CS771 : Assignment 3 : [Group 10]

Aakash [19111261] **Dhanish Kumar [170243]**

Drashti Pathak [19111032] **Lt Cdr Dhurender [19111030]**

Rohit Chouhan [170590] **Shreya Sharma [19111087]**

QUESTION NO. 1

- Bonsai (1) follows a tree-based approach, i.e., label partitioning followed by learning classifiers at the internal node, where as at leaf nodes, linear classifiers are trained to predict the actual labels.
- The branching factor in the Bonsai method will decide the depth of tree architecture (be it diverse and shallow trees). As cited in the paper, the branching factor is set to ≥ 100, against Parabel (2) where it is set to 2.
- It has three main features, which distinguish it from state-of-the-art tree based approaches which are:

  1. **Generalized label representation**

     (a) Input space representation as a function of feature vectors.

     (b) Output space representation based on their co-occurrence with other labels.

     (c) A combination of the output and input representations.

     The algorithms learnt by partitioning the input space, output space and the joint space are Bonsai-i, Bonsai-o, and Bonsai-io respectively.

     Formally, we assume that leaf nodes have labels (1.....L) and we aim at finding K-cluster centers (c1,.......,ck)$\in$ $\mathbb{R}^n$ in an appropriate space (Input, Output or Joint) by optimizing the following:

     $$\underset{c_1,....,c_k \in \mathbb{R}^n}{\text{argmin}} [\sum_{k=1}^{K} \sum_{\ell \in c_i} d(v_\ell, c_k)]$$

     where d(.,.) represents the distance function and v_ℓ represents the vector representation of the label ℓ.

     The solution of the above equation can be found out using k-means algorithm. The k-means unconstrained clustering in Bonsai has few advantages over Parabel, which are explained in the Second part of the report.

2. **Shallow trees**

   (a) A shallow tree architecture is used to avoid error propagation in the tree cascade. This is achieved by enabling: -

      i. A flexible clustering via K-means for K > 2.

      ii. Relaxing balanced-ness constraints in the clustering step. Multi-way partitioning initializes diverse sub-groups of labels, and the unconstrained nature maintains the diversity during the entire process.

3. **Learning node classifiers**

   (a) Once the label space is partitioned into a diverse and shallow tree structure, A K-way One-vs-All linear classifier at each node are trained independently. The leaf nodes and non-leaf nodes are distinguished in the following way: -

      i. for non-leaf nodes, the classifier learns K linear classifiers separately, each maps to one of the K children. During prediction, the output of each classifier determines whether the test point should traverse down the corresponding child.

      ii. for leaf nodes, the classifier learns to predict the actual labels on the node.

- **Prediction error propagation in shallow versus deep trees**

   * Bonsai uses beam search to avoid the possibility of evaluating all nodes, where classifier decides, whether to traverse down some child node or not.

   * The above search space pruning strategy implies errors made at non-leaf nodes could propagate to their descendants, but since the Bonsai method assign high value to branching factor (of the order of 100), resulting in much shallower trees as compared to Parabel and hence reducing error propagation.

- **Algo for Label partitioning via k-mean clustering**

   1. First we obtain the Input space representation for each label ℓ in the set $\mathcal{S}$ = {1,...,L}.
   2. In next step we iteratively partition $\mathcal{S}$ (where $\mathcal{S}$ = {1,2,.....,L}) into disjoint subsets (this is achieved by K-means clustering. Also we would like to avoid propagation error in a deep tree cascade due to which we choose a relatively large value of $\mathcal{K}$ which leads to shallow trees).
      The clustering step in Bonsai first partitions $\mathcal{S}$ into $\mathcal{K}$ disjoint sets $\{\mathcal{S}_1...\mathcal{S}_k\}$. Each of the elements $\mathcal{S}_k$, of the set can be thought of as a meta-label which semantically groups actual labels together in one cluster.
   3. Then, $\mathcal{K}$ child nodes of the root are created, each contains one of the partitions.
   4. The same process is repeated on each of the newly-created $\mathcal{K}$ child nodes in an iterative manner.
   5. In each sub-tree, the process terminates either when the node's depth exceeds predefined threshold dmax(max of each of the tree) or the number of associated labels is no larger than $\mathcal{K}$.

