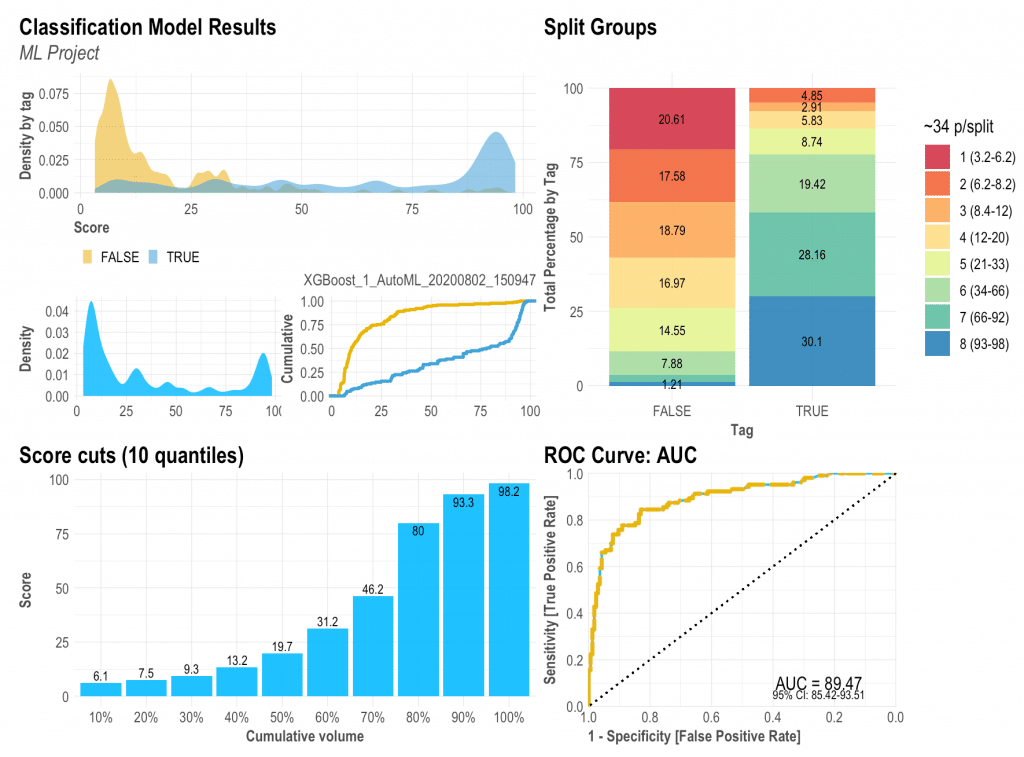
To improve the process selecting the best models and inspecting the results with visualizations, I have created some functions included in the [lares library](https://github.com/laresbernardo/lares) that boost the task. This post is focused on classification models, but the main function (mplot\_full),

Before we start, let me show you the final outcome so you know what we are trying to achieve here with just a simple line of R code:  
[](https://datascienceplus.com/wp-content/uploads/2018/07/Machine-Learning-Results-in-R-one-plot-to-rule-them-all-Part-1-Classification-Models.png)

Now that (hopefully) you are excited with the outcome, let’s install the lares library so we can replicate all the visualizations:

install.packages('lares')

Remember that all of this post’s functions have their respective documentation which will give you more details on the arguments you may set as inputs. To access them, just run ?function\_name. To train a model as I did, you may use the h2o\_automl() function to quickly have a good predictive model in seconds.

