Distillation of a Random Forest to a Decision Tree

Brijesh patel
Registration Number : 2003331

Abstract*— Random Forest, which is a machine learning method is part of the supervised learning technique and is found in many different software packages. When applied to ML problems, it is used for both classification and regression tasks. In order to tackle a difficult problem and increase the model's performance, it is based on the concept of ensemble learning, which is the act of combining numerous classifiers. In this work, I used knowledge distillation on a random forest, which is a technique in which an ensemble of trees is combined to produce a single tree that has the same or better performance than the ensemble of trees that was used previously.

Keywords*— Random Forest, Decision Tree, Classification, Knowledge Distillation, Python, Pandas, Dtreeviz.

I. INTRODUCTION

Knowledge Distillation (KD) is a technique for shrinking models without sacrificing their performance. Transferring knowledge from a large pre-trained model, which is also known as the teacher model, to a smaller model is the goal of this approach (student model). In order to solve a binary classification problem, we will focus on minimizing a custom loss function instead of matching softening teacher logits and ground-truth labels.

Random forest classifier is a type of ensemble learning method in which a large number of randomized decision trees are formed and the findings from all of the trees are merged to get a final prediction for the forest as a whole. As a result of their inception in 2001, random forests and their various versions have become widely employed in a wide range of applications, including deep learning and even outlier identification. Its theoretical properties have also been thoroughly investigated, in addition to its practical applications [1].

Those sophisticated algorithms, such as random forests and GBDT, achieve excellent success in many aspects, but this high prediction performance comes at the expense of a significant reduction in interpretability. Considering the fact that, when compared to decision trees, random forests' bootstrap and voting procedure makes explaining the predictions considerably more difficult to accomplish. On the contrary, decision trees are well-known for having the finest interpretability of any machine learning algorithm, but having relatively poor performance in comparison to other algorithms. Furthermore, forest-based algorithms or even deep neural networks (DNN) typically demand significantly more storage than decision trees, which is unsuitable when the model is running on a mobile device with severe storage constraints (such as a cellular device). The tension between empirical soundness and interpretability in the context of flexible storage is a constant source of inspiration for scholars.

Breiman proposed the use of a tree ensemble to create more samples for the further development of the decision tree in 1996 to address these issues, which can be considered the first attempt to address this problem. This strategy uses soften labels given by fully trained DNN to create a more intelligible model in the form of soft decision trees, which has recently been presented as an alternative to traditional distillation methods. In contrast, because this method is based on soft decision trees and their backpropagation, it cannot be used to traditional decision trees. Aside from that, the interpretability of soft decision trees is significantly lower than that of classical decision trees [2].

II. LITERATURE REVIEW

When it comes to data analysis, classification is a typical technique that divides data points into multiple categories. The programme enables you to arrange data sets of all sizes and complexity, including complex and huge datasets, in addition to small and basic datasets.

Data quality can be improved by the use of algorithms that can be readily updated. As a result, supervised learning has become a common method for classifying data. One of the basic goals of classification is to establish a connection between a variable of interest and the necessary variables. In this section, I will discuss about two classification techniques Random Forest Algorithm and Decision Tree Algorithm.

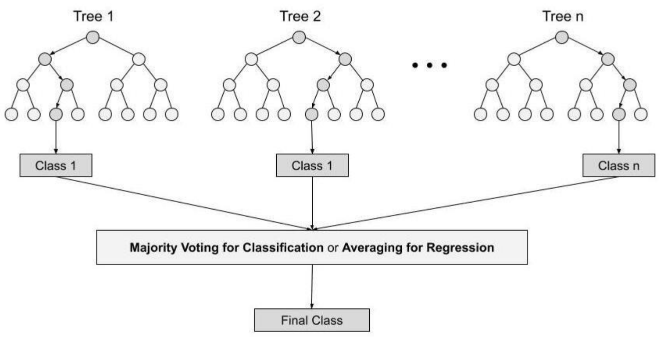
a. Random Forest Algorithm

In machine learning, a supervised learning technique known as a random forest is utilised for classification as well as regression. However, it is mostly employed in the solution of categorization problems. As we all know, a forest is composed of trees, and a forest with more trees is a more vigorous forest. Additionally, the random forest algorithm constructs decision trees from data samples, then obtains the forecast from each of them, and eventually selects the best solution through a process known as vote selection. In comparison to a single decision tree, an ensemble approach performs better since it minimizes over-fitting by aggregating the outcomes.

Regression and classification can both benefit from the Random Forest Algorithm's ability to handle data sets that include both continuous and categorical variables. It provides superior results when dealing with categorization challenges.

