Practical Work 2: Random Forest, an Ensemble Learning Classifier implementing CART

Reynoso Aguirre, Pablo Eliseo
pablo.eliseo.reynoso@est.fib.upc.edu

December 14, 2017

1 Context - Theoretical Aspects

The aim of the practical is focused on the implementation and validation of an ensemble classifier that combines different small predictors, for this particular work we considered *Random Forest* algorithm with a *Classification Regression Tree* approach for the small predictors, this type of *Decision Trees* consider a *Gini-Impurity* measure to find the most suitable attribute-value combination for each tree's node in terms of how mixed will be each split given by a discrimination of a particular attribute-value combination, *CART* intends to find decision trees with a less or non-mixtured binary splits representing a natural classification. In short we are going to mention some of the *Random Forest CART* algorithm properties according to [3]:

- *Ensemble Learning* technique.
- Combines simple *Decision Trees* in parallel at learning & predicting.
- *Decision Trees* predictors based on *CART Gini-Index.*
- Induces every tree learning by a dataset *Bootstrap* sample.
- Builds every tree by a binary split evaluation of random selection of sample features.
- Finds class popoulation balancing error in unbalanced datasets.
- Runs efficiently on large data bases with thousands of input variables.

According to [2] [1] The main advantage of the *Random Forest* relies in the underlaying bagging approach of the ensemble of decision trees, which reduces the variance resulting as a robustness to over-fitting. Moreover, another relevant aspect is that every tree structure is induced with a random subset of features, this lead to a combination of uncorrelated simple classifiers by encouraging trees to be different, being straight forward with the motivation of a natural combination of classifiers in *Ensemble Learning.* A possible disadvantage is that at inducing every tree, we have to consider different combinations of binary splits by different attributes values measuring their impurity in terms of split class mixtures and this may make the model complex when variables are numeric continous or tend to have a wide range of values.

1.1 Random Forest Pseudocode

The following section describes the pseudocode of Random Forest from a high-level perspective, for descriptive purposes according to [2] [3]:

Algorithm 1 Random Forest - CART

```
procedure RandomForest(S=[X,y],NT,F,tree_max_depth,node_min_samples,train_size,labels)
    $H \leftarrow \emptyset$
    for $i \in 1, .., NT$ do:
        $S(i) \leftarrow BootstrapSmple(S)$
        $h_i \leftarrow CART(S(i), F)$
        $H \leftarrow H \cup h_i$
    return $H$
procedure CART(S,F)
    $tree \leftarrow \emptyset$
    while Depth(tree) <= max_depth do:
        $features \leftarrow RandomSubsetFeatures(F)$
        $best\_gini \leftarrow 1$
        $best\_feature \leftarrow -1$
        $best\_value \leftarrow -1$
        $best\_split \leftarrow \emptyset$
        for $feature \in features$ do:
            for $s \in S$ do:
                $split \leftarrow BinarySplit(feature, S, S_{feature})$
                $gini \leftarrow GiniIndex(split, labels)$
                if gini < best_gini then:
                    $best\_gini \leftarrow gini$
                    $best\_feature \leftarrow feature$
                    $best\_value \leftarrow S_{feature}$
                    $best\_split \leftarrow split$
        $node \leftarrow [best\_gini, best\_feature, best\_value, best\_split]$
        ExpandTree($node$)
    return $tree$
```

2 Application - Technical Aspects

The assignment is break in the following computational and technical aspects:

1. Development of *Random Forest* algorithm with a *Decision Tree CART* approach in *Python.*
    - Development of *.CSV* data pre-processor class including: missing values, mix attributes type, irregular data shape.
    - Development of a *.TXT* report generator module, comprising: pre-processing times, learning/classification times, hyper-parameters(*Number of Trees, Number of Features, Tree Depth, Node Min Instances, Training Size*), mapping dictionary of categorical attributes, and features relevance frequencies in random forest performance for an specific data.
2. Implementation of *Random Forest* learning/validation of a set of datasets from different scopes, obtaining classification accuracy scores in a *5-Fold Cross Validation* framework.

- Simulation of learning/validation done by the combination of the following hyper-parameters values *Number of Trees* = {50,100}, *Number of Features* = $\{1,3,int(log_2(M)+1),sqrt(M)\}$.
- Simulation of learning/validation classification task on the following *UCI/Kaggle* databases: {*Contact Lenses*, *Zoo Animals*, *SMS Spam-Ham*, *Mushrooms Edible-Poisoned*, *Taekwondo Belts*}

3 Datasets

3.1 Contact Lenses Dataset

Contact Lenses Dataset [6], this dataset was originally obtained from the slides of the *Supervised and Experience Learning* course of the *Master in Artificial Intelligence*. It comprises data related to human-eye attributes that we asume as factors in order to make an inference of an automated discrimination model of lenses. However, if there is an available amount of considerable cases to induce the model, we can always obtain a good approximation to a human-eye lenses classifier.

