ACOTSP 1000-4500: Estimation of parameter importance with Random Forests (Ranking)

Leslie Pérez Cáceres

In this document we are testing how to use Random Forest to assess importance and interactions of parameters based on the data gathered by irace.

For the analysis below we use as example ACOTSP:

- 20 secs cut off time (long)
- 11 parameters
- 400 instances sizes (1000,1500,2000,2500,3000,3500,4000,4500)
- 3000 experiments configuration budget / 5000 experiments configuration budget

We use random forest for predicting

1. configuration normalized ranking
2. configuration imputed ranking
3. configuration imputed ranking quartile

Models are trained using default settings of the package Random Forest, excepting the number of trees that was set to 300. We have access to the following measures that can be considered indicators of importance:

- mean_min_depth: mean depth of the subtree closest to the tree root, where a variable is root of the sub tree.
- no_of_nodes: number of nodes in which the variable was used to split
- mse_increase: increment of prediction mean squared error
- no_of_trees: number of trees in which the variable was used
- times_a_root: number of times a variable was selected as root variable
- accuracy_decrease: measure of the classification accuracy (only for classification models)

For this analysis we use data generated by irace, it is possible to select a subset of the data and perform the analysis. The data is imputed and there is a procedure to analyze importance based on a reference variable and a retraining scheme to assess real importance in conditional parameters. The instance is also used as a predictor in this data. (details in another document)

There are some things that should be investigated regarding the best way to use the models to asses interaction and importance:

1. Discretising numerical variables for prediction: based on a comment I got that RF are biased to select numerical variables as split. This will be particularly interesting and in line with the fact that irace defines a sampling range around the current value.

2. Adjusting the number of trees and depth of them. My intuition is that smaller more smaller trees would be more useful in the task of detecting importance. This is because lower level splits are not as interesting and higher level splits. Also, to be used as a post-execution analysis tool the execution time required to build the model depends on these parameters.

3. How to understand how this importance or interaction is materialized i.e., which are the best parameter values ans how these interact. Can we have an heuristic idea of this interpreting the forest splits?

4. How to interpret instance importance and interaction when using models not predicting directly the performance. Can this help detecting heterogeneous sets?

In the following examples we use the *ranking as the dependent variable* when training, thus the trained random forest predicts the mean ranking of a configuration. In these experiments the variable instance should not be a good as predictor compared to a model trained to predict performance directly. Despite this, interactions of the instance variable with other variables are possible and might indicate heterogeneity of the benchmark. In this model all irace data execution was used for training. Note that due to the focused nature of irace data (given model convergence) there might be contradicting or not consistent interactions in the data.

1. Configuration normalized ranking as dependent variable

The data set used to train this model defines the variable to predict as the normalized ranking. Ranking is normalized as:

```
(rank - 1) / (n_instances - 1)
```

where `rank` is the rank of a of a configuration in an instance and `n_instances` is the number of instances used during the execution of irace.

We show importance measures ordered by the mean min depth given that the interaction analysis is based on this measure.

```
load("../model_data/model-acotsp1000-4500-ranking.Rdata")
importance_frame = model$importance_frame
important_parameters = model$important_parameters
full_interactions_frame = model$full_interactions_frame
interactions_frame = model$interactions_frame
kable(importance_frame[order(importance_frame[,"mean_min_depth"]),])  %>%
      kable_styling(latex_options="scale_down") %>%
  kable_styling(latex_options = "HOLD_position")
```

