Model summary

Competition: Trends Neuroimaging

Team: Neo Cortex & N. Gin

Private LB score: 0.15612

Private LB place: 1st

Names: Nikita Churkin, Dmitry Simakov

Location: Moscow, Russian Federation

University: Higher School of Economics (HSE)

1. Summary

Our solution includes making feature bundles of 3D fMRI data and FNC matrices, preprocessing and postprocessing, training a lot of models and creating diverse ensemble. Final solution is averaging of 7 Kfold seeds but computing 1 seed is sufficient to get the 1st place score.
Our validation scheme is simple Kfold for all targets: it correlates with public/private LB perfectly.

2. Data preprocessing steps

Important preprocessing step: apply additive bias to different columns in test set to make it closer to train set. Biases (offsets) calculation involves:

- Minimization (we use differential evolution in scipy.optimize) of Kolmogorov-Smirnov statistic between `train[col]` and `test[col]+ b`.
- Biases for both the test as a whole and only for Site2.
- The same logic applied to the model predictions, but between Site2 and not Site2 observations.

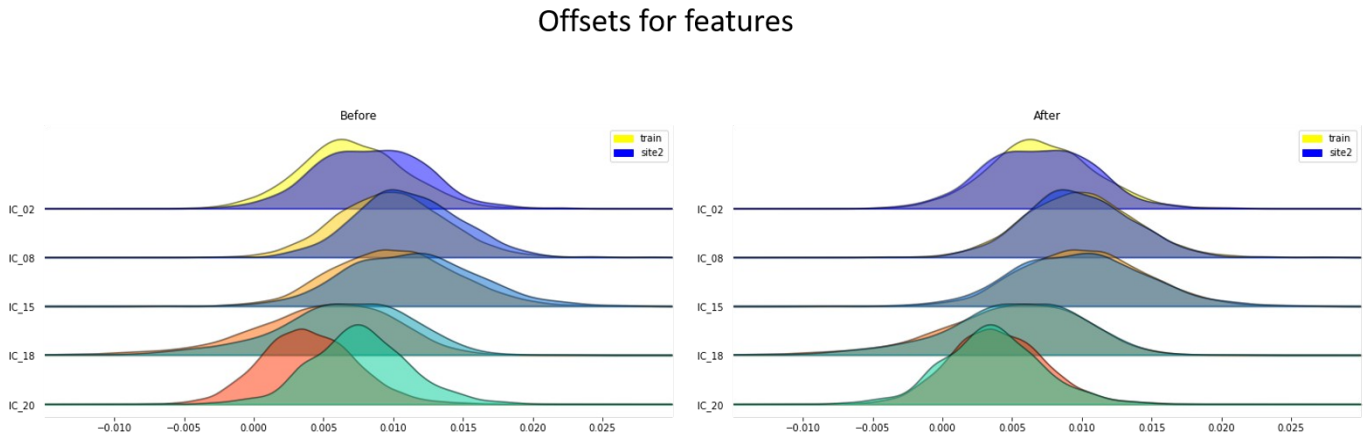

Figure 1. Example of distributions matching.

3. Feature engineering

Main constructed feature block is PCA features:

- Incremental PCA on fMRI data with n_components 200, batch-size 200.
- Channels were splitted in groups by 10 and flattened inside them (6 groups in total).
- All training took 9 hours for 24 threads processor and 64gb RAM.
- As a result, 1200 we got PCA features.

Second important feature pack is dictionary-learning (DL features):

- Optimal parameters: n-components 100, batch-size 100 and n-iters 10.

Additional features (AGG features):

- Compute simple statistics, like mean, std, and quantiles inside each feature of 3D scans and similar stats for FNC matrices.

The most important features were IC components from loading file and PCA from 3d fMRI scans.

4. Feature selection

Classic selection approaches (forward/backward selection, rank by permutation importance, etc.) led to CV overfitting so we used selection by blocks: use feature only if corresponding feature block improves both CV and LB.
Only removed feature is IC_20: it has the biggest discrepancy between train and test; Removing of IC_20 is good for CV and LB score.

