Penguins

Our journey starts in Antarctica. Quite a weird place for XAI stuff one may say. However, in today's blog, 3 different species of penguins exactly from that continent will serve as our companions. penguins is a dataset coming from the palmerpenguins R package. As README states, the data were collected and published by Dr. Kristen Gorman and contain information about 344 penguins living in 3 different islands i the Plamer Archipelago. Authors of the package see penguins as an alternative for iris, let's see!

```
library(palmerpenguins)
data_penguins <- na.omit(palmerpenguins::penguins)
```

Predict function target column

DALEX 2.1.0 brought a new parameter to explain a function which is predict_function_target_column. It allows users to actually steer the flow of the model's response that is taken into consideration in explainir classification task models. In previous versions, whenever the default predict_function was used, DALEX was returning the second column of the output probabilities matrix for binary classification, and the whole matrix of probabilities for multiclass classification. While that default behavior was preserved, now bypassing predict_function_target_column parameter we can force DALEX to take the column that we ar interested in without being forced to pass the custom predict_function. What's even more fantastic is that you can do it not only for binary classification models but also for multiclass classification changing the task to one vs others binary classification. It's a super handy tool whenever one of the classes in your multiclass task is far more important than the others and such analysis, made from both perspectives ma be useful. The usage of that parameter is super useful. It accepts both numeric and character inputs and should be understood as either order of the column in the probabilities matrix or the name of the column that should be extracted. To be even more precise, the parameter's value is used directly to index the column. Keep in mind that some of the engines like gbm returns a single vector for multiclass classification. The change does not affect such models.

Creation of model and explainer

That being said it's high time to make some models and show new functionalities in practice. For that purpose, we will need two predictive models. Let them be simple, performance is not what we seek for today. The first model is going to be a binary classification. For this purpose, we will need to create a new variable, is_adelie which I think is really self-explanatory. The second model will be a multiclass classification for species variable. The engine behind both of them will be a ranger.

```
library("ranger")
library("DALEX")
model_multiclass <- ranger(species~., data = data_penguins,
    probability = TRUE, num.trees = 100)
explain_multiclass_one_vs_others <- explain(
    model_multiclass,
    data_penguins,
    data_penguins$species == "Adelie",
    label = "Ranger penguins multiclass",
    predict_function_target_column = "Adelie")
explain_multiclass <- explain(
    model_multiclass,
    data_penguins,
    data_penguins$species,
```

```
    label = "Ranger penguins multiclass",
    colorize = FALSE)

model_binary <- ranger((species=="Adelie")~., data = data_penguins,
    probability = TRUE, num.trees = 100)
explain_binary <- explain(
    model_binary,
    data_penguins,
    data_penguins$species == "Adelie",
    label = "Ranger penguins multiclass",
    colorize = FALSE,
    predict_function_target_column = 2)
```

As you see the usage is very simple! Therefore let's explain some models!

Model performance is always a good place to start. An important note is that multiclass models with passed predict_function_target_column parameter are treated as standard binary classification, so measures for binary will be displayed. Let's see the difference