	variable	mean_min_depth	no_of_nodes	mse_increase	node_purity_increase	no_of_trees	times_a_root	p_value
9	localsearch	1.410000	6795	0.0093434	32.8834391	300	78	1
8	instance	1.616667	68604	0.0130306	38.5686370	300	22	0
5	dlb	2.020000	5015	0.0064653	16.9390213	300	39	1
11	q0	2.063333	35802	0.0048461	26.4201492	300	79	0
10	nnls	2.166667	42729	0.0043035	14.1360058	300	28	0
1	algorithm	2.436667	9312	0.0040585	16.1638329	300	40	1
2	alpha	2.760000	56018	0.0022852	7.2487097	300	8	0
3	ants	3.186667	50796	0.0021940	5.8293829	300	5	0
7	elitistants	3.273333	5264	0.0009673	2.6775270	300	1	1
13	rho	3.633333	50891	0.0004153	3.1086769	300	0	0
4	beta	3.670000	53628	0.0005077	3.5299944	300	0	0
6	dummy	4.200000	39887	0.0000843	1.8058573	300	0	0
12	rasrank	4.936667	6034	0.0000619	0.5835483	300	0	1

Ordered importance measures:

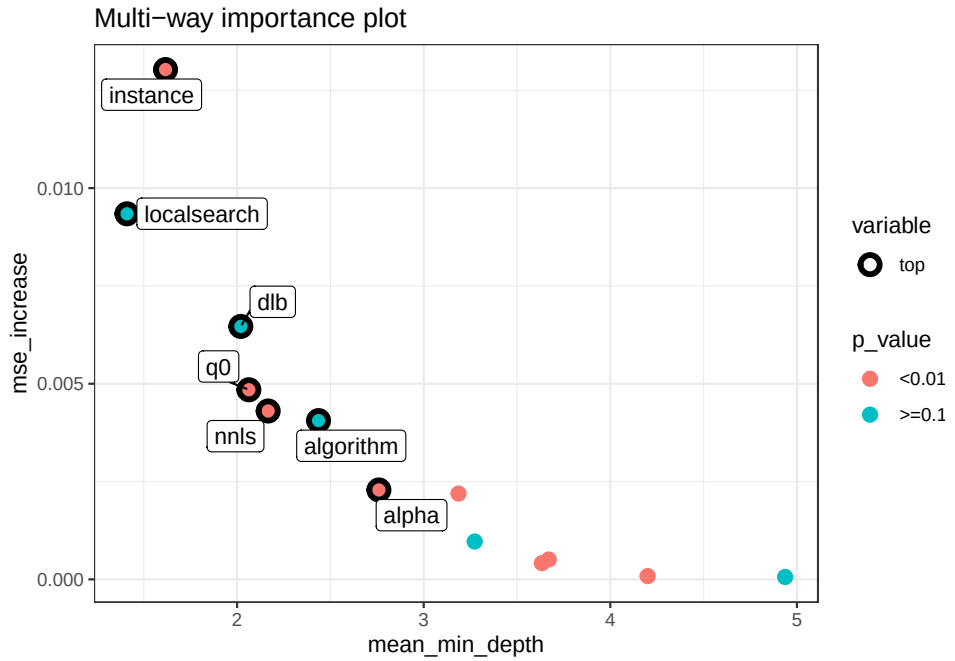
```
imp <- model$importance_frame
imp[,"variable"] <- as.character(imp[,"variable"])
ordered_imp <- matrix(NA, nrow=nrow(imp), ncol=ncol(imp)-2)
measures <- colnames(imp)[!(colnames(imp) %in% c("variable", "p_value"))]
colnames(ordered_imp) <- measures
ordered_imp[,"mean_min_depth"] <- imp[,"variable"][order(imp[,"mean_min_depth"])]
ordered_imp[,"no_of_nodes"] <- imp[,"variable"][order(imp[,"no_of_nodes"], decreasing=TRUE)]
ordered_imp[,"mse_increase"] <- imp[,"variable"][order(abs(imp[,"mse_increase"]), decreasing=TRUE)]
ordered_imp[,"node_purity_increase"] <- imp[,"variable"][order(imp[,"node_purity_increase"], decreasing=
```

```
ordered_imp[,"no_of_trees"] <- imp[,"variable"][order(imp[,"no_of_trees"], decreasing=TRUE)]
ordered_imp[,"times_a_root"] <- imp[,"variable"][order(imp[,"times_a_root"], decreasing=TRUE)]
kable(ordered_imp) %>%
      kable_styling(latex_options="scale_down") %>%
  kable_styling(latex_options = "HOLD_position")
```