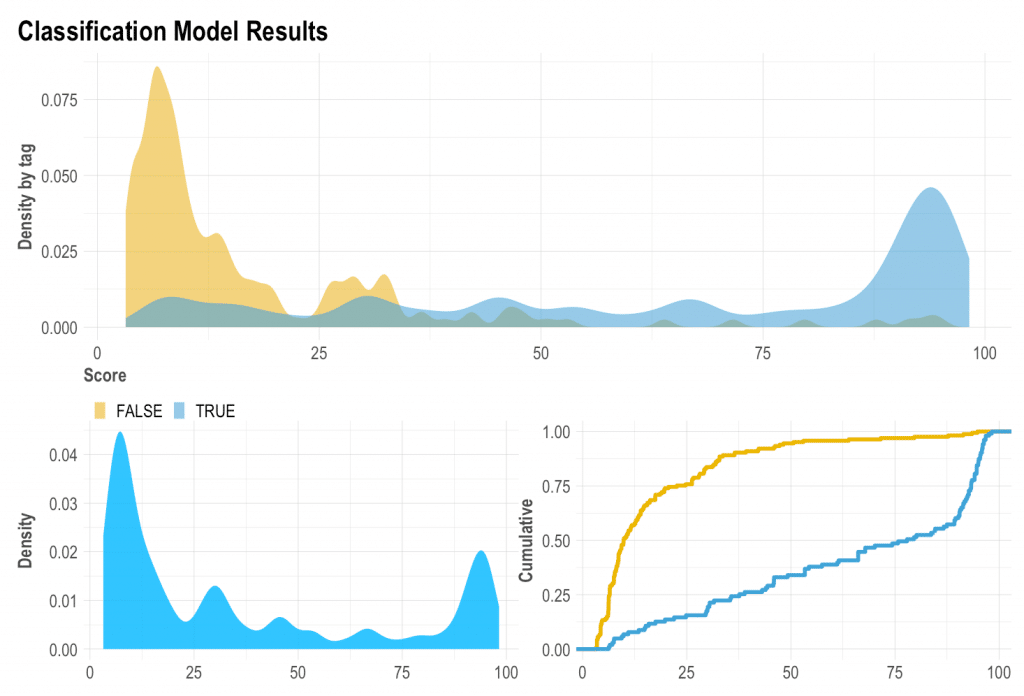
**The results object**

To create the visualizations we need to have at least the **labels and scores for each categorical value**. There are some arguments that will let you customize stuff: your project’s name to be added in titles, thresholds for multi-categorical confusion matrix plot, number of splits, highlights, captions, etc.

