Instance-level model exploration with triplot

It is a common situation that a tabular model uses dozens, hundreds or even more correlated variables. In such cases local explanation techniques like Shapley values or Break-Down make little sense as they do not reflect the model behaviour in an easy to understand way. The local triplot addresses this problem.

Local triplot combines the approach to explanations used by LIME methods and visual techniques introduced in a global triplot. It's based on the instance-level explainer called ***predict_aspects***, also provided by the triplot package. The main goal of *predict_aspects* is to assess the feature contribution to the prediction, similarly to Break Down or Shapley values. However, *predict_aspects* provides the additional ability to calculate the importance of groups of variables. Those groups of variables we are calling aspects.

We are going to observe how local triplot and *predict_aspects* work by using them on the same dataset that was explored in the previous blog post—FIFA 20 (dataset description) from Kaggle.

```
library("DALEX")
data(fifa)
fifa$value_eur <- fifa$value_eur/10^6
fifa[, c("nationality", "overall", "potential",
         "wage_eur")] <- NULL
```

In the beginning, we get the FIFA dataset from the DALEX. There are a couple of variables that could be treated as target (player's potential, wage, etc). Among them we choose player's value in Euro as the predicted variable and remove the rest of them, as well as we remove nationality variable. We also rescale the target, so it's more readable.

After the data preparation step is finished, we build a random forest model from ranger package. Afterward, we wrap it in a DALEX explainer.

```
library("ranger")
set.seed(2020)
fifa_model_rf <- ranger(value_eur~., data = fifa)
fifa_explainer_rf <- DALEX::explain(fifa_model_rf,
                                    data = fifa[,-1],
                                    y = fifa$value_eur,
                                    label = "Random Forest")
```

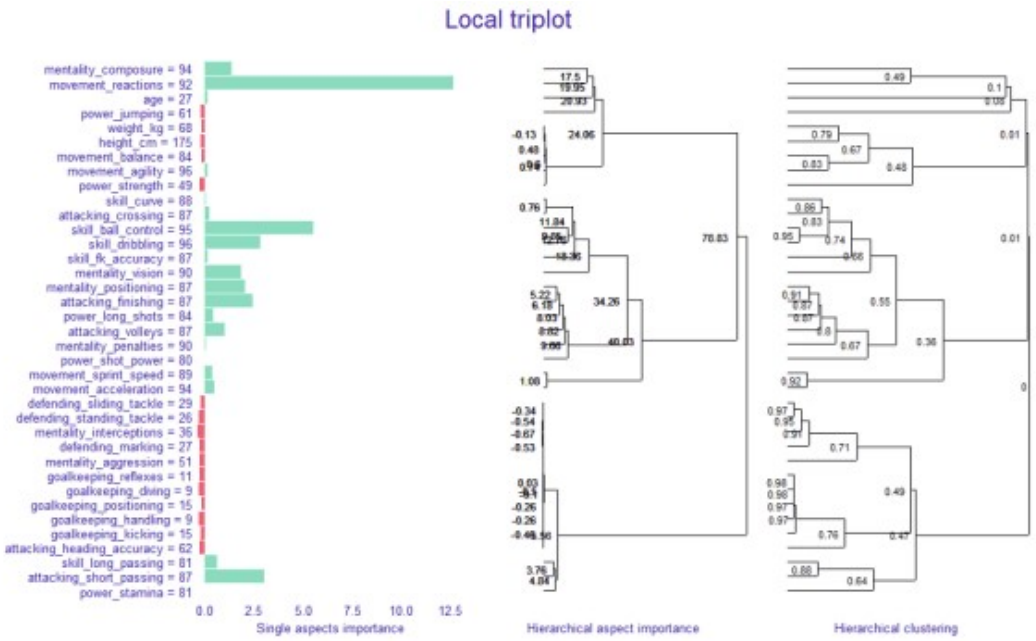
Main difference between global and local triplot in terms of usage, is providing local triplot with the selected observation. In this example, we are looking to explain the model's prediction for the the player with the highest value in Euro.

```
top_player <- fifa[order(fifa$value_eur, decreasing = TRUE),][1,]
```

Now, we can finally call *predict_triplot*. We provide it with some additional parameters: we are using Pearson correlation method and setting the sample size to be at 5000.

```
fifa_triplot_local <- predict_triplot(fifa_explainer, top_player,
                                      N = 5000,
                                      cor_method = "pearson")
plot(fifa_triplot_local)
```

In result we see local triplot, a picture that is an analog of the global triplot.



Local triplot for top_player

Here we can see in one image:

- the contribution to the prediction of every **single feature (**the left panel),
- **correlation structure** of features visualized by hierarchical clustering (the right panel),
- **the contribution of aspects**, that are built in the order determined by the hierarchical clustering (the middle panel).