Part 2

- **ADVANTAGES**

  * Bonsai achieves the best of both worlds - fast training which is comparable to state-of-the-art tree-based methods in Extreme Multilabel Classification, and much better prediction accuracy.

  * Bonsai sets relatively large values to the branching factor K,resulting in much shallower trees compared to Parabel, and hence significantly reducing error propagation, particularly for tail-labels i.e.,those which have very few training instances that belong to them.

    Sustaining Label Diversity: Bonsai sustains the diversity in the label space by not enforcing the balanced-ness constraint of the three, among the partitions. Smaller partitions with diverse tail-labels are very moderately penalized under this framework.

  * Initializing label diversity in partitioning : By setting $K > 2$, Bonsai allows a varied partitioning of the labels space, rather than grouping all labels in two clusters. This facet of Bonsai is especially favorable for tail labels by allowing them to be part of separate clusters if they are indeed very different from the rest of the labels.

- **DISADVANTAGES**

  * It is observed that Bonsai leads to approximately 2-3x increase in training time compared to Parabel.

  * The Bonsai partitions are more data-dependent so in case of noisy data regularization is required.

PART 3

prec@k and mprec@k result, when trained on 100 percent of train data for k = 1, 3, 5:

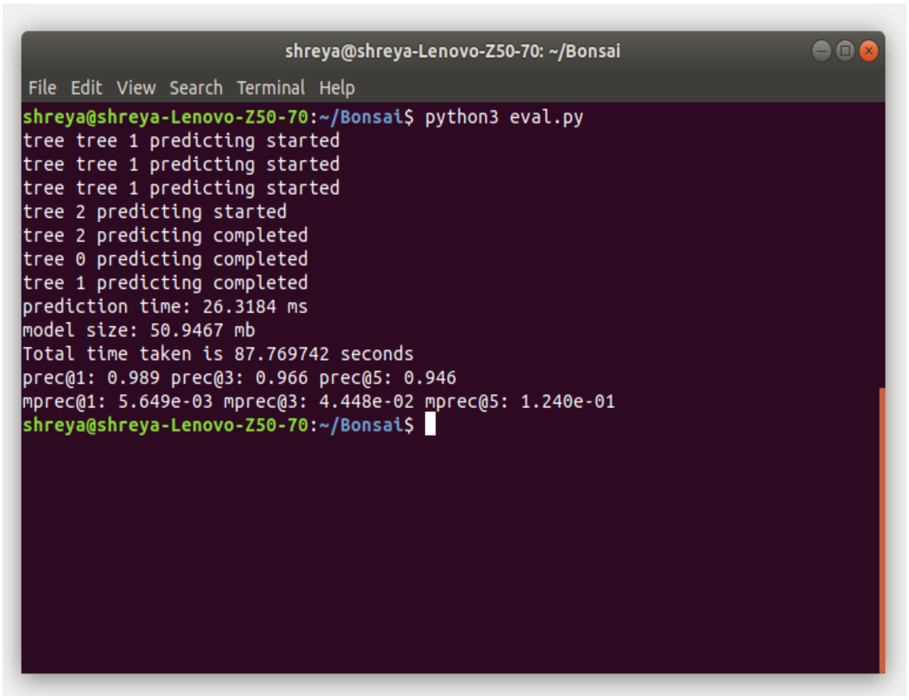
1. prec@1: 0.989
2. prec@3: 0.966
3. prec@5: 0.946
4. mprec@1: 0.2812
5. mprec@3: 0.6019
6. mprec@5: 0.4561

prec@k and mprec@k result, when trained on 80 percent of train data for k = 1, 3, 5:

1. prec@1: 0.805
2. prec@3: 0.646
3. prec@5: 0.552
4. mprec@1: 0.052
5. mprec@3: 0.137
6. mprec@5: 0.318

PART 4

- We suggest to use Bonsai (3) model which falls under the umbrella of 1-vs-All classifiers.
- In addition to Bonsai method, we have tried implementing the Parabel and FastXML methods as well. Screenshots of the implementation result, depicting the values obtained using all methods are placed below (Figure - 1, 2 and 3 respectively).
- As we can clearly observe from under-placed screenshots that, the prediction results using the Bonsai and Parabel are far better then FastXML (4) method, so we primarily focused on these two methods.
- Amongst Bonsai and Parabel also, the Bonsai method was generating the better results, in terms of prediction and model size, so we tried first using Bonsai method. However, the result of Bonsai method that we are getting, is at the cost of time. Clearly, the time taken by bonsai method is more than rest of the two methods.