So, when asked to provide ‘tag’ or ‘label’ for the functions, it would be your classifier’s **real labels**. When asked to provide the ‘score’, you should input the model’s result [continuous] **values**, results of the prediction.

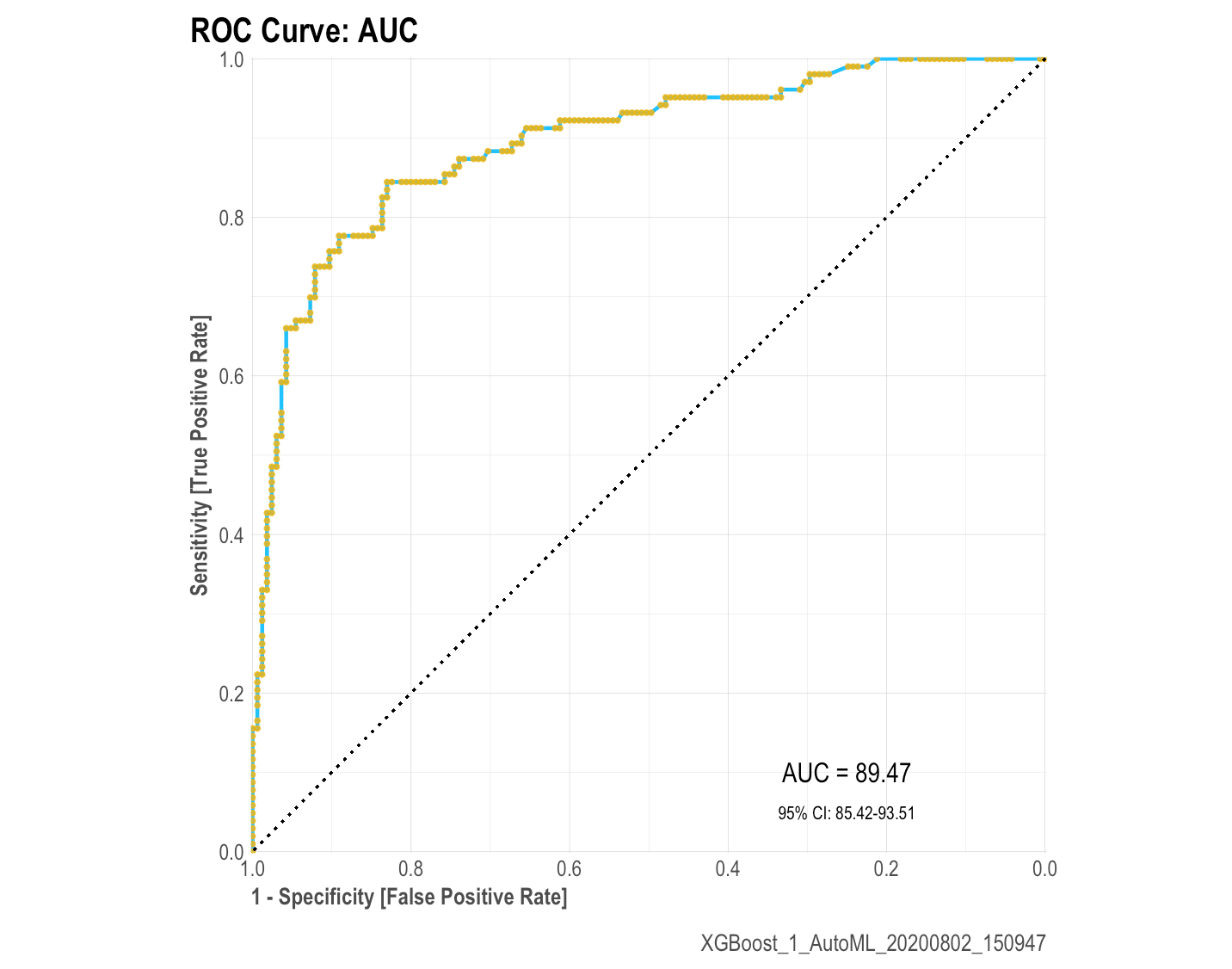
**Density Plot: mplot\_density(label, score)**

