TREES ENSEMBLE

Ensemble methods involve combining the predictions of multiple models to obtain a general model that combines the results of the intermediate models using a voting method. In the case of parallel ensemble methods like bagging, the models are trained on different data samples, allowing for variations in the training data. This enables the models to focus on specific patterns, enhancing their ability to capture different aspects of the data.

Bagging consists in creating multiple sub-samples of data randomly with replacement, meaning that in each sub-sample, certain observations may appear multiple times while others may not appear at all. The goal is to generate a substantial number of sub-samples to mitigate the risk of certain patterns not being adequately represented. Then, a learning model is applied to each data sub-sample, and the results of these models are combined using a voting approach (Fig. 6).

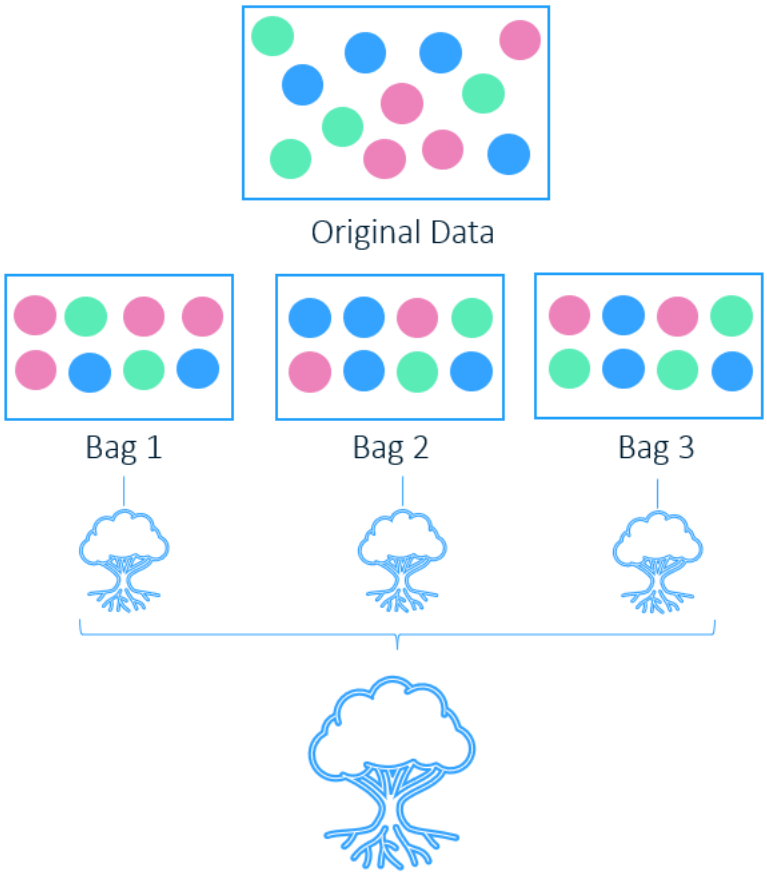

Fig. 6., Bagging for Trees Ensemble.

The models built on SAS Viya produce non-deterministic results, meaning that running a model twice will yield slightly different results. This is due to the calculations being performed in a distributed environment and the presence of randomness in the algorithms. This feature brings the models even closer to the Random Forest model, as each tree will retain slightly different variables, thereby reducing the correlation between the trees and providing an extra layer of randomness.

Before constructing the different bagging sets, we perform discretization of continuous variables. Since decision trees can potentially make an infinite number of splits for continuous variables, we aim to reduce the number of possible splits by discretizing these variables. This is done in order to proceed with a final selection of decision rules based on the frequency of appearance of these rules. It is not necessary to group nominal variables since their cardinality is much lower.

For discretization, we use the same grouping technique as the scorecard/logistic regression method, which involves using a decision tree to group values into intervals. Again, using the quantile method, each continuous variable is divided into a maximum of 10 intervals. This approach allows us to maintain a satisfactory level of precision, although it may be slightly lower than what we could achieve without discretization. The rule extraction process involves making certain trade-offs.

The Interactive Grouping node enables the creation of two new variables for each included variable. These new variables are automatically added to the dataset: `GRP_` + variable name contains the group number for the associated variable and `WOE_` + variable name contain the calculated Weight of Evidence.

To replace the continuous variables in our dataset with their assigned groups from discretization, we can use the program presented in Code 7, which allows replacing only the continuous variables using SAS arrays.

```
data '/home/sasdemo/casuser/bagging_tables/table_with_bins' (drop=i GRP_: WOE_:);
    set &dm_data;
    if _n_=1 then set &dm_data;
    array varorigin{*} %dm_interval_input;
    array varbin{*} $ GRP_:;
    do i=1 to dim(varorigin);
    if varorigin{i} then varorigin{i}=varbin{i};
    varorigin{i}=put(varorigin{i}, 5.);
    end;
run;
```

Code 7., Replace continuous values by their associated group.