5. Training methods

Site2 classifier

Site2/Site1 classifier was designed to figure out more (hidden) Site2 objects. New Site2 objects are involved in bias computation and postprocessing.
Classifier is made of following steps:

- Fit on raw loading, fnc features and pca features
- `StandardScaler()` for train + test data (it boosts score by ~0.02 roc-auc)
- Regression `ElasticNet()` as model
- Pseudo-labels of test set inside training fold
- Retrained model with pseudo-labels

Final metrics:

- 0.973 roc-auc
- 0.76 F1-score (subject to site2 class)
- ~1400 new site2, with 200-400 False Positive errors

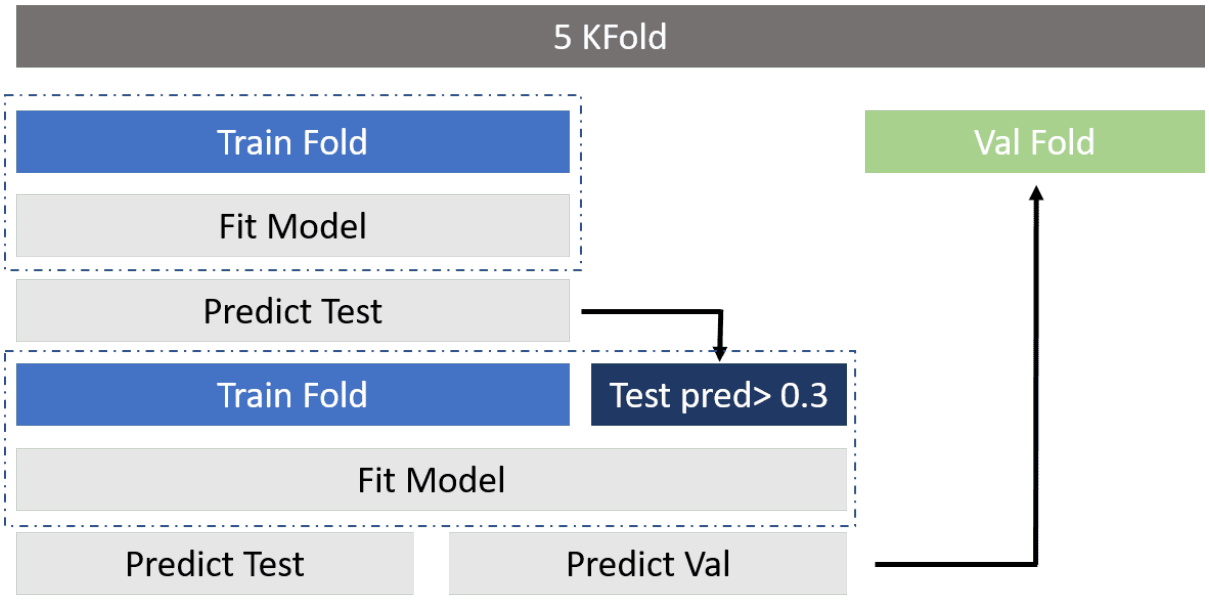

Figure 2. Site2/Site1 classifier training scheme.

1st layer of stacking - scores

For domain1_var2 median prediction was used: it was extremely hard to come up with better models.
Our team prepared so called “Scores” for every target and every feature block (like FNC features, PCA features, image statistics, etc.). Scores are just OOF predictions on the data of only one source at time.

Making of scores allowed to effectively use more sophisticated and/or slow algorithms (like MLP or RGF) on top of, in some way, all data at once (escape from high dimensionality).

Scores were built by blending some fitted linear models (like Ridge or OMP) and RGF. Different scores for different labels include different bundles of models.

We designed 5 separate scores: FNC, FNC aggregations, image stats, PCA features, DL features.

Here we list used models for all pairs **target** / **feature bundle**.

Age / FNC

```
RGFRegressor(max_leaf=1000, reg_depth=5, normalize=True),

ElasticNet(alpha=0.05, l1_ratio=0.5, random_state=0),

BayesianRidge(),

HuberRegressor(epsilon=2.5, alpha=1),
```

```
OrthogonalMatchingPursuit(n_nonzero_coefs=300)
```

Age / FNC aggregations

```
RGFRegressor(max_leaf=1000, reg_depth=5, min_samples_leaf=100, normalize=True),
ElasticNet(alpha=0.05, l1_ratio=0.3, random_state=0),
HuberRegressor(epsilon=2.5, alpha=1)
```

Age / PCA features

```
RGFRegressor(max_leaf=1000, reg_depth=5, min_samples_leaf=100, normalize=True),
ElasticNet(alpha=0.2, l1_ratio=0.2, random_state=0),
BayesianRidge(),
OrthogonalMatchingPursuit()
```