I have always given importance to the density plot because it gives us visual information on skewness, distribution, and our model’s facility to distinguish each class. Here we can see how the model has distributed both our categories, our whole test set, and the cumulative of each category (the more separate, the better).

[](https://datascienceplus.com/wp-content/uploads/2018/07/Density-Plot.png)

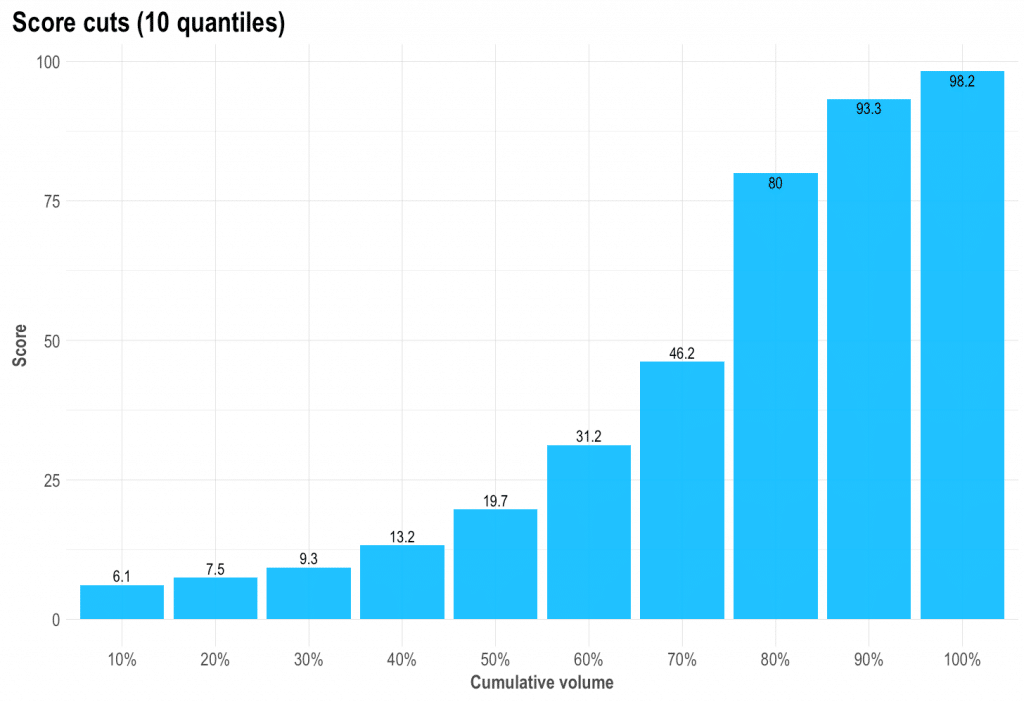
**ROC Curve: mplot\_roc(label, score)**

The ROC curve will give us an idea of how our model is performing with our test set. You should know by now that if the AUC is close to 50% then the model is as good as a random selector; on the other hand, if the AUC is near 100% then you have a “perfect model” (wanting or not, you must have been giving the model the answer this whole time!). So it is always good to check this plot and check that we are getting a reasonable Area Under the Curve with a nice and closed 95% confidence range.

[](https://datascienceplus.com/wp-content/uploads/2018/07/roc-curve.png)

**Cuts by quantile: mplot\_cuts(score)**

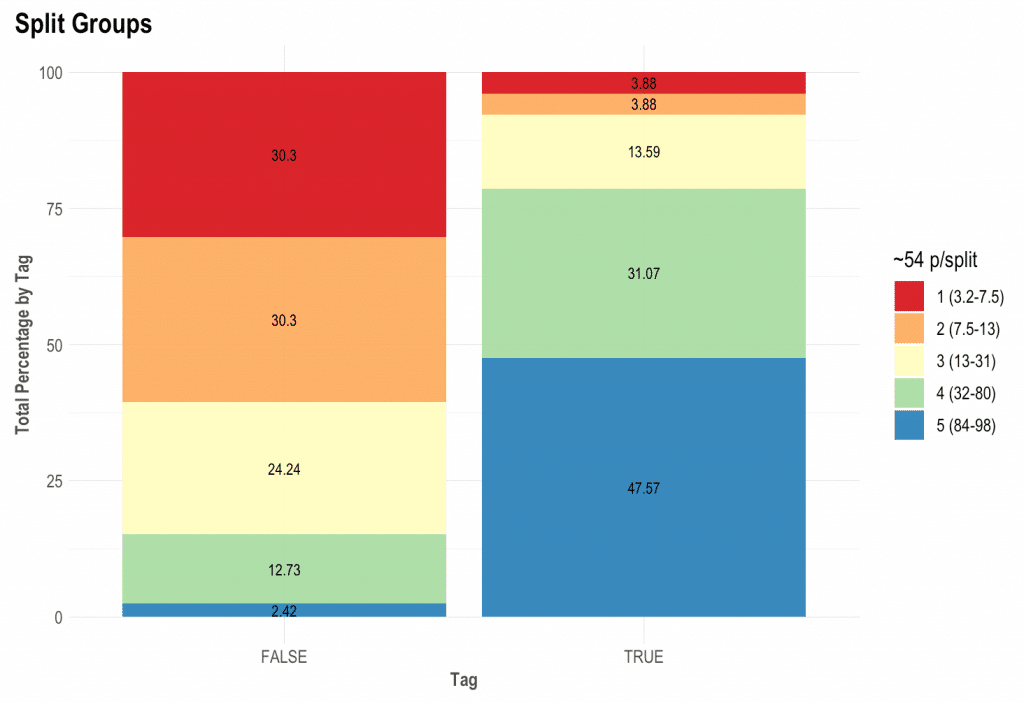
If we’d have to cut the score in n equal-sized buckets, what would the score cuts be? Is the result like a ladder (as it should), or a huge wall, or a valley? Is our score distribution lineal and easy to split?

[](https://datascienceplus.com/wp-content/uploads/2018/07/Quantiles.png)

**Split and compare quantiles: mplot\_splits(label, score)**

This parameter is the easiest to sell to the C-level guys. “Did you know that with this model, if we chop the worst 20% of leads we would have avoided 60% of the frauds and only lose 8% of our sales?” That’s what this plot will give you.