mean_min_depth	no_of_nodes	mse_increase	node_purity_increase	no_of_trees	times_a_root
localsearch	instance	instance	instance	localsearch	q0
instance	alpha	localsearch	localsearch	instance	localsearch
dlb	beta	dlb	q0	dlb	algorithm
q0	rho	q0	dlb	q0	dlb
nnls	ants	nnls	algorithm	nnls	nnls
algorithm	nnls	algorithm	nnls	algorithm	instance
alpha	dummy	alpha	alpha	alpha	alpha
ants	q0	ants	ants	ants	ants
elitistants	algorithm	elitistants	beta	elitistants	elitistants
rho	localsearch	beta	rho	rho	rho
beta	rasrank	rho	elitistants	beta	beta
dummy	elitistants	dummy	dummy	dummy	dummy
rasrank	dlb	rasrank	rasrank	rasrank	rasrank

We can plot two importance measures using the randomForestExplainer package, we choose the to show the mean min depth in the x axis and the mse increase in the y axis. Top 7 variables are highlighted. In this case, since we are not predicting performance the effect of instance variable in model performance must be interpreted carefully. These plots are interesting given that contrast importance for ranking prediction in terms of accuracy (mse_increase) and in terms of early importance the tree more in line with classification goals (mean min depth).

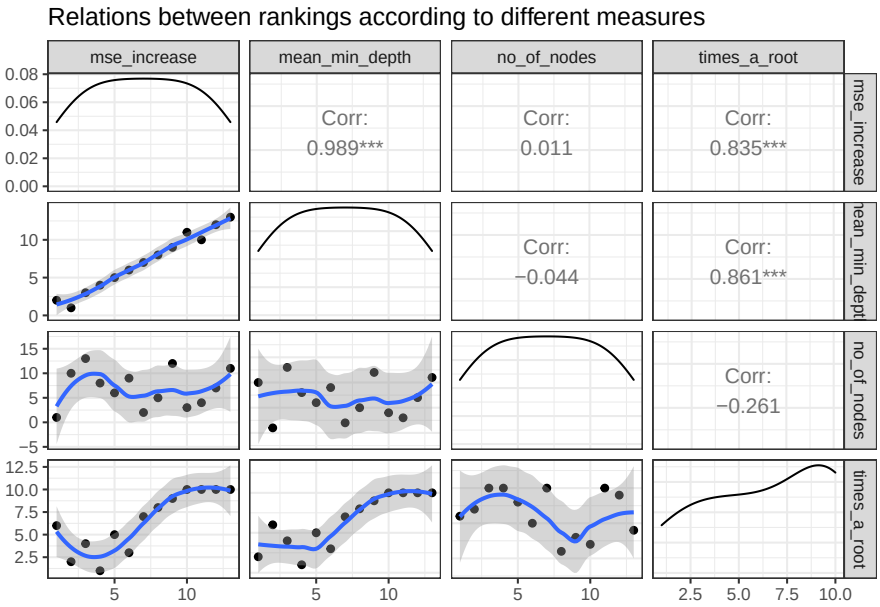
```
suppressWarnings(plot_multi_way_importance(importance_frame, size_measure = "p_value",
                          y_measure="mse_increase", x_measure="mean_min_depth",
                          no_of_labels=7))
```



We can also visualize the relationship between importance measures, this could help to understand which

indicator is more suited or can be used as a complement of other:

```
plot_importance_rankings(importance_frame, measures=c("mse_increase", "mean_min_depth",
                                                      "no_of_nodes", "times_a_root"))
```



We perform an analysis of conditional parameters importance using a filtering and re training strategy and apply an irrelevant parameter filter by including a reference parameter. After this process the most important 5 parameters are detected.