The steps involved in the random forest method are as follows**:**

- In the beginning, select a random sample say n from a pre-existing dataset.
- Each sample is given its own decision tree, which is created individually.
- There will be an output for each decision tree.
- Final results are taken into account for regression and classification using the Majority Voting or Averaging methods.



Tree 1
Tree 2
Tree n
Class 1
Class 1
Class n
Majority Voting for Classification or Averaging for Regression
Final Class

Figure-1: Random Forest Example [2]

Random Forests Have a Few Important Characteristics:

Diversity- The qualities, variables, and properties pertinent to a particular tree are taken into account in the creation of that tree.

Resistance to dimensionality's curse - The feature space is confined by the fact that no single tree can account for all of the attributes.

Parallelization- Each tree is built from scratch using a new set of data and attributes for each one. This means that we will be able to make complete advantage of the CPU when constructing random forests.

Train-Test split- We don't have to split our training and testing data in a random forest because the decision tree will never view more than 30% of the data.

Stability- A great degree of predictability is possible because the outcome is determined by majority voting/averaging.

Important Hyperparameters

Hyperparameters are used when dealing with random forests to either improve model performance and predictive ability or speed up the model.

Predictive power is increased by following hyper parameters:

n_estimators - A measure of how many trees the algorithm constructs before averaging the results.

max_features - Node splitting considered by random forest to be the maximum amount of features.

mini_sample_leaf– The minimal number of leaves required to separate an internal node is determined by this function.

The following hyperparameters accelerate the process:

n_jobs - It provides instructions to the engine on how many processing units it is permitted to employ. A single processor can only be used if the value is 1, and there is no limit on the number of processors if the value is -1.

random_state - This option affects the sample's randomness. It will always produce the same results because it is trained with the very same hyperparameters and training data.

oob_score – 'Out of the bag,' as the term suggests, means It is a random forest-based technique to cross-validation. One third of the sample is not utilized to train the data, but to evaluate its performance instead.

Advantages of Random Forest

- ➢ In addition to classification and regression tasks, Random Forest is also capable of accomplishing both.
- ➢ A huge dataset with a high dimensionality can be handled by this system.
- ➢ It improves the accuracy of the model and helps to avoid the problem of overfitting the data set.

Disadvantages of Random Forest

- ➢ Notwithstanding the way that random forests can be utilized for both characterization and regression tasks, they are not more qualified for regression based tasks.

b. Decision Tree

There are several ways to solve classification and regression problems using Decision Trees, although they are most commonly employed for Classification difficulties. Tree-based classifier, where internal nodes contain dataset properties, branching represents decision rules and each child node provides the classification conclusion.

One node in a decision tree is known as the Decision Node, and the other is known as the Leaf Node. Whereas Decision nodes are used to make decisions and have multiple branches, Child nodes are the outcome of such decisions and do not have any more branches.

In this situation, the results of the tests or judgments are based on the dataset's properties. It is helpful to represent data as graphs when calculating all possible outcomes of a problem or choice under defined criteria. With its root node at the top and extra branches sprouting out to form a tree-like shape, it is known as a decision tree.

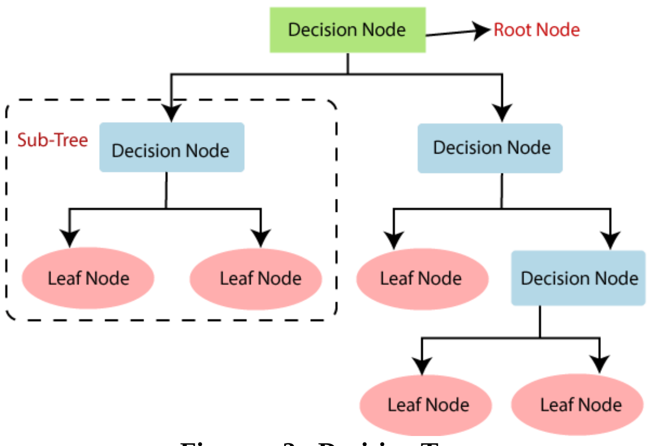


Figure – 2 : Decision Tree

Decision Trees Terminologies

- ➢ Root Node: As a result of this partition into two or more homogeneous groups, it represents the entire population or sample.
- ➢ Splitting: It is the process to divide a node into multiple subnodes.
- ➢ Decision Node: Sub-nodes dividing into additional sub-nodes have the decision node as their name suggests.
- ➢ Leaf / Terminal Node: Nodes where no further division is possible are referred to as Leaf or Terminal nodes.
- ➢ Pruning: When a decision node's sub-nodes are removed, it is said to be pruned. Splitting can be viewed as the polar opposite of that.
- ➢ Branch / Sub-Tree: There are many different types of branches and subdivisions of trees, each of which has its own distinct function.