3.1.1 Considerations

For this particular dataset, we had to perform a preprocessing doing a mapping of the categorical variables to ordinalnumerical ones.

In short, we are going to describe in a breif way data specifications:

- 24 instances
- 5 attributes (All Categorial)
  1. Age
  2. Visual Deficiency
  3. Astigmatism
  4. Tear Production
  5. Recommended Lenses

3.2 Mushrooms Poisoned-Edible Dataset

Mushrooms Poisoned-Edible Dataset [7], this dataset was provided by *UCI-Kaggle* colaborators. The dataset motivation is focused on classifying the poisoned and edible mushrooms according to different mushroom internal quemical-color features from different types of mushrooms. This dataset includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the *Agaricus Lepiota Family Mushrooms* drawn from *The Audubon Society Field Guide to North American Mushrooms*.

3.2.1 Considerations

For this particular dataset, we had to change the order of the first column to related to Class to the end for dataset easy management at the classification task. Besides, we had to perform a preprocessing imputing missing values by pandas and consequently perform a mapping of the categorical variables to ordinalnumerical ones.
The following especifications describe briefly the dataset:

- 8124 instances
- 23 attributes (All Categorical)
  1. Cap Shape
  2. Cap Surface
  3. Cap Color
  4. Bruises
  5. Odor
  6. Gill Attachment
  7. Gill Spacing
  8. Gill Size
  9. Gill Color

10. Stalk Shape
11. Stalk Root
12. Stalk Surface above ring
13. Stalk Surface below ring
14. Stalk Color above ring
15. Stalk Color below ring
16. Veil Type
17. Veil Color
18. Ring Number
19. Ring Type
20. Spore Print Color
21. Population
22. Habitat
23. Class

3.3 Zoo Animals Dataset

Zoo Animals Dataset [4], this dataset was originally donated from *Richard S. Forsyth* and distributed by *UCI-Kaggle* colaborators. It comprises data related to all the animals from a zoo, where each feature in data contain every animal specifications. The class types of animals are: *Mammal, Bird, Reptile, Fish, Amphibian, Bug and Invertebrate.* The main aim of this dataset, is centered in predicting animals classification types, based upon the variables. In short, we are going to describe in a breif way data specifications:

- 101 instances
- 16 attributes

  1. Name (Categorical)
  2. Hair (Boolean)
  3. Feathers (Boolean)
  4. Eggs (Boolean)
  5. Milk (Boolean)
  6. Airbone (Boolean)
  7. Aquatic (Boolean)
  8. Predator (Boolean)
  9. Toothed (Boolean)
  10. Backbone (Boolean)
  11. Breathes (Boolean)
  12. Venomous (Boolean)
  13. Fins (Boolean)
  14. Legs (Integer)
  15. Tail (Boolean)
  16. Domestic (Boolean)
  17. Catsize (Boolean)
  18. Class Type (Integer)

3.4 SMS Spam-Ham Dataset

SMS Spam-Ham Dataset [8], this dataset was provided by *UCI-Kaggle* colaborators. *The SMS Spam Collection* is a set of English*SMS* messages tagged as spam/ham. These messages have been collected for *SMS Spam Research.* The main purpose of the data usage is focused on building an accurate prediction model that can be able to classify wheter texts are flagged as spam or not.

3.4.1 Considerations

For this particular dataset, we had to compose a *CORPUS* of text from all the instances(messages) in the data, after that extract the meaningful and dsicriminant features applying *TFIDF* criteria transforming the messages dataset into a matrix, which comprises the 10 most significant words that appeared maximum in 35% of the document i.e. (non trivial terms, neither stopwords and considering $n = \{12\}grams$). Each intersection of the $i - th$ row and $j - th$ column of the matrix, is a continouos value that stands for the relvance of term j in the message i considering information from the message itself and from the knowledge base, which is the *CORPUS*.

The following especifications describe briefly the dataset:

- 5574 instances
- 10 attributes (All Continouos)

1. W1: *call*
2. W2: $free$
3. W3: *get*
4. W4: *go*
5. W5: *gt*
6. W6: *know*
7. W7: *lt*
8. W8: $\{lt, gt\}$
9. W9: *ok*
10. W10: *ur*
11. Class

3.5 Taekwondo Techniques Classification Dataset

Taekwondo Techniques Classification Dataset [5], this dataset was originally donated from *Ali Ghafour* and distributed by *Kaggle* colaborators. It comprises data related to information obtained from an impact sensor within a taekwondo chest protector. In this experiment, participants were asked to perform various taekwondo techniques (*RoundhouseRound Kick, Back Kick, Cut Kick & Punch*) on this chest protector for analysis of sensor readings. The main concerns about this experiment are centered in determining impact intesity according to experiecne level. Furthermore, the classificatio among different types of impacts.