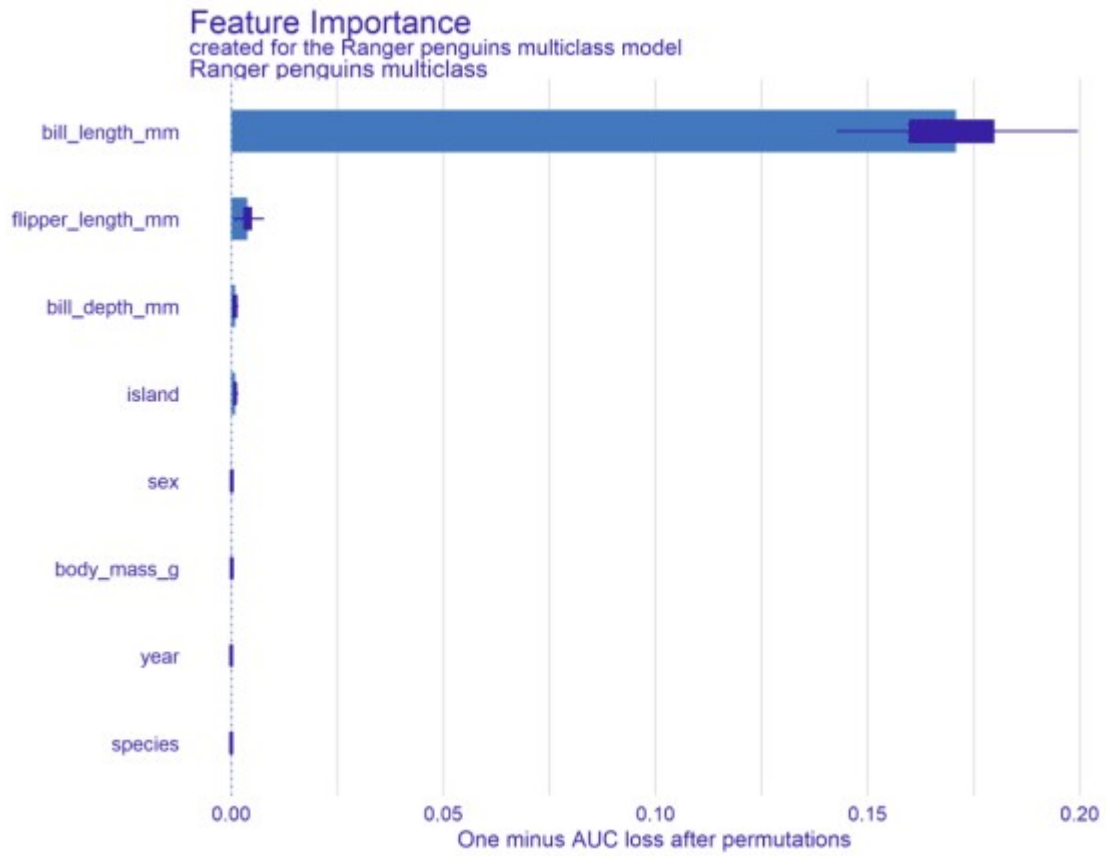
```
(mp_one_vs_others <-
    model_performance(explain_multiclass_one_vs_others))
(mp <-
    model_performance(explain_multiclass))
```

```
Measures for:  classification
recall     : 1
precision  : 0.9931973
f1         : 0.996587
accuracy   : 0.996997
auc        : 0.9999634

Residuals:
          0%          10%          20%          30%          40%          50%          60%          70%          80%          90%         100%
-0.596725950 -0.046987590 -0.013203030 -0.001923077  0.000000000  0.000000000  0.001000000  0.009201587  0.020106756  0.039770635  0.424615079
Measures for:  multiclass
micro_f1   : 0.996997
macro_f1   : 0.99746
w_macro_f1 : 0.9972875
accuracy   : 0.996997
w_macro_auc: 0.9999839

Residuals:
          0%          10%          20%          30%          40%          50%          60%          70%          80%          90%         100%
0.000000000 0.000000000 0.000000000 0.001923077 0.007552381 0.012571429 0.018762221 0.031489744 0.043319048 0.097809524 0.603725950
```