Age / image statistics

```
RGFRegressor(max_leaf=1000, reg_depth=5, min_samples_leaf=100, normalize=True),
ElasticNet(alpha=0.2, l1_ratio=0.2, random_state=0),
BayesianRidge(),
OrthogonalMatchingPursuit()
```

Age / DL features

```
RGFRegressor(max_leaf=1000, reg_depth=5, min_samples_leaf=100, normalize=True),
ElasticNet(alpha=0.2, l1_ratio=0.2, random_state=0),
BayesianRidge()
```

Domain1_var1 / FNC

```
ElasticNet(alpha=0.05, l1_ratio=0.5, random_state=0),
BayesianRidge()
```

Domain1_var1 / FNC aggregations

```
ElasticNet(alpha=0.05, l1_ratio=0.3, random_state=0),
HuberRegressor(epsilon=2.5, alpha=1)
```

Domain1_var1 / PCA features

```
ElasticNet(alpha=0.2, l1_ratio=0.2, random_state=0),
```

```
BayesianRidge()
```

Domain1_var1 / image statistics

```
ElasticNet(alpha=0.2, l1_ratio=0.2, random_state=0),
BayesianRidge()
```

Domain1_var1 / DL features

```
ElasticNet(alpha=0.2, l1_ratio=0.2, random_state=0),
BayesianRidge()
```

Domain2_var1 / FNC

```
ElasticNet(alpha=0.05, l1_ratio=0.5, random_state=0),
BayesianRidge()
```

Domain2_var1 / FNC aggregations

```
RGFRegressor(max_leaf=1000, reg_depth=5, min_samples_leaf=100, normalize=True),
ElasticNet(alpha=0.05, l1_ratio=0.3, random_state=0),
HuberRegressor(epsilon=2.5, alpha=1)
```

Domain2_var1 / PCA features

```
ElasticNet(alpha=0.2, l1_ratio=0.2, random_state=0),
BayesianRidge(),
OrthogonalMatchingPursuit()
```

Domain2_var1 / image statistics

```
ElasticNet(alpha=0.2, l1_ratio=0.2, random_state=0),
BayesianRidge(),
OrthogonalMatchingPursuit()
```

Domain2_var1 / DL features

```
ElasticNet(alpha=0.2, l1_ratio=0.2, random_state=0),
BayesianRidge(),
OrthogonalMatchingPursuit()
```

Domain2_var2 / FNC

```
ElasticNet(alpha=0.05, l1_ratio=0.5, random_state=0),
```

```
BayesianRidge(),
HuberRegressor(epsilon=2.5, alpha=1),
OrthogonalMatchingPursuit(n_nonzero_coefs=300)
```

Domain2_var2 / FNC aggregations

```
RGFRegressor(max_leaf=1000, reg_depth=5, min_samples_leaf=100, normalize=True),
ElasticNet(alpha=0.05, l1_ratio=0.3, random_state=0),
HuberRegressor(epsilon=2.5, alpha=1)
```

Domain2_var2 / PCA features

```
ElasticNet(alpha=0.2, l1_ratio=0.2, random_state=0),
BayesianRidge(),
OrthogonalMatchingPursuit()
```

Domain2_var2 / image statistics

```
ElasticNet(alpha=0.2, l1_ratio=0.2, random_state=0),
BayesianRidge(),
OrthogonalMatchingPursuit()
```

Domain2_var2 / DL features

```
ElasticNet(alpha=0.2, l1_ratio=0.2, random_state=0),
BayesianRidge(),
OrthogonalMatchingPursuit()
```

After computing every model we blend outputs with weights in [0, 1] to minimize normalized MAE (using custom differential evolution optimizer of OOF metric value).

2nd layer of stacking — learning on different datasets

Our 2nd stacking approach involved different models on different datasets (including scores) for each target (excluding d12).

- Final stacking for more “complicated” labels such domain2_var1 and domain1_var1 included only blending of basic linear models + NuSVM, KernelRidge and Gaussian Process trained on different datasets.
- Final ensemble of age and domain2_var2 contained more models (and more layers in case of age).