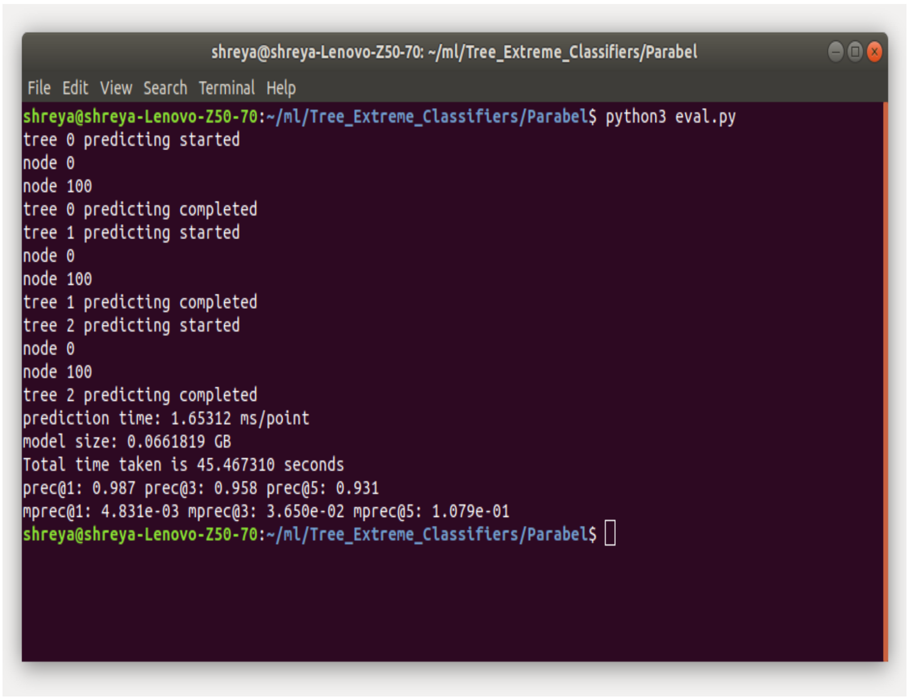

shreya@shreya-Lenovo-Z50-70: ~/Bonsai
File Edit View Search Terminal Help

```
shreya@shreya-Lenovo-Z50-70:~/Bonsai$ python3 eval.py
tree tree 1 predicting started
tree tree 1 predicting started
tree tree 1 predicting started
tree 2 predicting started
tree 2 predicting completed
tree 0 predicting completed
tree 1 predicting completed
prediction time: 26.3184 ms
model size: 50.9467 mb
Total time taken is 87.769742 seconds
prec@1: 0.989 prec@3: 0.966 prec@5: 0.946
mprec@1: 5.649e-03 mprec@3: 4.448e-02 mprec@5: 1.240e-01
shreya@shreya-Lenovo-Z50-70:~/Bonsai$
```