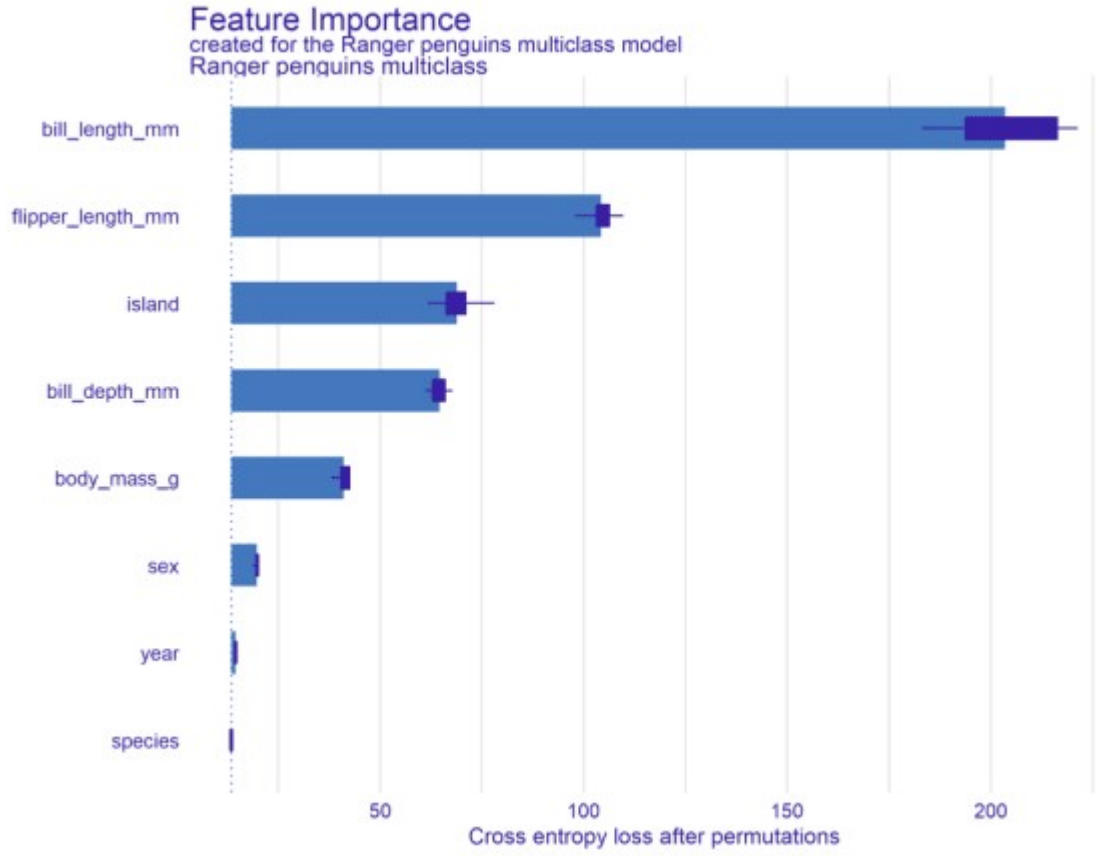
model_performance for both models

Another way that lays in front of us with the new option is calculating feature importance using different measures. Default measures are loss in cross-entropy for multiclass and one minus AUC for binary. With the change we can in fact calculate which features are most important for predictions of one specific class isn't that amazing?

```
fi_one_vs_others <- model_parts(explain_multiclass_one_vs_others)
fi <- model_parts(explain_multiclass)
plot(fi_one_vs_others)
plot(fi)
```