Important parameters:

```
print(important_parameters)
```

```
## [1] "localsearch" "instance"    "dlb"         "algorithm"   "dummy"
```

Next, we run the interaction analysis only over important parameters. This is assuming the interactions one cares to detect are related to them, which might be not entirely correct ans should be evaluated as heuristic. The importance of interactions is calculated based on the mean min depth indicator in its conditional version.

Parameter interaction importance:

```
kable(full_interactions_frame)  %>%
  kable_styling(latex_options="scale_down") %>%
  kable_styling(latex_options = "HOLD_position")
```

variable	root_variable	mean_min_depth	occurrences	interaction	uncond_mean_min_depth
instance	algorithm	0.9185185	270	algorithm:instance	1.627122
instance	localsearch	1.1765806	264	localsearch:instance	1.627122
instance	dummy	1.2953517	262	dummy:instance	1.627122
algorithm	algorithm	1.7574260	260	algorithm:algorithm	1.503789
instance	instance	1.2434287	260	instance:instance	1.627122
dummy	algorithm	1.4733556	258	algorithm:dummy	1.670911
dummy	dummy	1.6271688	257	dummy:dummy	1.670911
instance	dlb	1.3394321	256	dlb:instance	1.627122
localsearch	localsearch	1.9741863	252	localsearch:localsearch	1.674700
dummy	instance	1.5942648	250	instance:dummy	1.670911
dummy	localsearch	1.6132934	250	localsearch:dummy	1.670911
dummy	dlb	1.6199630	243	dlb:dummy	1.670911
localsearch	algorithm	2.2847021	231	algorithm:localsearch	1.674700
localsearch	instance	2.0432429	229	instance:localsearch	1.674700
algorithm	dummy	2.2826498	225	dummy:algorithm	1.503789
localsearch	dummy	2.2382053	225	dummy:localsearch	1.674700
algorithm	localsearch	2.1587230	224	localsearch:algorithm	1.503789
algorithm	instance	2.3624303	221	instance:algorithm	1.503789
algorithm	dlb	2.2858025	220	dlb:algorithm	1.503789
dlb	algorithm	2.6056485	220	algorithm:dlb	1.636667
localsearch	dlb	2.3965556	219	dlb:localsearch	1.674700
dlb	dummy	2.6040268	210	dummy:dlb	1.636667
dlb	instance	2.8070358	208	instance:dlb	1.636667
dlb	localsearch	3.1263749	198	localsearch:dlb	1.636667
dlb	dlb	3.7296667	177	dlb:dlb	1.636667

For this example we aggregate bidirectional (`param1:param2` and `param2:param2`) interactions, this is done given that we assume that the hierarchy of the interaction is not well represented in the forest and this should be analyzed separately. We also filter interactions using the reference parameter to remove all not relevant interactions. Once this process is done, the importance of most relevant interactions is summarized.

Relevant parameter interactions:

```
kable(interactions_frame) %>%
  kable_styling(latex_options="scale_down") %>%
  kable_styling(latex_options = "HOLD_position")
```

variable	root_variable	mean_min_depth	occurrences	interaction	uncond_mean_min_depth

2. Configuration imputed ranking as dependent variable

For training the model we must impute the dependent variable (ranking) since some configurations are not executed in some instances. We assume these configurations to be worst than the configurations executed in the instance, after imputation all configurations have a ranking assigned for each instance.

