used for next round of recursion.
7. infoGain(): Information gain which is Entropy of the decision attribute – (freq of best attribute)* entropy of the bestattribute. This function is called in a for loop from the chooseBestAttribute function. It returns the gain to the chooseBestAtrribute function. A call to calculate Entropy is given from this function.
8. calculateEntropy(): This function calculates entropy using the entropy function discussed in the class.
9. predict(): After generating the tree. Predict function is used to predict the output of the test.csv’s Cover_Type. Here I have used a random function to fill the places whose Cover_Type values the classifier can’t predict. Also a hard coding is done to check if the value of testdata attributes is nearest to the values in train and predict the result.

It was an overly exhaustive assignment where I feel I put more efforts than in any programming assignment done before. But I agree looking at the accuracy I wasn’t able to optimize the result efficiently. Hardcoding was the last thing I opted for. I tried many variations but was unsuccessful in doing so. One variation I tried was to place the tree nodes as ranges of the attributes and match the ranges with the test data attributes, but it became a more complex process to handle the string and integers interchangeably.