Now, let's build a second model. This time it will be a gradient boosting machine from gbm package.

```
library("gbm")
fifa_model_gbm <- gbm(value_eur ~ ., data = fifa,
                      n.trees = 250,
                      interaction.depth = 4,
                      distribution = "gaussian")
fifa_explainer_gbm <- DALEX::explain(fifa_model_gbm,
                                     data = fifa[,-1],
                                     y = fifa$value_eur,
                                     label = "Gradient Boosting")
```

When we compare the value prediction of the *top_player* in both models, we notice that the prediction is too low. However, in case of random forest, it is significantly lower, than in the case of gbm's model.

```
top_player$value_eur
## 105.5
fifa_expl_rf$y_hat[order(fifa$value_eur, decreasing = TRUE)[1]]
## 89.91145
fifa_expl_gbm$y_hat[order(fifa$value_eur, decreasing = TRUE)[1]]
## 102.1183
```

We can use *predict_aspects* to see which characteristics of the player contribute to predictions. Since there are many variables here, we use the *predict_aspects* capability of analysing group of variables: we group player features into aspects in following way.

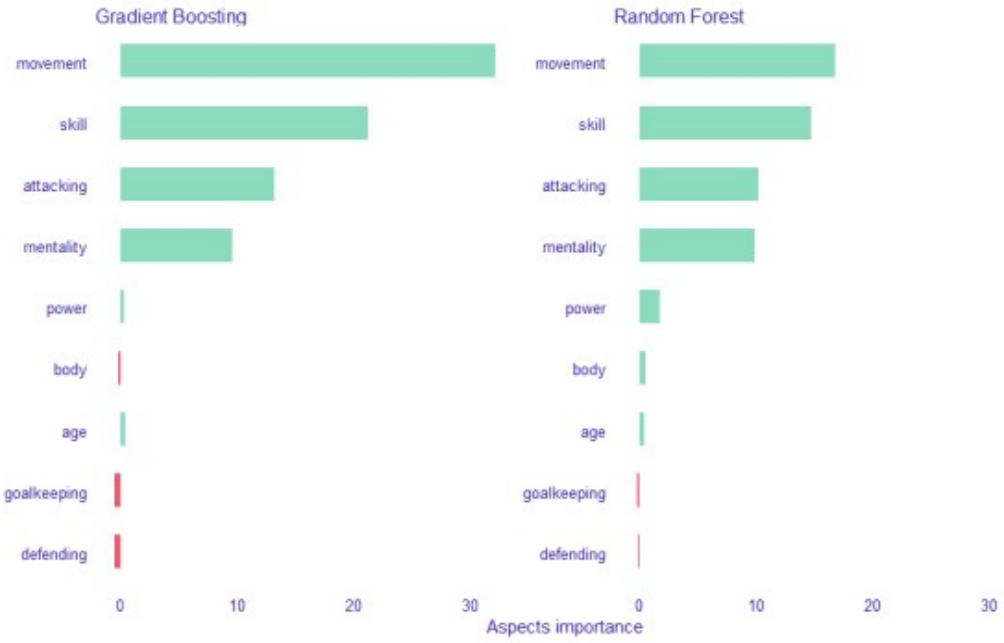
```
fifa_aspects <- list(
  "age" = "age",
  "body" = c("height_cm", "weight_kg"),
  "attacking" = c("attacking_crossing",
                  "attacking_finishing",
```

```
                "attacking_heading_accuracy",
                "attacking_short_passing",
                "attacking_volleys"),
  "skill" = c("skill_dribbling",
              "skill_curve", "skill_fk_accuracy",
              "skill_long_passing",
              "skill_ball_control"),
  "movement" = c("movement_acceleration", "movement_sprint_speed",
                 "movement_agility", "movement_reactions",
                 "movement_balance"),
  "power" = c("power_shot_power", "power_jumping", "power_stamina",
              "power_strength", "power_long_shots"),
  "mentality" = c("mentality_aggression", "mentality_interceptions",
                  "mentality_positioning", "mentality_vision",
                  "mentality_penalties", "mentality_composure"),
  "defending" = c("defending_marking", "defending_standing_tackle",
                  "defending_sliding_tackle"),
  "goalkeeping" = c("goalkeeping_diving",
                    "goalkeeping_handling", "goalkeeping_kicking",
                    "goalkeeping_positioning",
                    "goalkeeping_reflexes"))
```

To use *predict_aspects* method, we need to provide: an explainer, an observation to be explained and the features grouped in a form of a named list.

```
fifa_pa_rf <- predict_aspects(fifa_expl_rf,
                              new_observation = top_player,
                              variable_groups = fifa_aspects)
fifa_pa_gbm <- predict_aspects(fifa_expl_gbm,
                               new_observation = top_player,
                               variable_groups = fifa_aspects)
plot(fifa_pa_rf, fifa_pa_gbm)
```

In result we get two plots side by side and a possibility to compare aspects' contribution.



Movement, *general*, *attacking* and *mentality*—those groups of features have main contribution to the prediction in case of both models. However, in case of random forest we can see that the contribution of those 4 groups is more even. Additionally, in case of *movement* and *skill* aspects, the contribution is lower

than in case of gbm.

As shown above, we can use *predict_aspects* not only to assess aspects' contribution to the prediction. But also to compare between models the contribution of aspects . And alternatively, we can compare the contribution of aspects to the predictions of different observations in one model.