3.5.1 Considerations

For this particular dataset, we had to reshape the dataset that have sensor measurments in order to merge it with the dataset that have relevant information from the participants of the experiment. After that, we changed the order of the column belt to to the end for dataset easy management at the classification task of taekwondo belts. Finally, we performed a mapping of the categorical variables to ordinalnumerical ones.

In short, we are going to describe in a breif way data specifications:

- 344728 instances
- 9 attributes (Mixed)
  1. ID (Categorical)
  2. Sex (Categorical)
  3. Age (Integer)
  4. Weight (Integer)
  5. Experience
  6. Technique (Categorical)
  7. Trial (Integer)
  8. Measuring (Integer)
  9. Belt (Categorical)

4 Experimental Results

4.1 Running Python Code

In order to execute the software you have to go to the file __main__.py this file has the main method that implement all the tools of *Pre-Processing, Model Validation, Random Forest* and *CART* classes. As we can observe below we have an individual procedure for each dataset, two of them require an special pre-processing e.g. (*Taekwondo, SMS-Spam-Messages*). For each loop we evaluate a different dataset, which has been trough a csv processing and data cleansing and shaping. After that we run algorithm learning analysis by considering different combinations of NT & F and other parameters that are fixed per every dataset CV analysis with different settings for different datasets. We output the information of all the accuracy metrics for every dataset different configuration of parameters, comprising preprocessing time and learning time (training & validation). The learning results are stored in a folder named: *Learning_ Results* that exist in the root folder of the project.

```
import numpy as np;
import math;
import time;

import DataPreprocessing;
import EnsembleClassifiers;
import ModelValidation;

DP = DataPreprocessing.DataPreprocessing();
EC = EnsembleClassifiers.EnsembleClassifiers();
MV = ModelValidation.ModelValidation();

#::::Special Cases Pre-processing::::
#DP.merge_taekwondo_datasets();
#DP.preprocess_sms_dataset();



def dataset_learning(dataset,output_file,dataset_name,preprocess_time,NT,F,parameters):

    txtfile = open('./Learning_Results/'+output_file+'.txt','wb');
    txtfile.write("\n::::::::::::::::::::::::::::");
    txtfile.write("\nRandom Forest Classification");
    txtfile.write("\n\n********************************");
    txtfile.write("\nDataset Name: "+dataset_name);
    txtfile.write("\nPre-processing Time: "+str(preprocess_time)+" s");
    txtfile.write("\nLearning Configurations: ");

    for j in range(len(NT)):

        print("range_j:: "+str(j));

        for k in range(len(F)):

            print("range_k:: " + str(k));

            learning_t0 = time.time();
            scores = MV.evaluate_algorithm(dataset,EC.random_forest,parameters[0],parameters[1],parameter
                                         parameters[3],parameters[4],NT[j],F[k]);
            learning_tf = time.time();


            txtfile.write("\n\n------------------------------");
            txtfile.write("\nNumber of Trees: " + str(NT[j]));
            txtfile.write("\nNumber of Features: " + str(F[k]));
            txtfile.write("\nTree Max Depth: " + str(tree_max_depth));
            txtfile.write("\nNode Min Elements: " + str(node_min_elements));
            txtfile.write("\nDataset Usage Percent: " + str(train_size*100));
            txtfile.write("\nNumber of Folds: " + str(k_folds));
            txtfile.write("\nAccuracy Fold-Scores: ");
            txtfile.write(','.join(map(str, scores)));
            txtfile.write("\nOverall Accuracy: "+str(sum(scores)/float(len(scores))));
            txtfile.write("\nLearning Time: " + str(learning_tf - learning_t0) + " s");
            txtfile.write("\nRandom Forest features relevance: ");
            txtfile.write(str(EC.trees_features));

     txtfile.close();



 #::::Data - Preprocessing::::
 datasets_names = ["Contact Lenses","Zoo", "SMS Spam", "Mushrooms", "Taekwondo"];
 file_names = ["contact_lenses_learning","zoo_learning","sms_spam_learning","mushrooms_learning","taekwor
 regression_task = [False, False, False, False, False];


 #::::Ensemble - Classifiers::::
 NT = [50,100];
 tree_max_depth = 10;
 node_min_elements = 1;
 train_size = 0.2;
 regression_task1 = True;


 #::::Model - Validation::::
 k_folds = 5;

 #::::Dataset Learning Analysis::::
 for d in range(len(datasets_names)):

     preprocess_t0 = time.time();
     dataset = DP.run_preprocessing(d);
     preprocess_tf = time.time();

     F = [1,3,int(math.log(DP.n_features,2)+1),int(np.sqrt(DP.n_features))];
     dataset_cat_dicitonaries = DP.features_reference_dictionary;

     parameters = [k_folds,regression_task[d],tree_max_depth,node_min_elements,train_size];
     dataset_learning(dataset,file_names[d],datasets_names[d],(preprocess_tf-preprocess_t0),NT,F,paramete
```