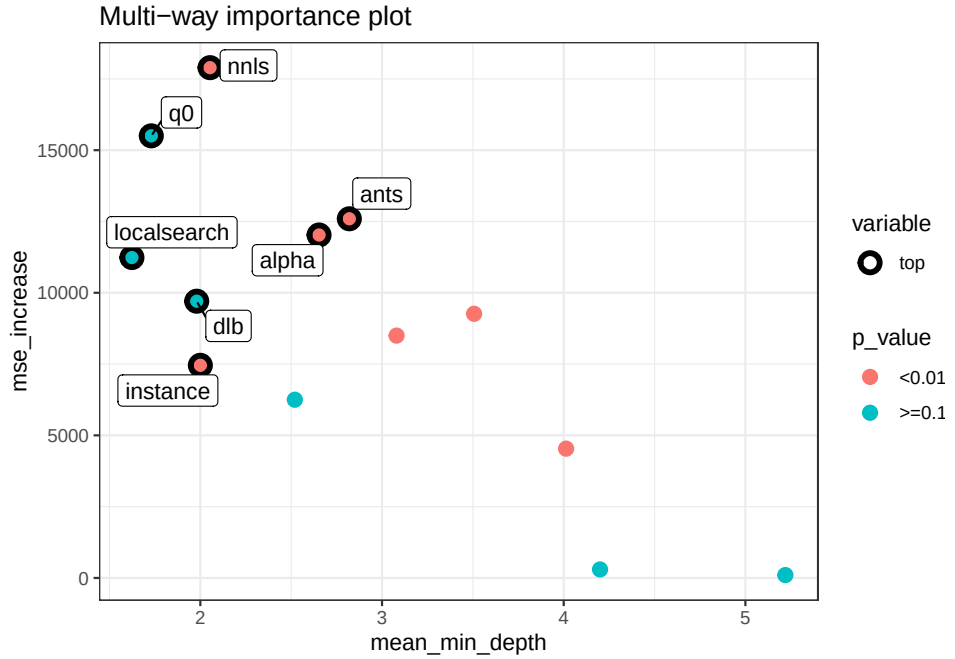
The parameter importance measures:

```
load("../model_data/model-acotsp1000-4500-iranking.Rdata")
importance_frame = model$importance_frame
important_parameters = model$important_parameters
full_interactions_frame = model$full_interactions_frame
interactions_frame = model$interactions_frame
kable(importance_frame[order(importance_frame[,"mean_min_depth"]),])  %>%
      kable_styling(latex_options="scale_down") %>%
  kable_styling(latex_options = "HOLD_position")
```

	variable	mean_min_depth	no_of_nodes	mse_increase	node_purity_increase	no_of_trees	times_a_root	p_value
9	localsearch	1.623333	11882	11235.5156	34967582	300	48	1
11	q0	1.730000	51788	15503.0753	67282717	300	107	1
5	dlb	1.980000	9196	9704.2813	27668905	300	35	1
8	instance	2.000000	229480	7456.2331	90066698	300	0	0
10	nnls	2.053333	69592	17894.0742	61461180	300	20	0
1	algorithm	2.520000	18334	6250.2807	26133923	300	64	1
2	alpha	2.653333	89693	12026.1719	46868703	300	8	0
3	ants	2.820000	85007	12595.0955	44566770	300	11	0
4	beta	3.080000	88708	8498.4007	36085079	300	3	0
13	rho	3.506667	85008	9263.9234	36924048	300	0	0
6	dummy	4.013333	66852	4533.3909	19901134	300	0	0
7	elitistants	4.200000	12483	297.8099	1887500	300	4	1
12	rasrank	5.220000	13336	101.2906	1837704	300	0	1