The '`varorigin`' array selects the continuous variables from the dataset, while the '`varbin`' array retains the variables containing the groups for the discretized variables. Using their column positions in the arrays, we replace the original variables with their discretized forms to use them as inputs in the models. The data discretization step is performed in the yellow portion of the pipeline (Fig. 7.).

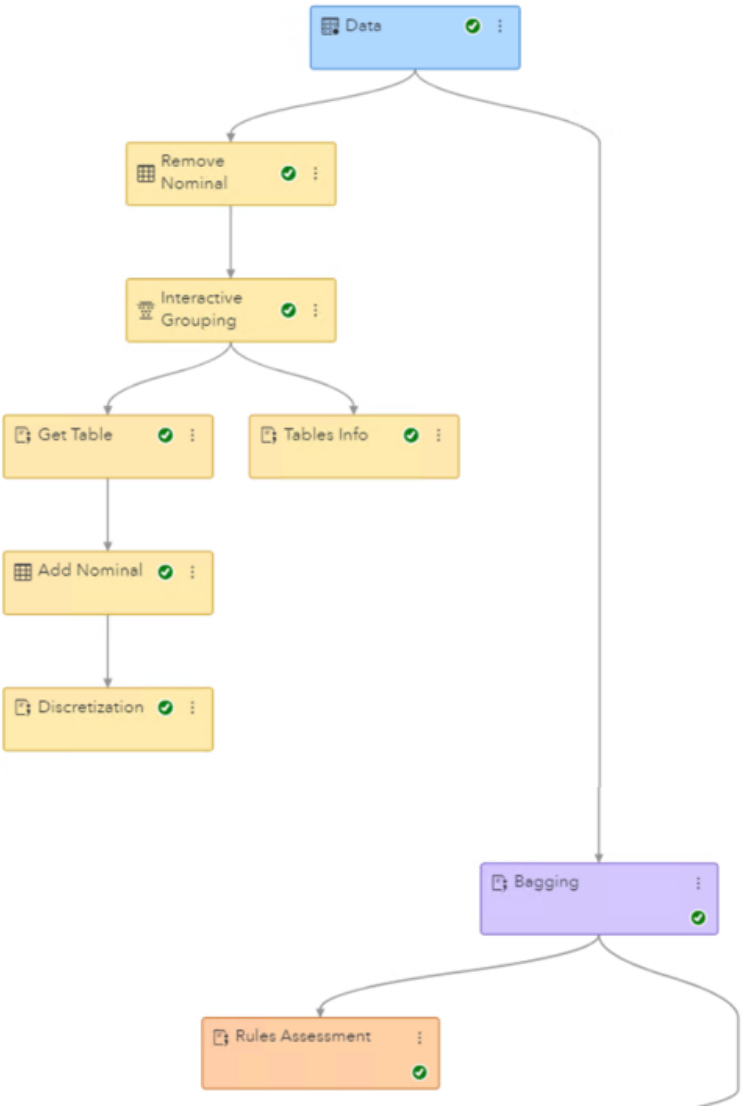

Fig. 7., Pipeline for Ensemble Trees.

The data table (Table 10.) no longer contains continuous variables, only nominal variables and numbers representing the group to which the original numerical value belongs. It is also possible to retrieve the value corresponding to the number of each variable using Table 11.

fraud_ind	ucm_pos	total_mcc	avg_val_day	card_present	night_hour	hct_mer_mcc
0	7	9	9	1	0	5812
1	1	6	6	0	0	5999
0	81	6	6	0	0	5999
1	81	8	6	0	0	5947
1	81	6	6	0	0	5999
0	7	4	6	1	0	1405
1	81	6	10	0	1	5999
1	81	3	10	0	0	5911
0	7	2	6	1	1	1340
0	7	1	5	1	0	5948

Table 10., Sample of data where continuous vars (total_mcc and avg_val_day) are replaced by group number.

ID	avg_trans_value	avg_val_day	avg_val_day_mcc
1	avg_trans_value < 9	avg_val_day < 2.22	avg_val_day_mcc < 19.98
2	9 <= avg_trans_value < 13	2.22 <= avg_val_day < 4.84	19.98 <= avg_val_day_mcc < 24.54
3	13 <= avg_trans_value < 17	4.84 <= avg_val_day < 9	24.54 <= avg_val_day_mcc < 25.47
4	17 <= avg_trans_value < 19	9 <= avg_val_day < 10.47	25.47 <= avg_val_day_mcc < 26.52
5	19 <= avg_trans_value < 21	10.47 <= avg_val_day < 16.85	26.52 <= avg_val_day_mcc < 28.15
6	21 <= avg_trans_value < 23	16.85 <= avg_val_day < 28.47	28.15 <= avg_val_day_mcc < 34.42
7	23 <= avg_trans_value < 28	28.47 <= avg_val_day < 32.5	34.42 <= avg_val_day_mcc < 35.22
8	28 <= avg_trans_value < 78, _MISSING_	32.5 <= avg_val_day < 38.07	35.22 <= avg_val_day_mcc < 36.4
9	78 <= avg_trans_value < 107	38.07 <= avg_val_day < 83.67, _MISSING_	36.4 <= avg_val_day_mcc < 42.05
10	107 <= avg_trans_value	83.67 <= avg_val_day	42.05 <= avg_val_day_mcc, _MISSING_