- ➢ Parent and Child Node: Whenever a node has been split into sub-nodes, the parent node is referenced to as the node that has the sub-nodes as its child node.

In order to use a decision tree to forecast a dataset's class, we commence at the parent node and work our way down. This algorithm follows the branch and jumps to the next node depending on whether the value of the root property is larger or less than the value of the record (real dataset) attribute.

A comparison is made between a feature values of the next node and all the other sub-nodes in the tree before moving on to the next node's value is evaluated. After that, the method continues until it reaches the tree's leaf node. Understanding the entire process is easier if you use the following algorithm:

- ➢ Start with the root node S, which includes the entire dataset, and working your way up the tree from there.
- ➢ Determine the best attribute in the dataset sing the Attribute Selection Measure (ASM).
- ➢ Subdivide the S into subsets that contain probable values for the best qualities, and then combine the results.
- ➢ Follow the above procedures to create the best attribute decision tree node.
- ➢ In a recursive fashion, create more decision trees using the dataset subsets produced in step-3. The final node will be known as a "leaf node" if you have completed this process and are not able to label the remaining nodes any further.

Advantages

- ➢ A human being would apply the same reasoning process when making a real-world decision, thus it's easy to understand
- ➢ When it comes to making decisions, it can be incredibly helpful.
- ➢ When trying to come up with a solution to a problem, it is best to think about all possible outcomes.
- ➢ The need for data purification is significantly reduced compared to other ways.

Disadvantages

- ➢ A large number of layers make the decision tree complicated to navigate.
- ➢ An overfitting problem may exist in this model; however, the Random Forest technique can be used to overcome this problem.
- ➢ Because of the addition of more class labels, the decision tree's computational complexity may rise.

III. Experiment and Results

Dataset

The dataset was obtained from the sklearn.datasets website. I took into consideration three different datasets. The first is a dataset of Iris flowers. The iris dataset is a multi-class classification dataset that is both classic and simple to use. The following is a description of the dataset:

Classes	3
Sample per Class	50
Samples total	150
Dimensionality	4
Features	Real, positive

The dataset provides a response. The following are the interesting attributes of this dictionary-like object: 'data', which represents the data to be learned from, 'target', which represents the classification labels, 'target names', which represents the labels meaning, 'feature names', which represents the features, and 'DESCR', which represents the full dataset description.

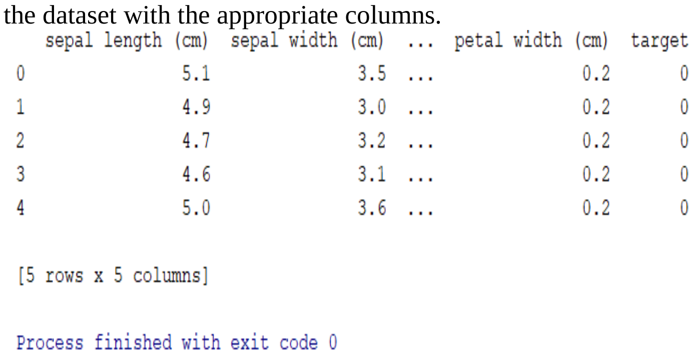
Secondly, a wine dataset was employed, and the characteristics of this dataset are stated in the following table:

Classes	3
Sample per Class	[59,71,48]
Samples total	178
Dimensionality	13
Features	Real, positive

This dataset also returns bunch of dictionary like objects such as data, target, feature_names, target_names, etc.
Third dataset used was of breast cancer winconsin dataset. It has 2 classes and 212 (M) and 357 (B) samples per class respectively. Total samples are 569 and dimensionality is 30. Features are described as real and positive.
It also returns data in a bunch like data dictionary with attributes such as data, feature names, target names, target, etc.

Methodology

At first, I will load the dataset and display it. Figure shows the dataset with the appropriate columns.



```
   sepal length (cm)  sepal width (cm)  ...  petal width (cm)  target
0                5.1               3.5  ...               0.2       0
1                4.9               3.0  ...               0.2       0
2                4.7               3.2  ...               0.2       0
3                4.6               3.1  ...               0.2       0
4                5.0               3.6  ...               0.2       0

[5 rows x 5 columns]

Process finished with exit code 0
```

Then I'm going to divide this dataset into two parts: a training set (which will contain a random 70% of the data) and a test set (the other 15 percent of the data).

((105,5), (45,5), 0.7)

This will be followed by training a Random Forest Classifier to find a solution for this problem. My main focus will not be on model performance (i.e., I will not attempt to make it generalizable) because I am more interested in replicating the same behaviour (whether good or bad as it now exists) using a Decision Tree proxy instead.