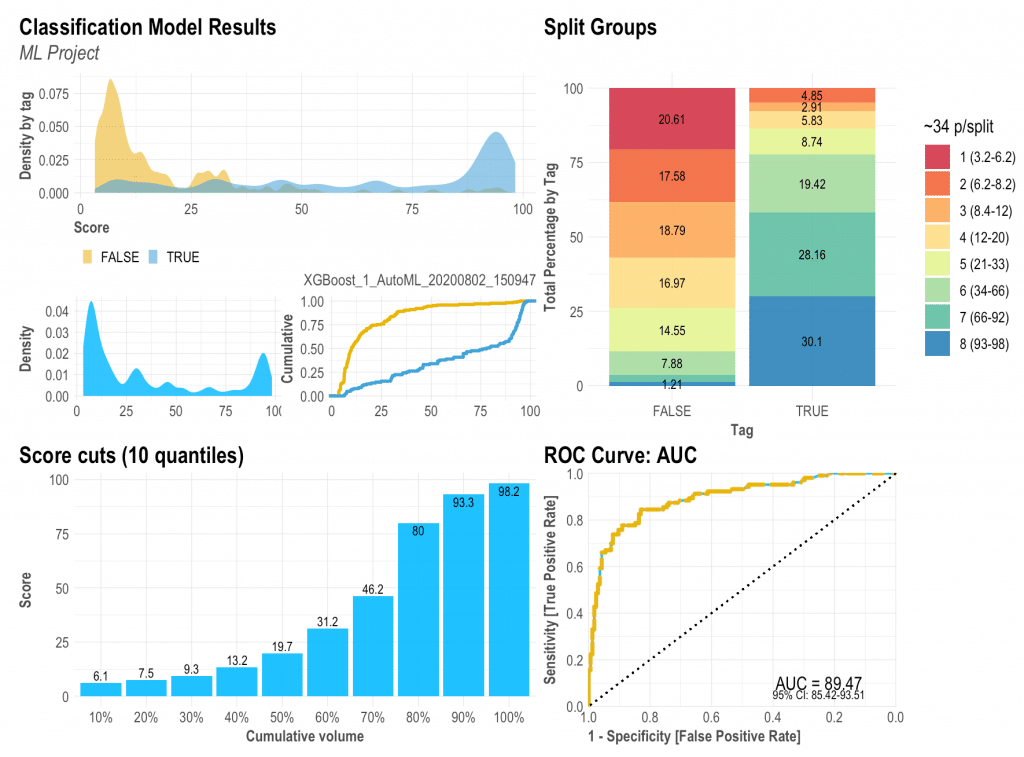
The math behind the plot might be a bit foggy for some readers so let me try and explain further: if you sort from the lowest to the highest score all your observations / people / leads, then you can literally, for instance, select the top 5 or bottom 15% or so. What we do now is split all those “ranked” rows into similar-sized-buckets to get the best bucket, the second-best one, and so on. Then, if you split all the “Goods” and the “Bads” into two columns, keeping their buckets’ colours, we still have it sorted and separated, right? To conclude, if you’d say that the worst 20% cases (all from the same worst colour and bucket) were to take an action, then how many of each label would that represent on your test set? There you go!

[](https://datascienceplus.com/wp-content/uploads/2018/07/mplot_splits.png)