feature importance for a multiclass model with chosen adelie variable

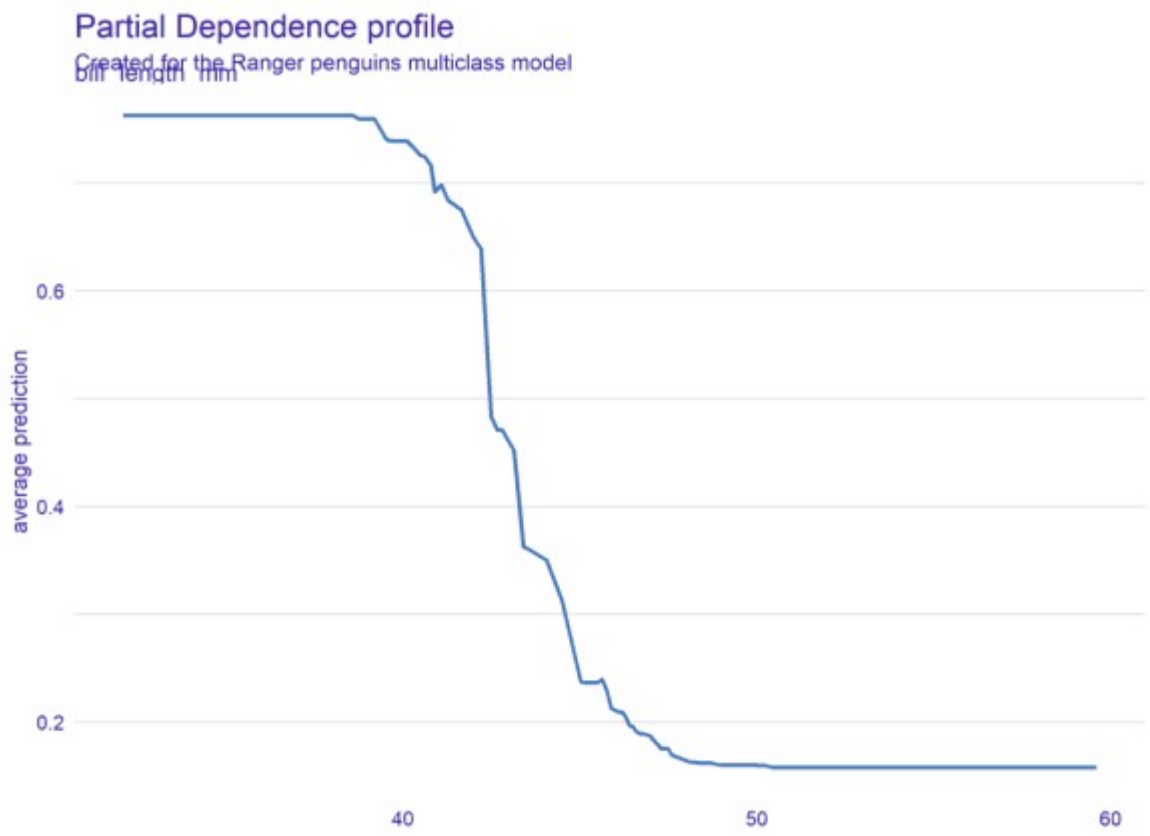


feature importance for a standard multiclass model

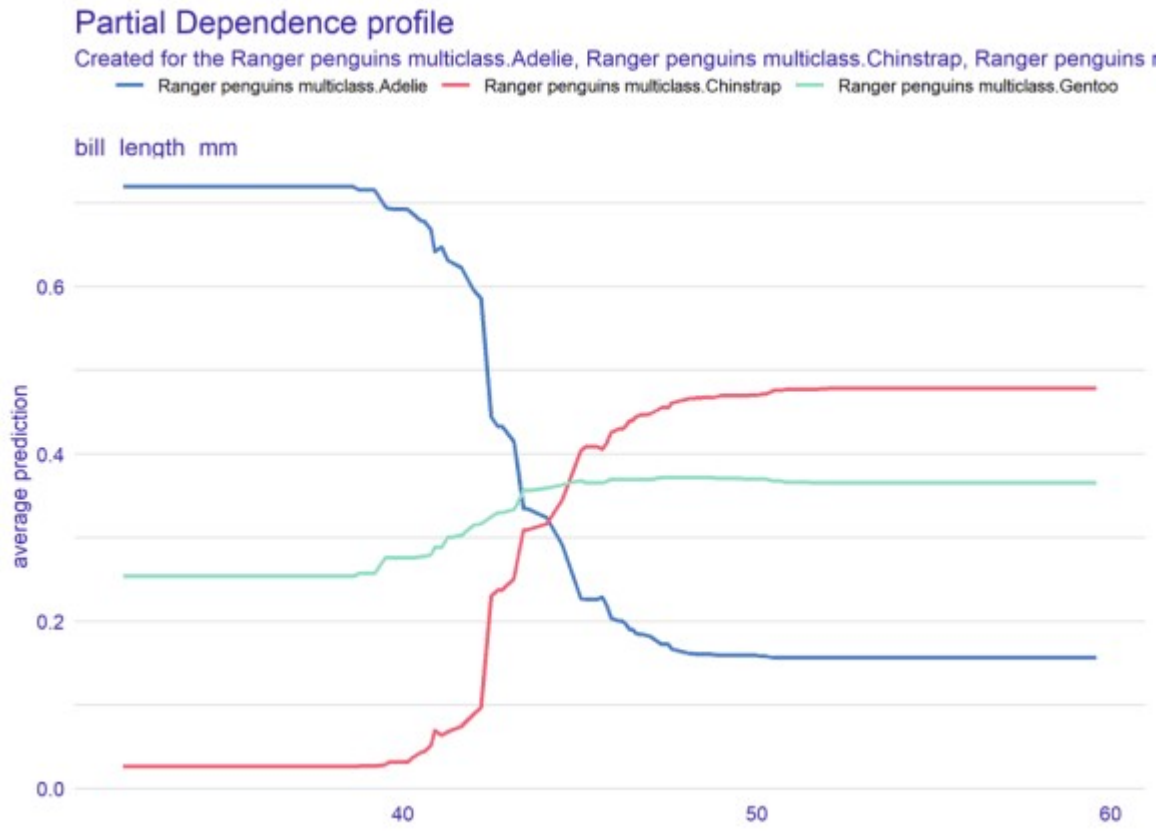
I'm quite sure you are familiar with how Predict Profile and Model Profile explanations handle multiclass models. They simply calculate explanations for each level of the y and then combine them together on th plot. But we are not only interested in profiles or breakdowns for all of the levels, right? Here comes another field when a new parameter can be utilized, we can simply choose which parameter should be