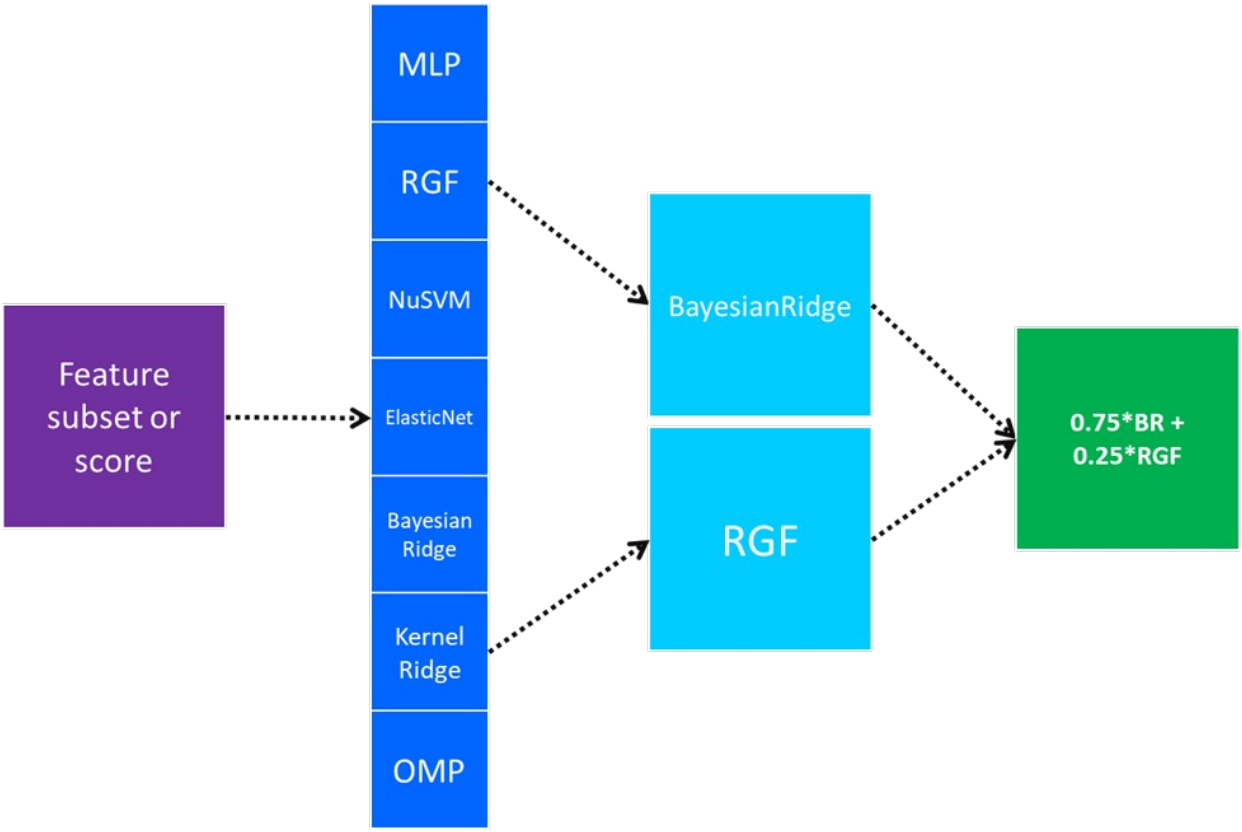

Figure 3. Simplified stacking scheme for age label.

All models involved in second layer of stacking correspond to **only one dataset**.

For example, MLP on the figure 3 is trained on raw loading features + scores, no other prepared dataset is used (so we have only one copy of MLP, RGF, etc.).

Here we list used datasets and corresponding models of second level for all targets[1].

Age

Set1 := (loading — IC_20) + fnc + pca + fnc_agg + im_st

`StandardScaler()` is used for preprocessing

Set2 := (loading — IC_20) + pca

`StandardScaler()` is used for preprocessing

Pca columns are multiplied by 0.1 after scaling

Set3 := (loading — IC_20) + scores

`StandardScaler()` is used for preprocessing

1 We name (count) datasets/models from 1.

Set4 := (loading — IC_20) + fnc + dl + fnc_agg + im_st

`StandardScaler()` is used for preprocessing

Model1 := `MLPRegressor(activation='tanh', random_state=0)` trained on Set3

Model2 := `RGFRegressor(max_leaf=1500, loss='Abs')` trained on Set3

Model3 := `NuSVR(C=10, nu=0.4, kernel='rbf')` trained on Set2

Model4 := `BayesianRidge()` trained on Set1

Model5 := `OrthogonalMatchingPursuitCV()` trained on Set1

Model6 := `ElasticNet(alpha=0.5, l1_ratio=0.7, random_state=0)` trained on Set2

Model7 := `KernelRidge(kernel='poly', alpha=0.5)` trained on Set2

Domain1_var1

Set1 := (loading — IC_20) + fnc + pca

`StandardScaler()` is used for preprocessing

Set2 := (loading — IC_20) + pca

`MinMaxScaler()` is used for preprocessing

Pca columns are multiplied by 0.2 after scaling

Set3 := (loading — IC_20) + scores

`StandardScaler()` is used for preprocessing

Model1 := `GaussianProcessRegressor(DotProduct(),random_state=0)` trained on Set3

