Report HW6

Meta info

I'm using MagicTelescope.csv and LGB models and PyTorch, with 0.33 train/test split.

Calculate Permutation-based Variable Importance for the selected model.

To calculate permutation-based variable importance for the selected model we used the `permutation_importance` function from the `sklearn.inspection` module. As we can see fAlpha is the most important feature. Followed by fWidth and fSize.

Results are presented in the image bellow:

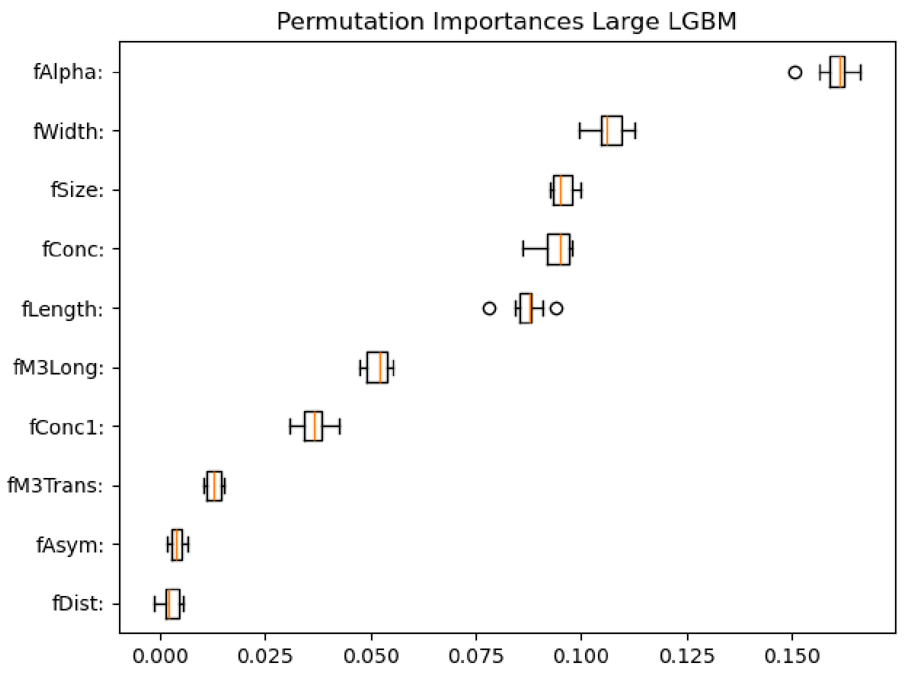

Figure 1: PVI for default LGBM

Train three more candidate models (different variable transformations, different model architectures, hyper-parameters) and compare their rankings of important features using PVI. What are the differences? Why?

Here are PVI results for three more candidate models:

- Smaller LGBM model than used in (1)

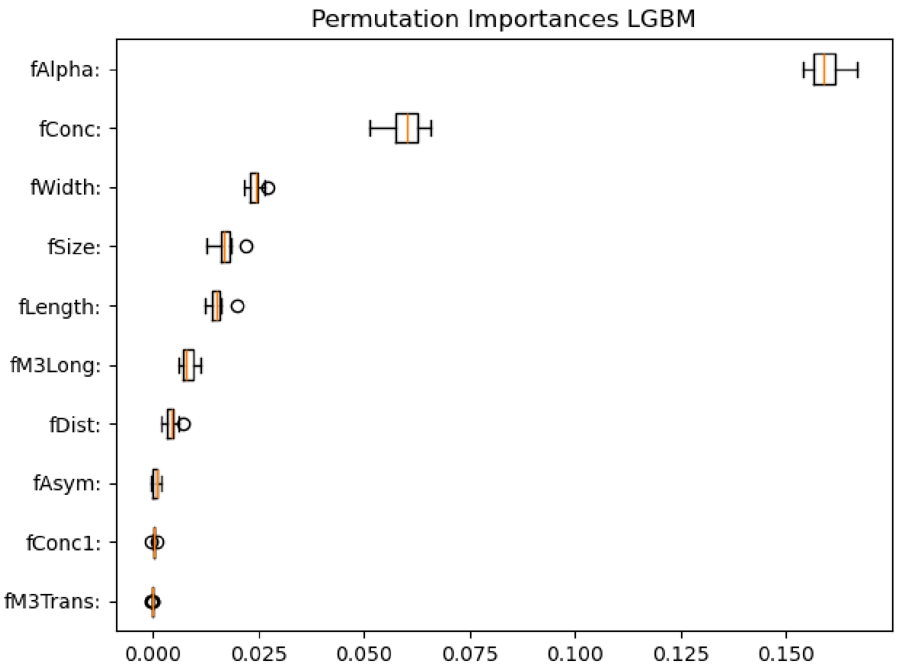

Figure 2: PVI for small LGBM

- Small neural network
- Large neural network (with batch normalization)