Figure 1: Screenshot depicting the result of Bonsai Method

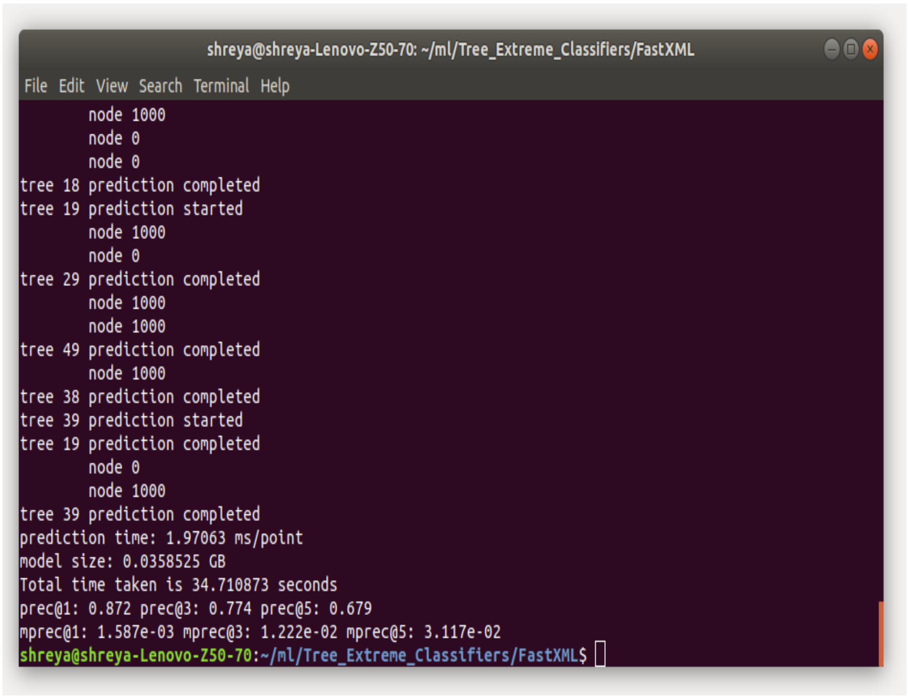
- The code (as per reference mentioned in the code file) of the Bonsai method was used directly, without modifications. There were certain changes w.r.t to parameter values in the code, we have tried, but we didn't get the better result. So, finally we have decided to stick with the Original code only.
- The detailed description of all the changes attempted by the group along with screenshots are placed below for understanding and justifying that why we have preferred not to changed the original code: -

  1. Value of k in K-means (Figure - 4) : - We have tried first by changing the value of k in k-means. The Value initially mentioned in the paper was .0001 and we tried by substituting the value with .001. However, we have observed that the prec@ values at


shreya@shreya-Lenovo-Z50-70: ~/ml/Tree_Extreme_Classifiers/Parabel
File Edit View Search Terminal Help

```
shreya@shreya-Lenovo-Z50-70:~/ml/Tree_Extreme_Classifiers/Parabel$ python3 eval.py
tree 0 predicting started
node 0
node 100
tree 0 predicting completed
tree 1 predicting started
node 0
node 100
tree 1 predicting completed
tree 2 predicting started
node 0
node 100
tree 2 predicting completed
prediction time: 1.65312 ms/point
model size: 0.0661819 GB
Total time taken is 45.467310 seconds
prec@1: 0.987 prec@3: 0.958 prec@5: 0.931
mprec@1: 4.831e-03 mprec@3: 3.650e-02 mprec@5: 1.079e-01
shreya@shreya-Lenovo-Z50-70:~/ml/Tree_Extreme_Classifiers/Parabel$
```

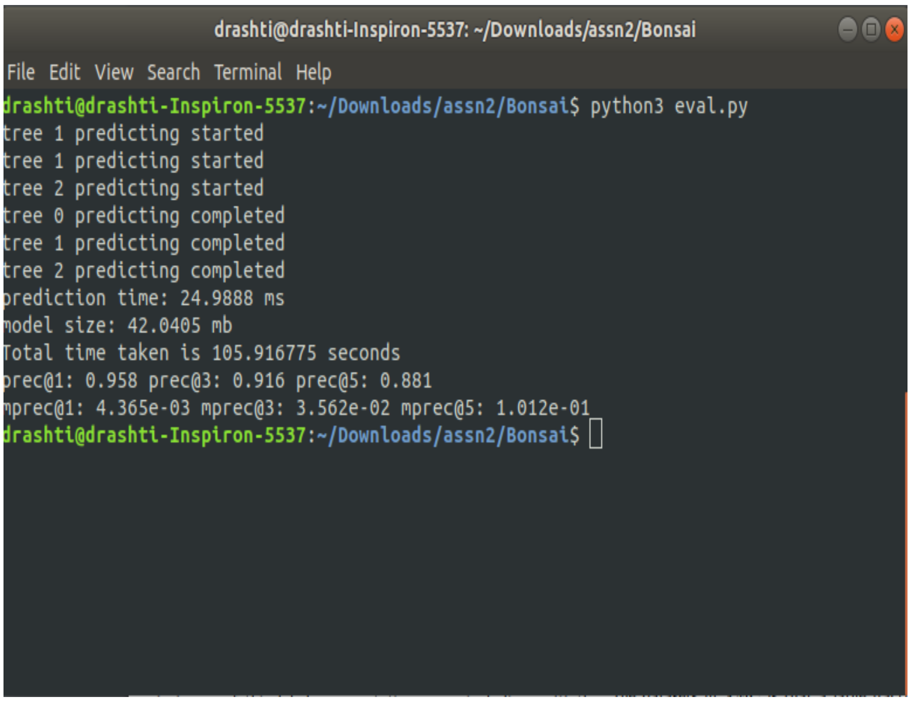
Figure 2: Screenshot depicting the result of Parabel Method