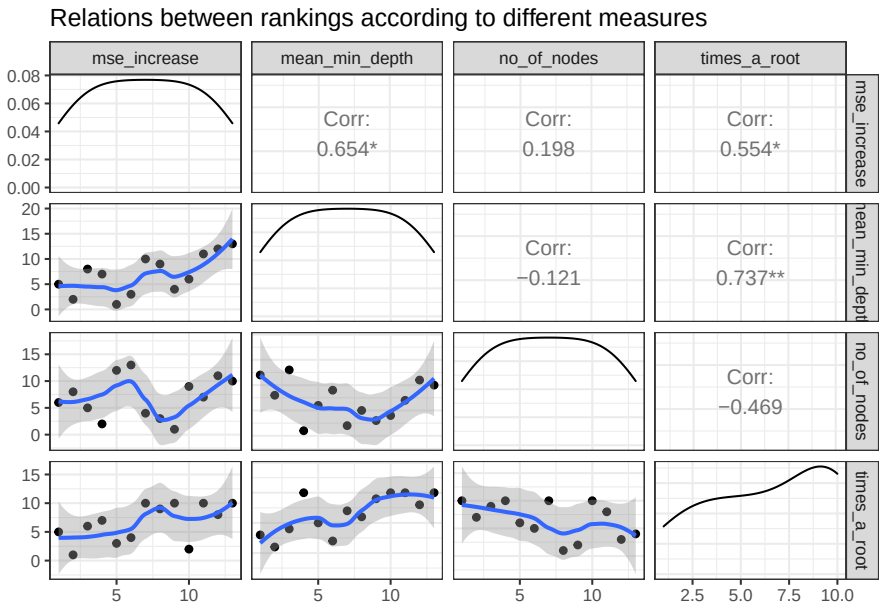
We plot importance as above to visualize how the mse increase and mean min depth are related:

```
suppressWarnings(plot_multi_way_importance(importance_frame, size_measure = "p_value",
                           y_measure="mse_increase", x_measure="mean_min_depth",
                           no_of_labels=7))
```



We visualize the relationship between importance measures, this could help to understand which indicator is more suited or can be used as a complement of other.

```
plot_importance_rankings(importance_frame, measures=c("mse_increase", "mean_min_depth",
                                                      "no_of_nodes", "times_a_root"))
```



Important parameters:

```
print(important_parameters)
```

```
## [1] "localsearch" "nnls"        "ants"        "dummy"
```

Parameter interaction importance:

```
kable(full_interactions_frame)  %>%
  kable_styling(latex_options="scale_down") %>%
  kable_styling(latex_options = "HOLD_position")
```

variable	root_variable	mean_min_depth	occurrences	interaction	uncond_mean_min_depth
ants	localsearch	0.8440678	295	localsearch:ants	1.270000
ants	nnls	0.8355480	294	nnls:ants	1.270000
ants	dummy	0.8185085	292	dummy:ants	1.270000
nnls	localsearch	1.0271638	291	localsearch:nnls	1.450000
ants	ants	0.9355367	290	ants:ants	1.270000
dummy	nnls	0.9285424	289	nnls:dummy	1.386667
localsearch	localsearch	1.5975367	288	localsearch:localsearch	1.330000
dummy	dummy	1.1363616	287	dummy:dummy	1.386667
dummy	localsearch	1.1119548	287	localsearch:dummy	1.386667
nnls	nnls	1.0708249	287	nnls:nnls	1.450000
dummy	ants	1.1124407	286	ants:dummy	1.386667
nnls	ants	1.2303955	285	ants:nnls	1.450000
nnls	dummy	1.2667345	284	dummy:nnls	1.450000
localsearch	ants	2.6102938	246	ants:localsearch	1.330000
localsearch	dummy	3.0259661	232	dummy:localsearch	1.330000
localsearch	nnls	3.3392090	225	nnls:localsearch	1.330000

For this example we aggregate bidirectional (`param1:param2` and `param2:param2`) interactions, this is done given that we assume that the hierarchy of the interaction is not well represented in the forest and this should be analyzed separately. We also filter interactions using the reference parameter to remove all not relevant interactions. Once this process is done, the importance of most relevant interactions is summarized.

Relevant parameter interactions:

```
kable(interactions_frame) %>%
  kable_styling(latex_options="scale_down") %>%
  kable_styling(latex_options = "HOLD_position")
```

variable	root_variable	mean_min_depth	occurrences	interaction	uncond_mean_min_depth
ants	localsearch	0.8440678	541	localsearch:ants	1.27

3.Configuration imputed ranking quantile as dependent variable

In this example we use the imputed ranking quartile as the dependent variable when training, thus the trained random forest predicts the mean ranking quartile of a configuration.