From this we can see: That all 4 models put the most importance on the 'fAlpha' feature. Something intresting is that both NN models have almost the exact same order of importance of features. Wheras LGBM models don't have such correlation of ordering. Now we can theorize how NN and TreeEnsamble models have a different ways of traveling the loss function. I think this can be also seen in other exericises where NN had much more consitent and more explainable results.

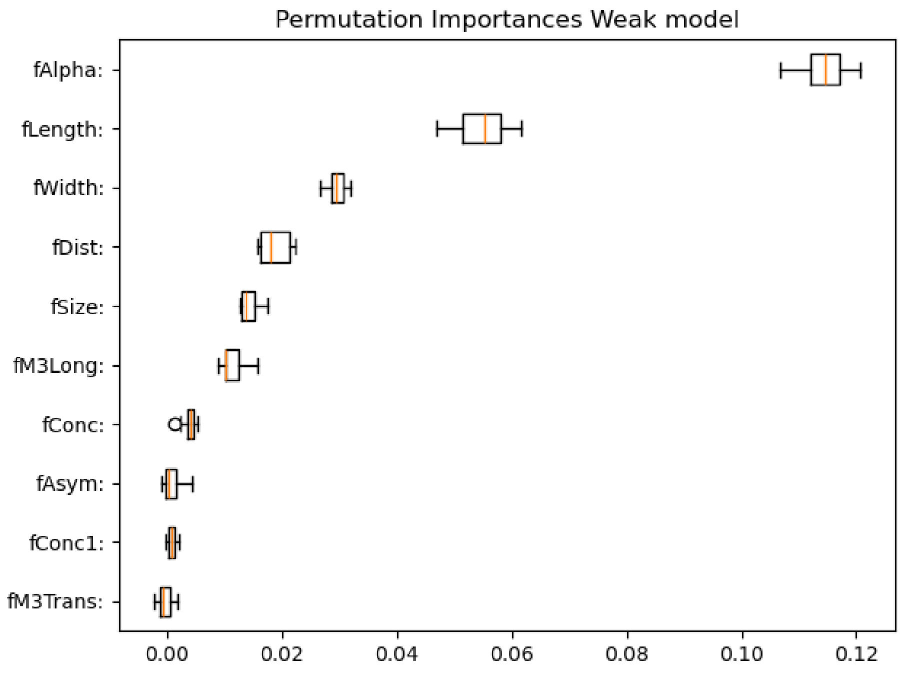

Figure 3: PVI for small DNN

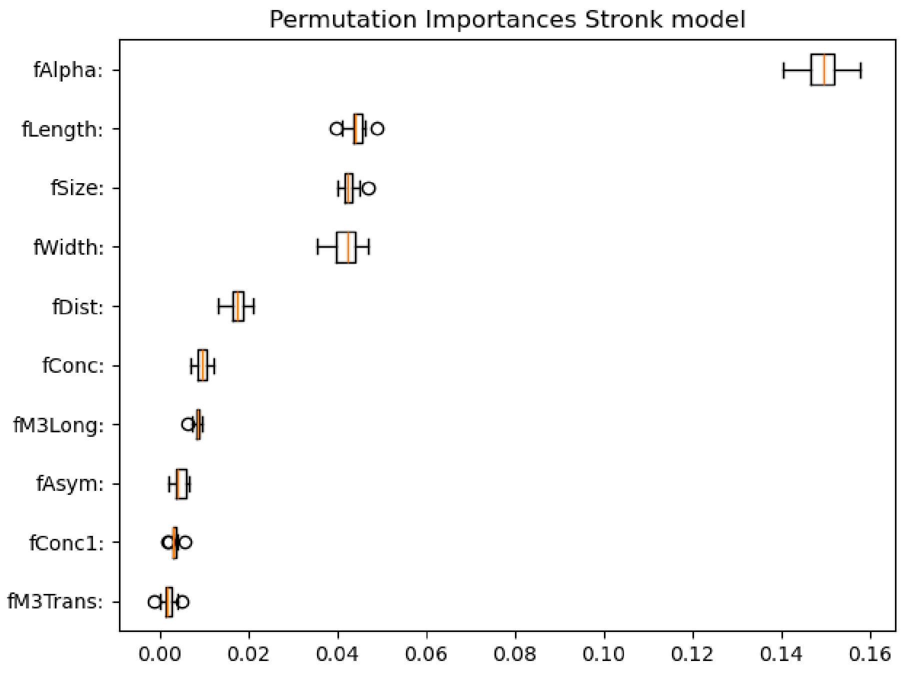

Figure 4: PVI for large DNN

For the tree-based model from (1), compare PVI with:

- A: the traditional feature importance measures for trees: Gini impurity etc.; what is implemented in a given library: see e.g. the feature_importances_ attribute in xgboost and sklearn.

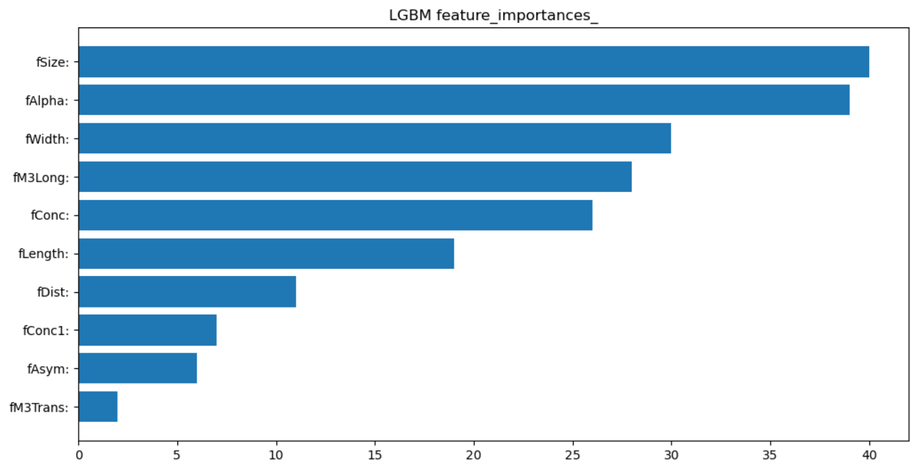

Figure 5: LGBM feature_importances_

I think feature importances comming from LGBM `feature_importances_` atribute might not be very accurate as they don't consider fAlpha feature as the most important one. This is highly at odds with the results from PVI acros all models. At the same time fAlpha is the second most important feature so those values can't be that bad. Something that I also this is intresting is the shape of the graph. It looks more convex than all other graphs which look concave. This suggests that this method is more liberal in assigning importance to features.

- B: [in Python] SHAP variable importance based on the TreeSHAP algorithm available in the shap package.

SHAP values are much more similar to PVI values than feature importances. Top 4 features are the same in both methods, whith slight reordering but I think this can be attributed to the fact that SHAP values are calculated for each sample and then averaged. Which can create some noise in the results.

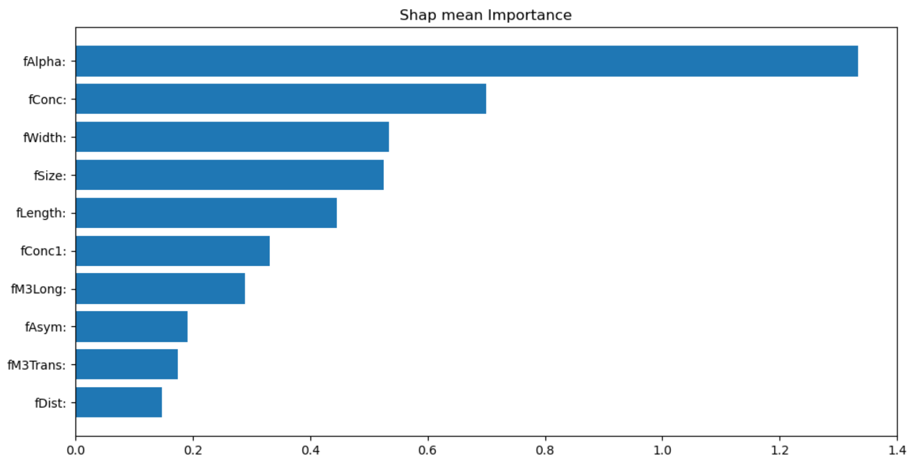

Figure 6: SHAP variable importance