shreya@shreya-Lenovo-Z50-70: ~/ml/Tree_Extreme_Classifiers/FastXML
File Edit View Search Terminal Help

```
        node 1000
        node 0
        node 0
tree 18 prediction completed
tree 19 prediction started
        node 1000
        node 0
tree 29 prediction completed
        node 1000
        node 1000
tree 49 prediction completed
        node 1000
tree 38 prediction completed
tree 39 prediction started
tree 19 prediction completed
        node 0
        node 1000
tree 39 prediction completed
prediction time: 1.97063 ms/point
model size: 0.0358525 GB
Total time taken is 34.710873 seconds
prec@1: 0.872 prec@3: 0.774 prec@5: 0.679
mprec@1: 1.587e-03 mprec@3: 1.222e-02 mprec@5: 3.117e-02
shreya@shreya-Lenovo-Z50-70:~/ml/Tree_Extreme_Classifiers/FastXML$
```

Figure 3: Screenshot depicting the result of FastXML Method

1,3 and 5 labels were decreased.



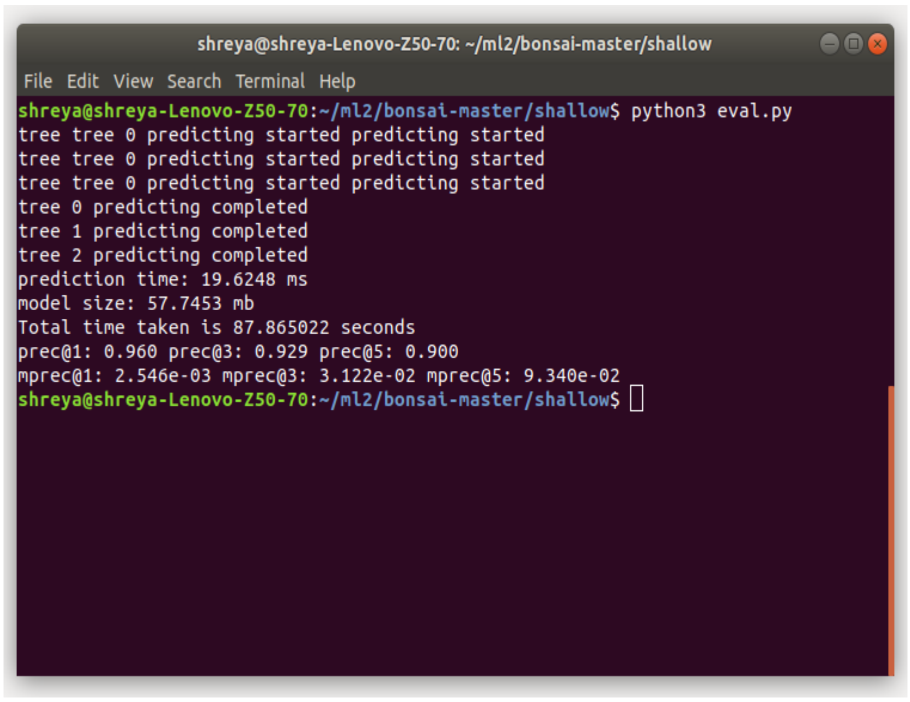
```
drashti@drashti-Inspiron-5537: ~/Downloads/assn2/Bonsai
File  Edit  View  Search  Terminal  Help
drashti@drashti-Inspiron-5537:~/Downloads/assn2/Bonsai$ python3 eval.py
tree 1 predicting started
tree 1 predicting started
tree 2 predicting started
tree 0 predicting completed
tree 1 predicting completed
tree 2 predicting completed
prediction time: 24.9888 ms
model size: 42.0405 mb
Total time taken is 105.916775 seconds
prec@1: 0.958 prec@3: 0.916 prec@5: 0.881
mprec@1: 4.365e-03 mprec@3: 3.562e-02 mprec@5: 1.012e-01
drashti@drashti-Inspiron-5537:~/Downloads/assn2/Bonsai$ 
```

Figure 4: Screenshot depicting the result of Training at k=.001

2. Value of m (Figure - 5): - We have tried training the bonsai method by increasing the number of level of the tree. By-default value of m was 2 and we have tried using m = 3. However, we have observed that value of prec@ has reduced drastically, with slight increase in model size as well as total time.