In this example the ACOTSP data is used to predict solution quality similarly of how it is used in SMAC and GGA++. The intuition is that in these models instance is probably the best predictor of the quality due to the different sizes that compose the instance set and the nature of the configuration objective (tour length).

Parameter importance:

```
load("../model_data/model-acotsp1000-4500-qranking.Rdata")
importance_frame = model$importance_frame
full_interactions_frame = model$full_interactions_frame
interactions_frame = model$interactions_frame
important_parameters = model$important_parameters
kable(importance_frame[order(importance_frame[,"mean_min_depth"]),]) %>%
      kable_styling(latex_options="scale_down") %>%
  kable_styling(latex_options = "HOLD_position")
```

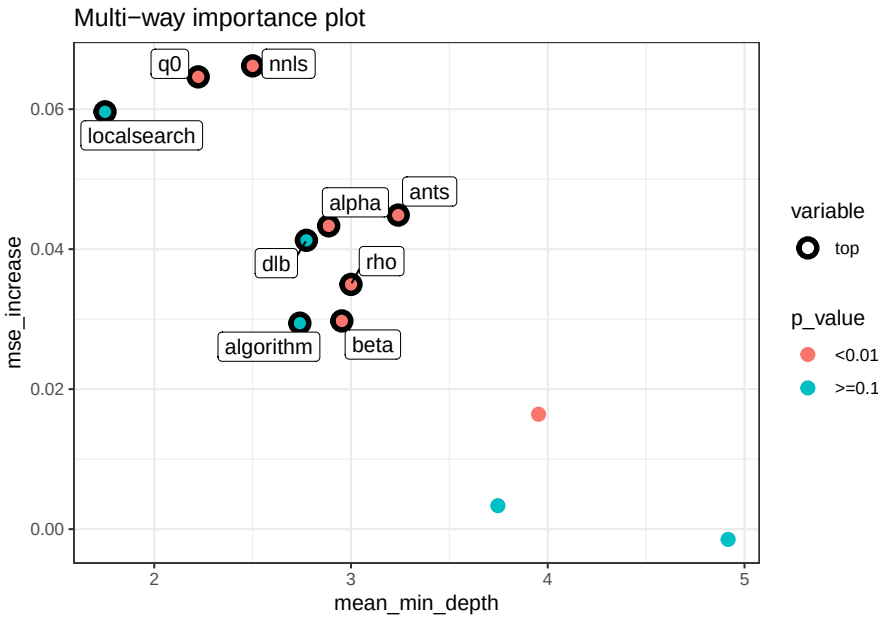
	variable	mean_min_depth	no_of_nodes	mse_increase	node_purity_increase	no_of_trees	times_a_root	p_value
8	instance	1.180000	92480	0.6756812	5449.1415	300	92	0
9	localsearch	1.750000	9784	0.0596248	248.6501	300	73	1
11	q0	2.223333	52232	0.0646193	378.0681	300	52	0
10	nnls	2.500000	65519	0.0661943	354.1018	300	12	0
1	algorithm	2.740000	12850	0.0294282	144.2972	300	34	1
5	dlb	2.773333	9676	0.0412675	138.7270	300	20	1
2	alpha	2.886667	80226	0.0433399	339.5327	300	7	0
4	beta	2.953333	79428	0.0297496	284.9653	300	0	0
13	rho	3.000000	76885	0.0349673	276.6383	300	0	0
3	ants	3.240000	75643	0.0448713	300.1704	300	8	0
7	elitistants	3.746667	8324	0.0033530	28.7553	300	2	1
6	dummy	3.953333	61504	0.0164171	165.5993	300	0	0
12	rasrank	4.916667	11433	-0.0014644	23.0296	300	0	1

As expected the instance is the most important variable, the other parameters do not have the same ranking in the different benchmarks.

We plot the importance measures min mean depth and the mse increase to observe better this difference. The instance is removed in the plot so its possible to see more clearly other parameters.