Table 11., Sample of table to retrieve the interval values based on the group ID.

The creation of different sets using the bagging method can be achieved using the PROC SURVEYSELECT (Code 8.), which allows for the creation of data samples from a reference dataset. By specifying the parameter '`method=urs`', we apply unrestricted sampling, meaning that each observation has the same probability of being selected, and replacement is allowed. The option '`samprate=1`' is used to specify that you want to obtain samples of the same size as the original dataset, with the particularity that some observations will be duplicated. We specify a random seed in a macro variable `&seed` that is based upon the computer's clock. A data table is created for each bag and has a dynamically generated name using the macro variable `&i`.

```
%macro bagging(n=);
    %do i = 1 %to &n. ;

    data _null_;
        x=  ceil(uniform(today()*&i)*1000);
        call symputx('seed', x);
    run;

    proc surveyselect data=train_set out=&dm_lib..bag_&&i.
        seed= &seed.
        method=urs
        samprate=1
        outhits;
    run;
```

Code 7., Creation of the bagging sets.

The macro function takes the parameter `&n`, which is the number of samples to create, and thus the number of decision trees to generate. An iteration is then created from 1 to the number of trees we want to generate. In our case, we propose to generate 300 decision trees. We specify the parameter `&n=300` when calling the macro function `%bagging()`.To reduce the number of possible splits per node, we decide to set a maximum of 5 branches (`maxbranch=5`). This is done to increase the chances of having similar rules across the trees. We could have further lowered this threshold to increase the frequency of certain rules appearing, but doing so might compromise the precision of the trees, which could impact the rule performance. With the aim of maximizing the likelihood of having rules that appear multiple times, we set a maximum tree depth of 3 levels (`maxdepth=3`). This is because when selecting rules based on the 1% occurrence criterion, the resulting rules are simple with typically only 1 or 2 conditions at most.

```
    proc hpsplit data=&dm_lib..bag_&&i.
        maxbranch=5 minleafsize=50 intervalbins=10 mincatsize=10
        nodes=detail cvmodelfit cvmethod=random(10) maxdepth=3
        splitonce assignmissing=similar;
        prune C45;
        class %dm_nominal_input %dm_binary_input %dm_interval_input %dm_ordinal_input %dm_dec_target  ;
        model %dm_dec_target (event="1") = %dm_ordinal_input %dm_interval_input %dm_binary_input %dm_nominal_input;
        ods output HPSplit.ModelAssessment.NodeTable=&dm_lib..table_tree&&i.;
        code file="/home/sasdemo/casuser/bagging_tables/score_tables/bag_&i..sas";
    run;

    %end;
%mend bagging;

%bagging(n=300);
```

Code 8., Decision Trees parameters.

Each of the 300 decision trees will generate its own set of rules. We then aim to identify rules that appear multiple times among the trees. For this purpose, we use a PROC FREQ that returns the occurrence of all rules, and we retain the rules that appear at least in 1% of the trees, so 3 appearances minimum.

Table 12 contains a sample of rules generated with the Trees Ensemble technique, The majority of decision rules are therefore composed of 1 or 2 conditions but are quite robust due to the rule selection through the voting approach.

Rule	DR	Frauds Detected	Amount Saved	FPR	Fraud Rate for Rule	Lift	Number of Alerts	IR	Number of Conditions
hct_term_cntry_code = 724 AND hct_mer_mcc = 7311	0,59%	7	$ 502,92	5:7	58%	4,99	12	0,12%	2
hct_term_cntry_code = 752	6,26%	74	$ 730,47	14:74	84%	7,19	88	0,87%	1
hct_term_cntry_code = 752 AND prev_tran_dt < 36	4,48%	53	$ 529,47	0:53	100%	8,55	53	0,52%	2
hct_term_cntry_code IN (203,344,376,504,752)	7,19%	85	$ 1 293,46	18:85	83%	7,06	103	1,02%	1
ucm_pos = 1 AND hct_term_cntry_code = 372	0,51%	6	$ -	6:6	50%	4,27	12	0,12%	2

Table 9., Sample of rules generated with Trees Ensemble.