Model2 := `NuSVR(C=5, kernel='rbf')` trained on Set3

Model3 := `NuSVR(C=5, kernel='rbf')` trained on Set2

Model4 := `OrthogonalMatchingPursuitCV()` trained on Set1

Model5 := `KernelRidge(kernel='poly', degree=2, alpha=10)` trained on Set1

Domain2_var1

Set1 := (loading — IC_20) + fnc + pca0[2]

`StandardScaler()` is used for preprocessing

Set2 := (loading — IC_20) + pca0

`StandardScaler()` is used for preprocessing

Pca0 columns are multiplied by 0.2 after scaling

Set3 := (loading — IC_20) + scores

`StandardScaler()` is used for preprocessing

Model1 := `GaussianProcessRegressor(DotProduct(),random_state=0)` trained on Set3

Model2 := `NuSVR(C=5, kernel='rbf')` trained on Set3

Model3 := `NuSVR(C=5, kernel='rbf')` trained on Set2

Model4 := `Lasso(alpha=0.1, random_state=0)` trained on Set1

Model5 := `BaggingRegressor(Ridge(alpha=1), n_estimators=100, max_samples=0.2, max_features=0.2, random_state=0)` trained on Set1

Domain2_var2

Set1 := (loading — IC_20) + fnc + pca

`StandardScaler()` is used for preprocessing

Set2 := (loading — IC_20) + pca

`StandardScaler()` is used for preprocessing

Pca columns are multiplied by 0.1 after scaling

Set3 := (loading — IC_20) + scores

`StandardScaler()` is used for preprocessing

2 Subset of PCA features, only 0 component.

Model1 := `GaussianProcessRegressor(DotProduct(),random_state=0)` trained on Set3

Model2 := `NuSVR(C=2, kernel='rbf')` trained on Set2

Model3 := `BayesianRidge()` trained on Set1

Model4 := `KernelRidge(kernel='poly', alpha=30)` trained on Set1

3rd layer of stacking — more models for age and blend for other labels

For the age label 2nd level models were stacked with `RGFRegressor(max_leaf=1000, reg_depth=25, verbose=False)` and `BayesianRidge()` models; their outputs were blended with 0.25 (RGF) / 0.75 (BaysianRidge) weights. The weights were set by hand to minimize risk of overfitting.

For domain1_var1, domain2_var1, domain2_var2, plain blending was used: blend outputs with weights in [0, 1] to minimize normalized MAE; use custom differential evolution optimizer of OOF metric value.

6. Interesting findings

Most features have linear relation with targets (see example on figure 4 and figure 5) and it is important to create different features to add diversity for the final ensemble.

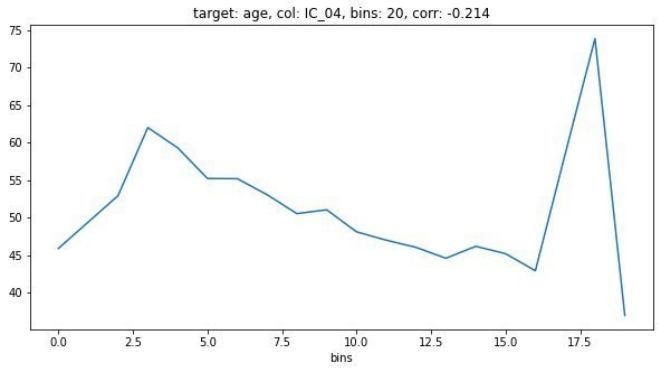

Figure 4. Mean age in 20 uniform bins for IC_04 feature

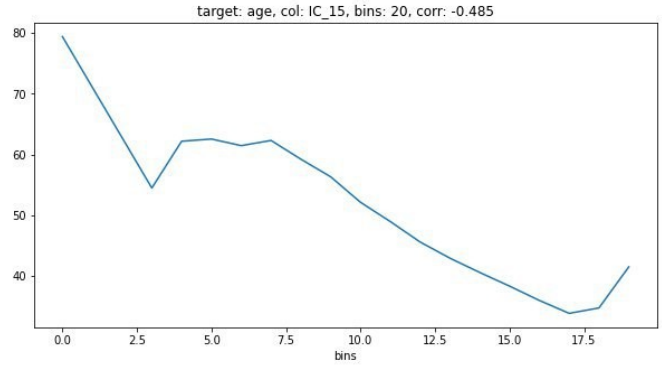

Figure 5. Mean age in 20 uniform bins for IC_15 feature

7. How to simplify the model?

First of all, you do not need 7 seeds to get good score: 1 is sufficient. Second, we belive that stacking procedure can be simplified to some extend: for example, you can remove RGF or MLP models from stack completely.