**Finally, let’s plot our results: mplot\_full(label, score)**

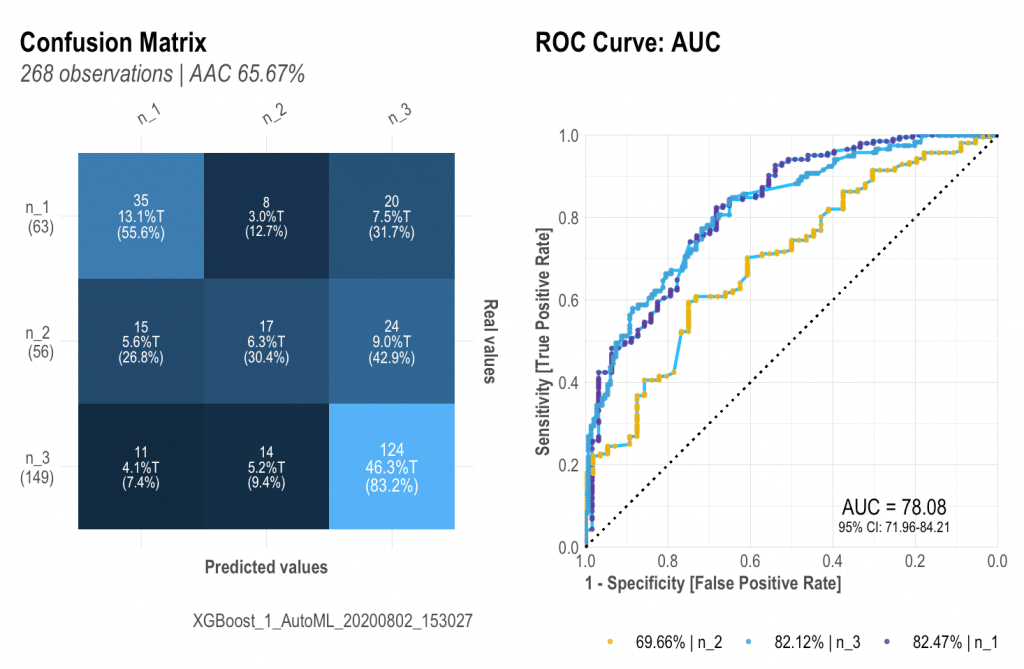
Once we have defined these functions above, we can create a new one that will bring everything together into one single plot. If you pay attention to the variables needed to create this dashboard you would notice it actually only needs two: the label or tag, and the score. You can customize the splits for the upper right plot, set a subtitle, define the model’s name, save it in a new folder, change the image’s name.

This dashboard will give us almost everything we need to visually evaluate our model’s performance into the test set.

[](https://datascienceplus.com/wp-content/uploads/2018/07/Machine-Learning-Results-in-R-one-plot-to-rule-them-all-Part-1-Classification-Models.png)

**Multicategorical (+2 labels)**

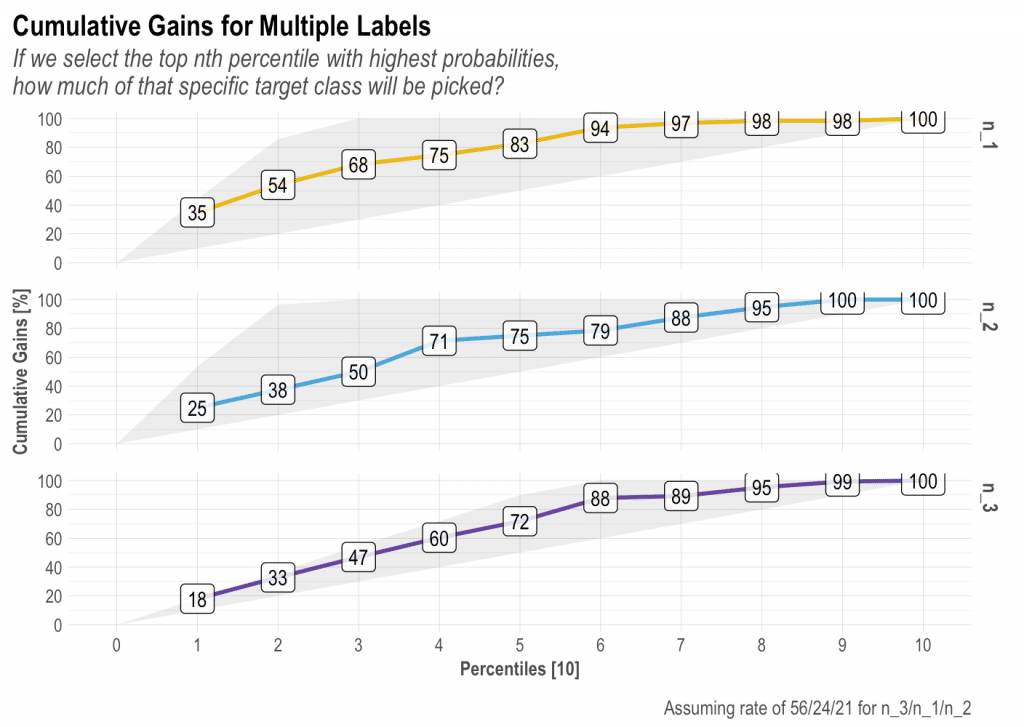
You can also use these functions for classifications non-binary models (more than 2 possible labels). Use the multis argument to pass each of your labels score. If you use the h20\_automl() function to train your models, you can set the argument as follows: multis = subset(your\_dataframe, select = -c(tag, score)