4.2 Contact Lenses Dataset

4.2.1 Random Forest - Features Relevance

Number of Trees	**Number of Features**	**5-CV** $\bar{Acc}(\%)$
50	1	63.0
	3	62.0
	$int(log_2(5)+1)$	80.0
	$\sqrt(5)$	63.0
100	1	50.0
	3	62.0
	$int(log_2(5)+1)$	80.0
	$\sqrt(5)$	63.0

Table 1: 5 Fold Cross Validation Accuracy Overall score from 100% samples for all the different configurations of hyperparameters in *Contact Lenses Dataset*.

Number of Trees	**Number of Features**	**Features Frecuency**
50	1	$tear\_production : 50$
	3	$age : 50$, $astigmatism : 50$, $tear\_production : 50$
	$int(log_2(5)+1)$	$age : 50$, $astigmatism : 50$, $tear\_production : 50$
	$\sqrt(5)$	$age : 50$, $tear\_production : 50$
100	1	$3 : 100$
	3	$age : 100$, $astigmatism : 100$, $tear\_production : 100$
	$int(log_2(5)+1)$	$age : 100$, $astigmatism : 100$, $tear\_production : 100$
	$\sqrt(5)$	$age : 100$, $tear\_production : 100$

Table 2: Features Relevance Frequency from 100% samples in all the different configurations of hyperparameters in Random Forest.

4.3 Mushrooms Poisoned-Edible Dataset

4.3.1 Random Forest - Features Relevance

Number of Trees	**Number of Features**	**5-CV** $\bar{Acc}(\%)$
50	1	69.0301553619
	3	84.6874194771
	$int(log_2(23)+1)$	96.6887836302
	$\sqrt(23)$	96.7257824934
100	1	69.0299658962
	3	84.6874194771
	$int(log_2(23)+1)$	96.6887836302
	$\sqrt(23)$	96.7257824934

Table 3: 5 Fold Cross Validation Accuracy Overall score from 25% samples for all the different configurations of hyperparameters in *Mushrooms Poisoned-Edible Dataset*.

Number of Trees	**Number of Features**	**Features Frecuency**
50	1	$habitat: 50$
	3	$cap\_surface: 50, population: 50, habitat: 50$
	$int(log_2(23)+1)$	$cap\_surface: 50, ring\_type: 50, population: 50, habitat: 50$
	$\sqrt(23)$	$cap\_surface: 50, ring\_type: 50, population: 50, habitat: 50$
100	1	$habitat: 100$
	3	$cap\_surface: 100, population: 100, habitat: 100$
	$int(log_2(23)+1)$	$cap\_surface: 100, ring\_type: 100, population: 100, habitat: 100$
	$\sqrt(23$	$cap\_surface: 100, ring\_type: 100, population: 100, habitat: 100$

Table 4: Features Relevance Frequency from 25% samples in all the different configurations of hyperparameters in Random Forest.

4.4 Zoo Animals Dataset

4.4.1 Random Forest - Features Relevance

Number of Trees	**Number of Features**	**5-CV $\bar{Acc}(\%)$**
50	1	41.6666666667
	3	45.6666666667
	$int(log_2(16)+1)$	68.380952381
	$\sqrt(16)$	57.4761904762
100	1	42.5238095238
	3	45.6666666667
	$int(log_2(16)+1)$	68.380952381
	$\sqrt(16)$	57.4761904762

Table 5: 5 Fold Cross Validation Accuracy Overall score from 100% samples for all the different configurations of hyperparameters in *Zoo Animals Dataset* .