Simple ElasticNet model on raw loading and generated PCA features (~1250 features at all) is able to produce 0.15753 score that corresponds to 14th private place[3].

8. Model execution time

Final submission computation time consist of:

13 hours for PCA features creation

47 hours for DL features creation

~1 hour for other features

~2.5 hours for one seed model training and inference (final submission is blend for 7 validation seeds)

- The slowest model is RGF, it takes~50% of the training time.

- All models (7 seeds for 5 targets) require ~160gb of disk space. 450gb are required for resaved fMRI data and ~30gb for PCA/DL models

9. LB progress

Figure 6 summarizes our progress and relates important solution steps with improvements in the LB score.

3 Of course, you have to use our preprocessing and postprocessing.

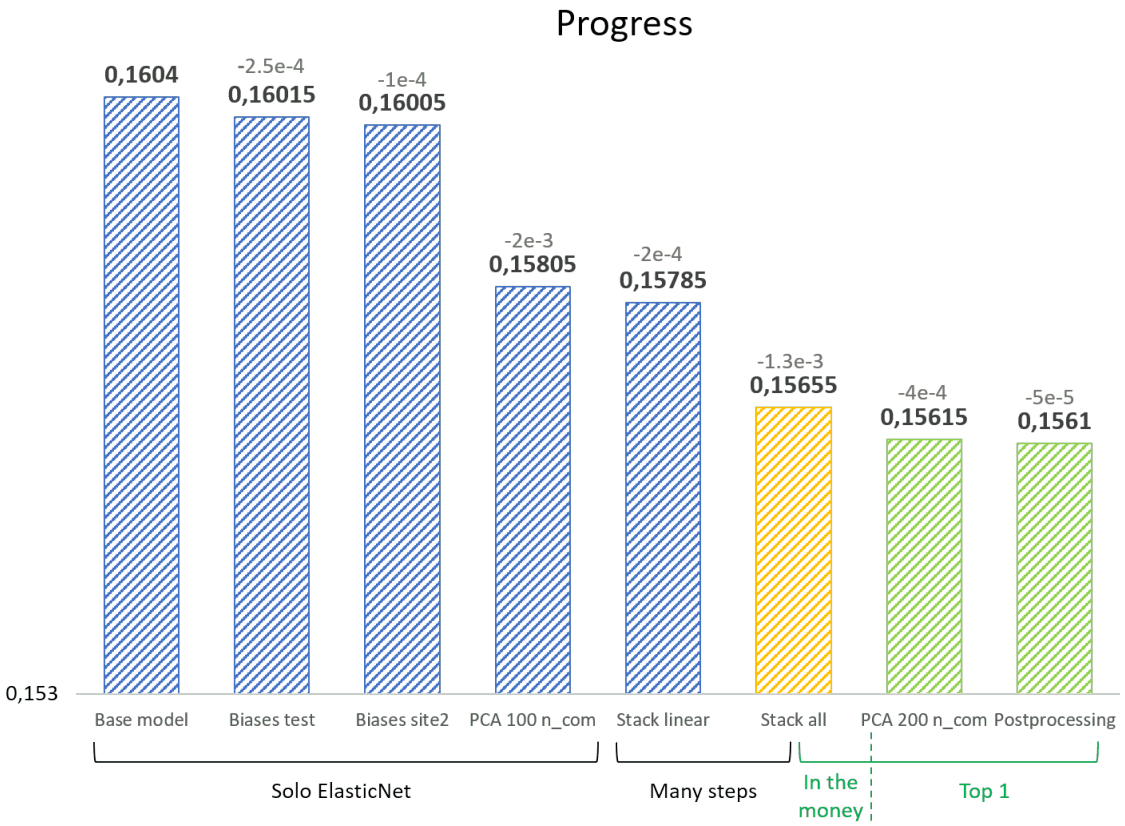

Figure 6. Competition progress chart