[](https://datascienceplus.com/wp-content/uploads/2018/07/Captura-de-Pantalla-2020-08-02-a-las-3.30.52-p.-m..png)

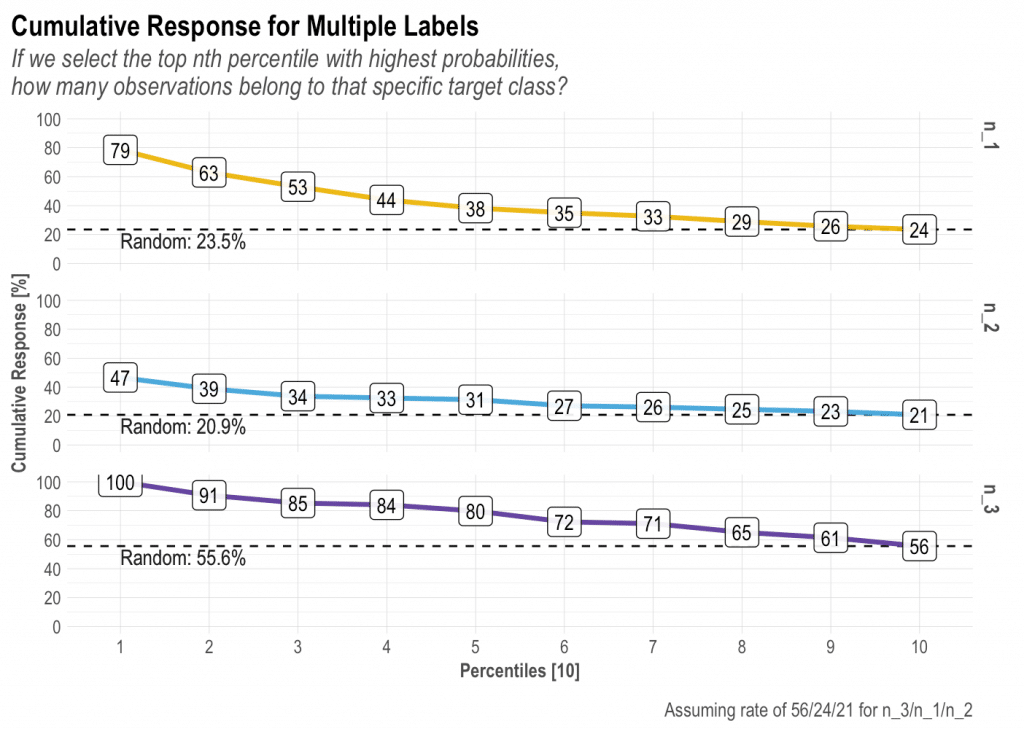
**BONUS 1: Gain and Response (2 or more labels)**

There is so much we can talk about these two plots, but I think I’ll leave that to a dedicated new post. In short, I think these are the most landed and value-centered visualizations to pitch our model results. Let’ check the plots for a 3-labels classification model:

**mplot\_gain(label, score, multis)**

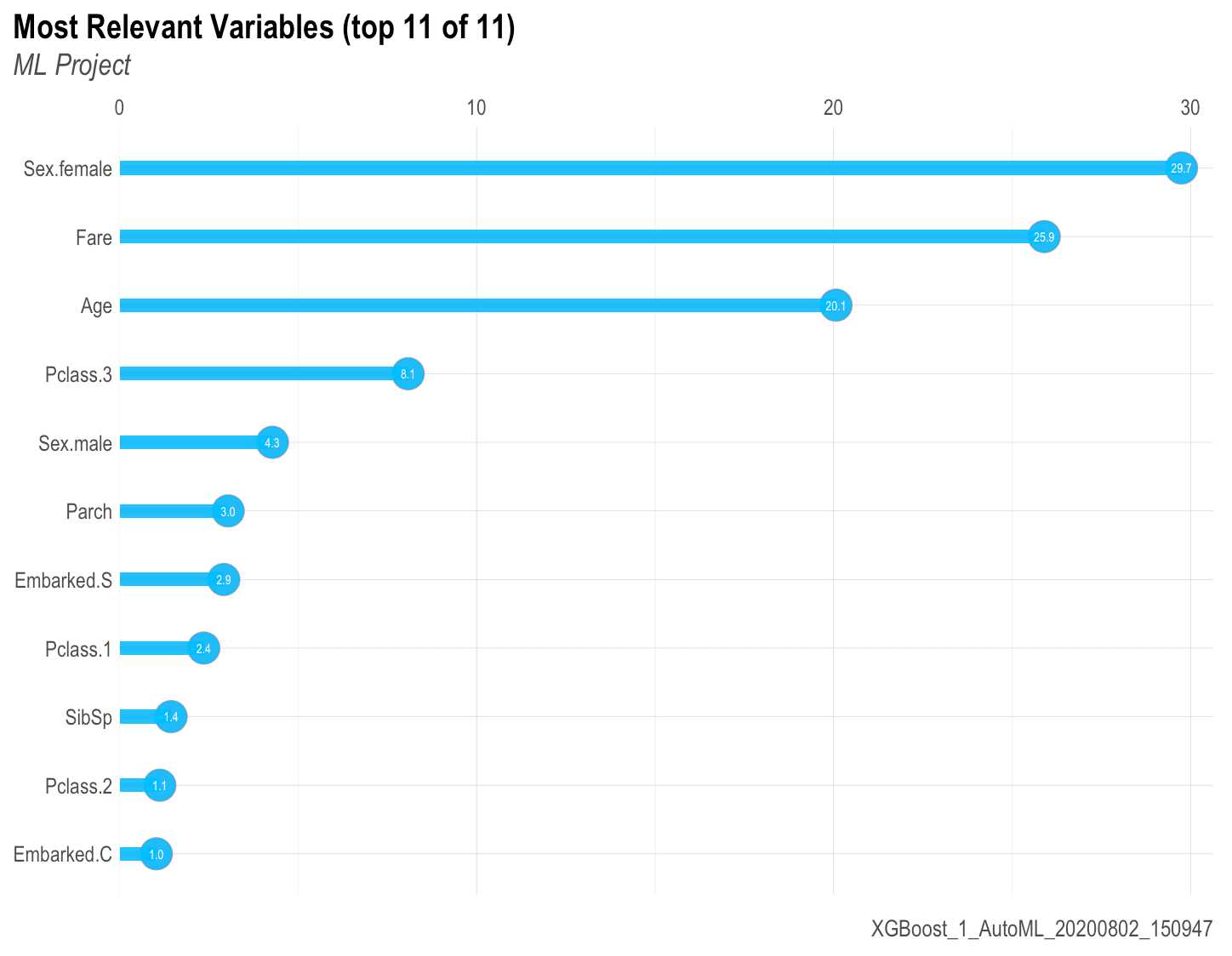
[](https://datascienceplus.com/wp-content/uploads/2018/07/Captura-de-Pantalla-2020-08-02-a-las-3.51.05-p.-m..png)

**mplot\_response(label, score, multis)**

[](https://datascienceplus.com/wp-content/uploads/2018/07/Captura-de-Pantalla-2020-08-02-a-las-3.50.48-p.-m..png)

**BONUS 2: Variables Importance: mplot\_importance(var, imp)**

If you are working with a ML algorithm that let’s you see the importance of each variable, you can use the following function to see the results:

[](https://datascienceplus.com/wp-content/uploads/2018/07/feature-importances.png)

Hope you guys enjoyed this post and any further comments or suggestions are more than welcome. Not a programmer here but I surely enjoy sharing my code and ideas!