Number of Trees	**Number of Features**	**Features Frecuency**
50	1	$catsize : 50$
	3	$catsize : 50, name : 50, hair : 50$
	$int(log_2(16)+1)$	$name : 50, catsize : 50, fins : 50, tail : 50, hair : 50$
	$\sqrt(16)$	$catsize : 50, hair : 50, tail : 50, name : 50$
100	1	$catsize : 100$
	3	$catsize : 100, name : 100, hair : 100$
	$int(log_2(16)+1)$	$name : 100, catsize : 100, fins : 100, tail : 100, hair : 100$
	$\sqrt(16)$	$catsize : 100, hair : 100, tail : 100, name : 100$

Table 6: Features Relevance Frequency from 100% samples in all the different configurations of hyperparameters in Random Forest.

4.5 SMS Spam-Ham Dataset

4.5.1 Random Forest - Features Relevance

Number of Trees	**Number of Features**	**5-CV $\bar{Acc}(\%)$**
50	1	87.3295762855
	3	88.1728510357
	$int(log_2(10)+1)$	89.0702594778
	$\sqrt(10)$	88.8192833163
100	1	87.311542456
	3	88.1728510357
	$int(log_2(10)+1)$	89.0702594778
	$\sqrt(10)$	88.8192833163

Table 7: 5 Fold Cross Validation Accuracy Overall score from 25% samples for all the different configurations of hyperparameters in *SMS Spam-Ham Dataset*.

Number of Trees	**Number of Features**	**Features Frecuency**
50	1	$ur: 50$
	3	$ok: 50$, $ur: 50$, $call: 50$
	$int(log_2(10)+1)$	$call: 50$, $ur: 50$, $ok: 50$, *{lt, gt}* $: 50$
	$\sqrt(10)$	$ok: 50$, $ur: 50$, $call: 50$
100	1	$ur: 100$
	3	$ok: 100$, $ur: 100$, $call: 100$
	$int(log_2(10)+1)$	$call: 100$, $ur: 100$, $ok: 100$, *{lt, gt}* $: 100$
	$\sqrt(10)$	$ok: 100$, $ur: 100$, $call: 100$

Table 8: Features Relevance Frequency from 100% samples in all the different configurations of hyperparameters in Random Forest.

4.6 Taekwondo Techniques Classification Dataset

4.6.1 Random Forest - Features Relevance

Number of Trees	**Number of Features**	**5-CV $\bar{Acc}(\%)$**
50	1	44.3312887113
	3	100.0
	$int(log_2(9) + 1)$	100.0
	$\sqrt(9)$	100.0
100	1	44.6113286713
	3	100.0
	$int(log_2(9) + 1)$	100.0
	$\sqrt(9)$	100.0

Table 9: 5 Fold Cross Validation Accuracy Overall score from 0.72% samples for all the different configurations of hyperparameters in *Taekwondo Techniques Classification Dataset* .

Number of Trees	**Number of Features**	**Features Frecuency**
50	1	$measuring : 50$
	3	$id : 50, measuring : 50$
	$int(log_2(9) + 1)$	$id : 50, measuring : 50$
	$\sqrt(9)$	$id : 50, measuring : 50$
100	1	$measuring : 100$
	3	$id : 100, measuring : 100$
	$int(log_2(9) + 1)$	$id : 100, measuring : 100$
	$\sqrt(9)$	$id : 100, measuring : 100$

Table 10: Features Relevance Frequency rom 0.72% samples in all the different configurations of hyperparameters in Random Forest.

5 Conclusions

In particular, after a carefully analysis of 6 different types of databases from different scopes, we can observe that in some of the cases the overall accuracy from a 5-CV metric procedure can lead to overfitting when there exist one or more variables highly correlated with the output, in other cases the generalization accuracy seem to be a feasible taks, but we can observe that in other scenarios where we have, higher number of samples and columns, and perhaps attributes are not discriminant enough, something that can be solve by removing features with a *Chi Squared* anlaysis, *PCA* or generating new attributes and replace it from ancestors in the data preprocessing manually. Furthermore, we can also observe that the number of trees does not improve generalization accracy significantly, but the number of features does it in a explicit way.

References

[1] Matthew N. Bernstein. Random forests. Class Presentation 1, University of Wisconsin - Madison, March 2017.

[2] Leo Breiman. Random forests. *Journal Machine Learning*, 45(1):5–32, October 2001.

[3] Eric Debreuve. An introduction to random forests. Technical Report 1, University Nice Sophia Antipolis, Toulouse, France, July 2013.

[4] UCI Machine Learning Forsyth Richard. Zoo animal classification dataset, May 1990.

[5] Ali Ghafour. Taekwondo techniques classification, 2016.

[6] Miquel Sanchez iMarre. Contact lenses dataset. Supervised and Experienced Learning, 2014.

[7] UCI Machine Learning. Mushroom classification dataset, April 1987.

[8] Kaggle UCI Machine Learning. Sms spam collection dataset, 2011.