included.

```
pdp_one_vs_others <- model_profile(
    explainer = explain_multiclass_one_vs_others,
    variables = "bill_length_mm")
pdp <- model_profile(
    explainer = explain_multiclass,
    variables = "bill_length_mm")
plot(pdp_one_vs_others)
plot(pdp)
```

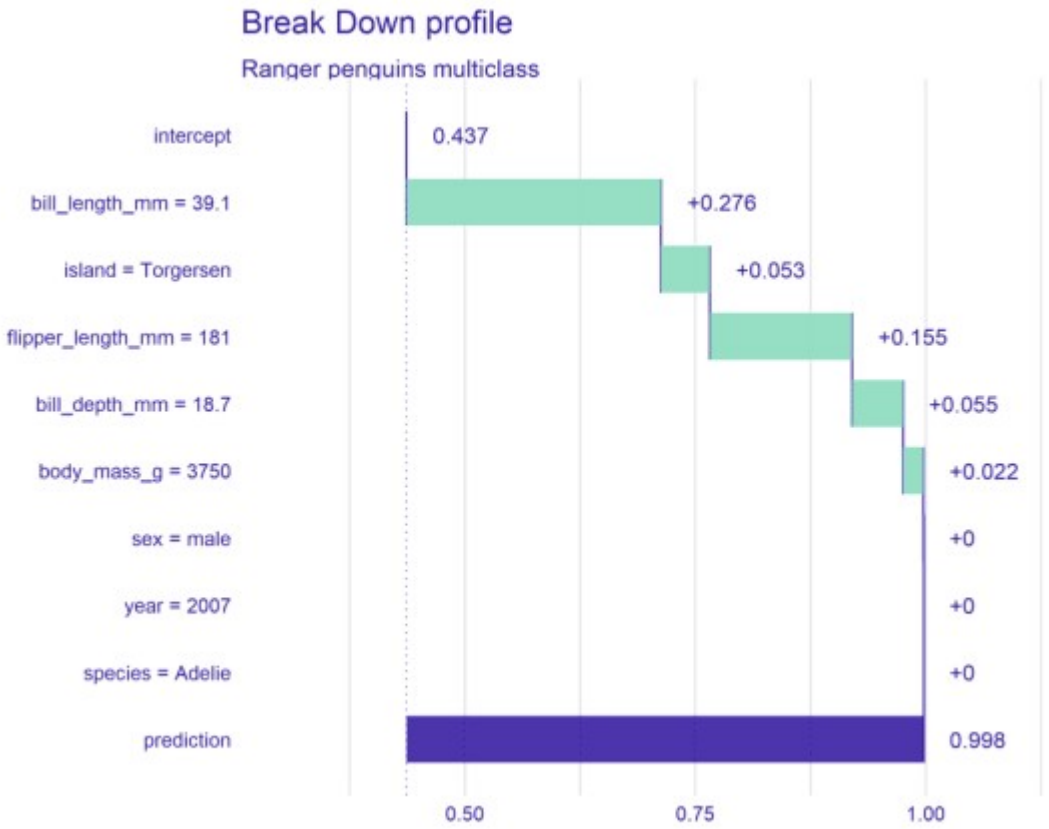


pdp feature importance for a multiclass model with chosen adelie variable

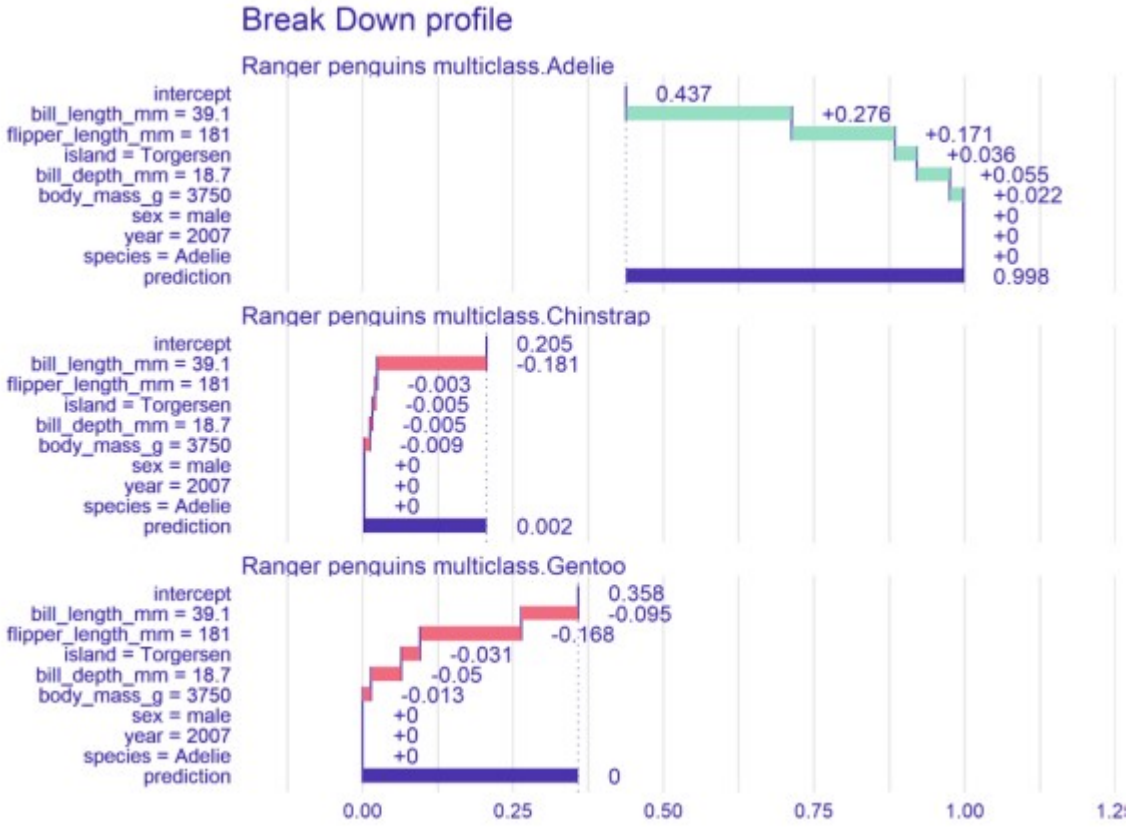


pdp feature importance for a standard multiclass model

```
bd_one_vs_others <- predict_parts(
    explainer = explain_multiclass_one_vs_others,
    new_observation = data_penguins[1,])
bd <- predict_parts(
    explainer = explain_multiclass,
    new_observation = data_penguins[1,])
plot(bd_one_vs_others)
plot(bd)
```



iBreakDown for a multiclass model with chosen adelie variable



iBreakDown for a standard multiclass model

Summary

That will be enough for today! I hope you are as excited about a new feature as I am.…