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        18
           1       1.00      0.88      0.94        17
           2       0.83      1.00      0.91        10

    accuracy                           0.96        45
   macro avg       0.94      0.96      0.95        45
weighted avg       0.96      0.96      0.96        45


Process finished with exit code 0
```

Now I'm going to retrain a single decision tree that will approach the Random Forest as closely as possible
The steps would be

- ➢ Practice with a vast, complicated ensemble (the Random Forest model above).
- ➢ All of the data (including the test set) should be compiled, and the predictions should be added on top of it (even for the test set).
- ➢ Overfitting a single Decision Tree across all of the data while training on the predictions of the Random Forests is an example of this.

As the fact is, Decision Tree has the potential to overfit the training data perfectly.

```
    sepal length (cm)  sepal width (cm)  ...  target  relabel
71                6.1               2.8  ...       1        1
52                6.9               3.1  ...       1        1
6                 4.6               3.4  ...       0        0
3                 4.6               3.1  ...       0        0
33                5.5               4.2  ...       0        0

[5 rows x 6 columns]

Process finished with exit code 0
```

Overfitting the training data with the decision tree is desirable, as in this study I am looking for a single condensed tree that behaves identically to that of the original Random Forest. In addition, I am looking for the Decision Tree to behave exactly the same way as the Random Forest when it comes to the testing data. That is why I am training on the entire dataset in this study.

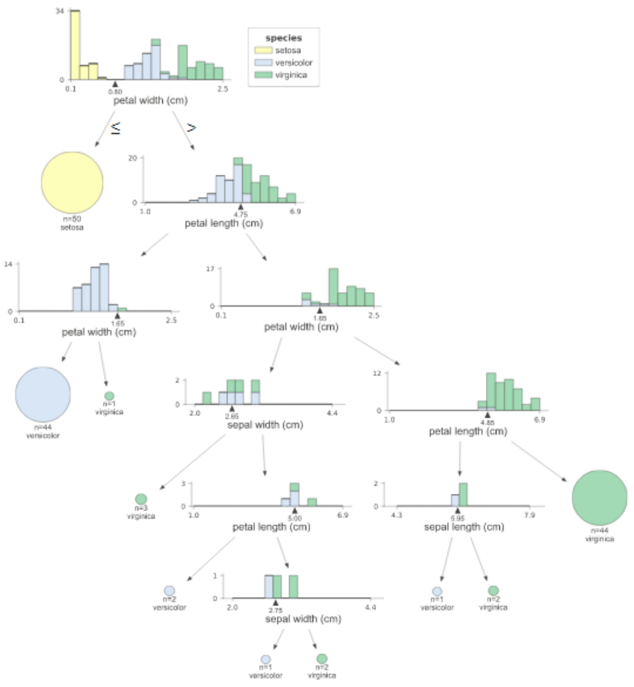
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        34
           1       1.00      1.00      1.00        32
           2       1.00      1.00      1.00        39

    accuracy                           1.00       105
   macro avg       1.00      1.00      1.00       105
weighted avg       1.00      1.00      1.00       105
```

The precision, recall, f1-score and support value for real test data is shown in the figure below.

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       0.94      0.94      0.94        18
           2       0.91      0.91      0.91        11

    accuracy                           0.96        45
   macro avg       0.95      0.95      0.95        45
weighted avg       0.96      0.96      0.96        45


Process finished with exit code 0
```

We now have a Decision Tree that behaves precisely the same as the Random Forest, proving that this method worked.

Finally, I have constructed a visualization of a tree that I obtained by utilizing the dtreeviz module that is accessible in the Python programming language.



Similar approach was applied to the other 2 datasets namely wine and breast cancer. Precision, recall and f1-score for wine dataset was found to be 0.98, 0.98 and 0.98 respectively.
For breast cancer dataset, precision, recall and f1-score was found out to be 0.95, 0.95 and 0.95 respectively.

Conclusion

First of all, I would like to thanks my professor for guiding me through this assignment. In this study, I have applied knowledge distillation on random forest to convert it into a decision tree with similar accuracy. For experiment purpose I have used 3 datasets, namely Iris, Wine and Breast Cancer. Accuracy found out to be 0.96, 0.98 and 0.95 respectively and it replicated the behavior of the random forest.

References

[1] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015

[2] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In CVPR, pages 4133–4141, 2017

[3] https://www.javatpoint.com/machine-learning-random-forest-algorithm

[4] https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_classification_algorithms_random_forest.htm

[5] https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/.

[6] https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

[7] https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html.