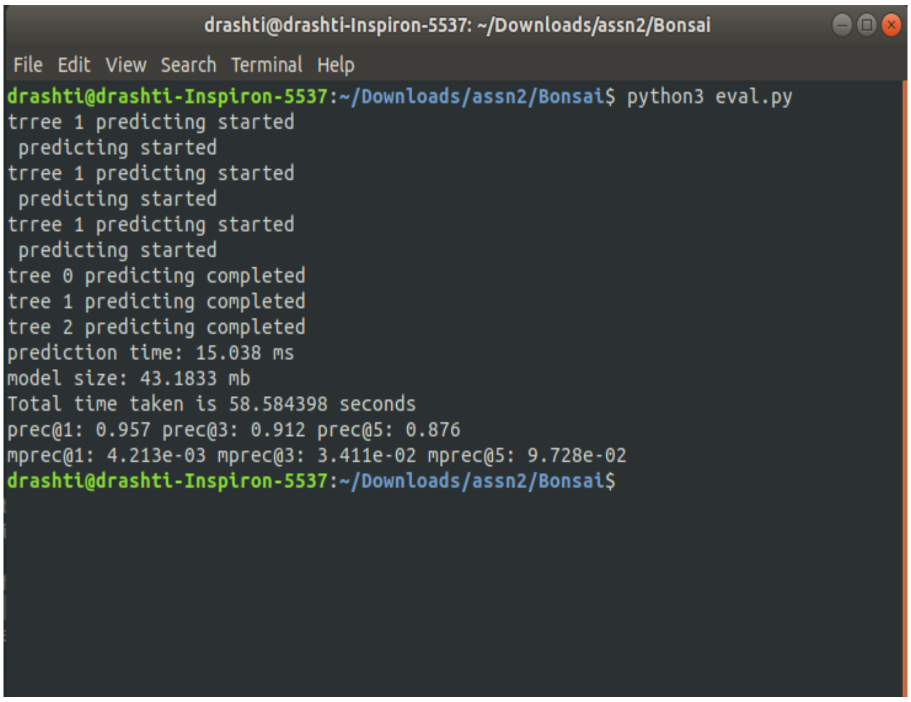
3. Value of w (Figure - 6): - We have also tried changing the value of w (number of children for each node). By-default value of w was 100 and we have tried by increasing the number of children per node and we have set w to 300. However, again increasing the number of children doesn't improved the method's performance.

4. Value of t (Figure - 7): - Finally, we have tried growing more number of trees by varying the value of parameter-t (number of trees to be grown). Initially the value was t=3 and we have tried growing more trees by increasing the value of t to 5. Again, we didn't get improved result.