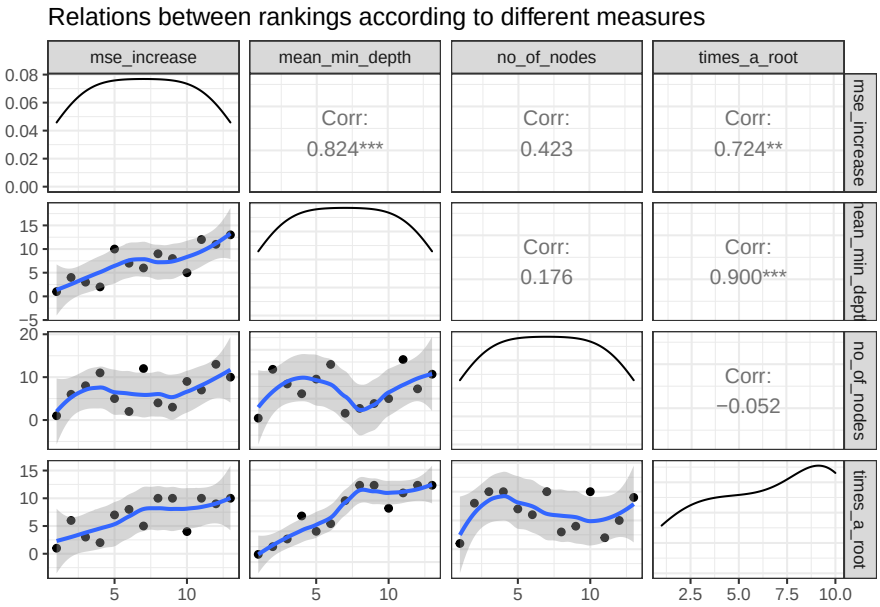
```
plot_multi_way_importance(importance_frame[importance_frame[,"variable"]!="instance",], size_measure =
                          y_measure="mse_increase", x_measure="mean_min_depth", no_of_labels=7)
```

```
## Warning: Using alpha for a discrete variable is not advised.
```



We visualize the relationship between importance measures, this could help to understand which indicator is more suited or can be used as a complement of other.

```
plot_importance_rankings(importance_frame, measures=c("mse_increase", "mean_min_depth",
                                                      "no_of_nodes", "times_a_root"))
```



Filtered important parameters:

```
print(important_parameters)
```

```
## [1] "instance"    "localsearch" "dummy"
```

We use the mean_min_depth measure as reference given that this measure can be extended to assess interactions and run the interaction analysis to find the importance of interactions between the selected important parameters. This is done by extending the definition of mean_min_depth to be calculated in max subtrees in which one variable is root.

```
kable(full_interactions_frame)  %>%
  kable_styling(latex_options="scale_down") %>%
  kable_styling(latex_options = "HOLD_position")
```

variable	root_variable	mean_min_depth	occurrences	interaction	uncond_mean_min_depth
instance	instance	0.5148148	270	instance:instance	1.1941667
instance	dummy	0.7153970	268	dummy:instance	1.1941667
dummy	dummy	0.9360585	267	dummy:dummy	0.9902778
instance	localsearch	0.8980741	258	localsearch:instance	1.1941667
dummy	instance	1.6828657	232	instance:dummy	0.9902778
localsearch	localsearch	2.2196543	229	localsearch:localsearch	1.0400000
dummy	localsearch	1.8261235	227	localsearch:dummy	0.9902778
localsearch	dummy	2.0773318	227	dummy:localsearch	1.0400000
localsearch	instance	2.6959596	207	instance:localsearch	1.0400000

We aggregate interactions with their inverse given that for now we are not interested in the direction of the interaction. Once we aggregate to have bidirectional interactions and filter dummy interactions, we get a matrix where the relevant importance of interactions are summarized:

```
kable(interactions_frame)  %>%
  kable_styling(latex_options="scale_down") %>%
  kable_styling(latex_options = "HOLD_position")
```

variable	root_variable	mean_min_depth	occurrences	interaction	uncond_mean_min_depth