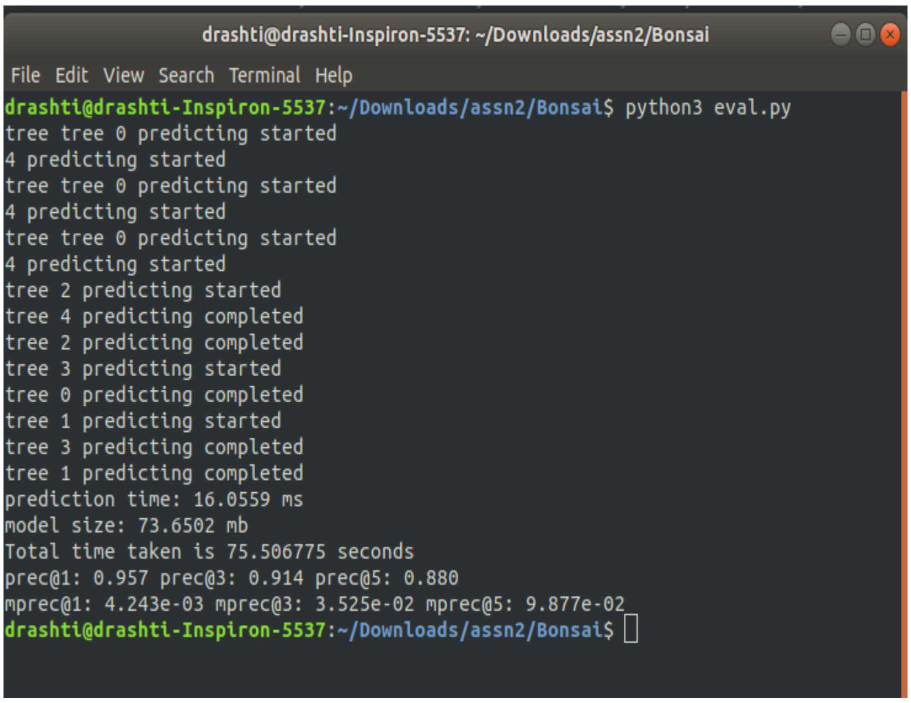
```
shreya@shreya-Lenovo-Z50-70:~/ml2/bonsai-master/shallow$ python3 eval.py
tree tree 0 predicting started predicting started
tree tree 0 predicting started predicting started
tree tree 0 predicting started predicting started
tree 0 predicting completed
tree 1 predicting completed
tree 2 predicting completed
prediction time: 19.6248 ms
model size: 57.7453 mb
Total time taken is 87.865022 seconds
prec@1: 0.960 prec@3: 0.929 prec@5: 0.900
mprec@1: 2.546e-03 mprec@3: 3.122e-02 mprec@5: 9.340e-02
shreya@shreya-Lenovo-Z50-70:~/ml2/bonsai-master/shallow$
```

Figure 5: Screenshot depicting the result of Training at m=3



```
drashti@drashti-Inspiron-5537:~/Downloads/assn2/Bonsai$ python3 eval.py
trree 1 predicting started
 predicting started
trree 1 predicting started
 predicting started
trree 1 predicting started
 predicting started
tree 0 predicting completed
tree 1 predicting completed
tree 2 predicting completed
prediction time: 15.038 ms
model size: 43.1833 mb
Total time taken is 58.584398 seconds
prec@1: 0.957 prec@3: 0.912 prec@5: 0.876
mprec@1: 4.213e-03 mprec@3: 3.411e-02 mprec@5: 9.728e-02
drashti@drashti-Inspiron-5537:~/Downloads/assn2/Bonsai$
```

Figure 6: Screenshot depicting the result of Training at w=300


drashti@drashti-Inspiron-5537: ~/Downloads/assn2/Bonsai
File Edit View Search Terminal Help

```
drashti@drashti-Inspiron-5537:~/Downloads/assn2/Bonsai$ python3 eval.py
tree tree 0 predicting started
4 predicting started
tree tree 0 predicting started
4 predicting started
tree tree 0 predicting started
4 predicting started
tree 2 predicting started
tree 4 predicting completed
tree 2 predicting completed
tree 3 predicting started
tree 0 predicting completed
tree 1 predicting started
tree 3 predicting completed
tree 1 predicting completed
prediction time: 16.0559 ms
model size: 73.6502 mb
Total time taken is 75.506775 seconds
prec@1: 0.957 prec@3: 0.914 prec@5: 0.880
mprec@1: 4.243e-03 mprec@3: 3.525e-02 mprec@5: 9.877e-02
drashti@drashti-Inspiron-5537:~/Downloads/assn2/Bonsai$
```

Figure 7: Screenshot depicting the result of Training at t=5

References

[1] S. Khandagale, H. Xiao, and R. Babbar, "Bonsai - diverse and shallow trees for extreme multi-label classification," in *ArXiv*, 2019.

[2] Y. Prabhu, A. Kag, S. Harsola, R. Agrawal, and M. Varma, "Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising," in *WWW*, 2018.

[3] *https://github.com/xmc-aalto/bonsai*.

[4] Y. Prabhu and M. Varma, "Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning," in *KDD*, 2014.