# Meta-Learning: NumerAi

Tanmay Singh
2021569
CSAI
Class of ‘25

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
**DELHI**

# Insights on NumerAi Dataset

Firstly, it's a tough dataset!

- The dataset is dynamic, i.e. the data is being added every week & the relationship between the features & the target values change every now and then.

- The data contains around 27 lakh entries in the training data, without any NaN values, while the validation data contains around 34 lakh entries with around 30k NaN values (which were processed).

- It is imbalanced/skewed in nature, as the target value of 0.5 (class 2) predominates the other values in the data, followed by an equal representation for class 1 & class 3 (0.25 & 0.75 respectively), followed by class 0 & class 4 (0 & 1 respectively).

```
dataset.isna().any().any()
```

```
False
```

```
dataset['target'].value_counts()
```

```
target
0.50    1373108
0.25     550995
0.75     550373
1.00     136126
0.00     135668
Name: count, dtype: int64
```

# My Approach to the NumerAi Problem

My approach towards the NumerAi prediction problem is a slight modification of the standard machine learning & meta-learning paradigm.

I divided the training data (the training parquet) into training & validation sets with an 80:20 ratio, with each having an equal number of samples per class (performed sampling to balance the skewed sets). I used them to train experts/weak classifiers (6 in total) on the training set & generated predictions on the validation set, much like in any standard machine learning paradigm.

These predictions were then neutralized to remove the influence of highly correlated features with respect to the targets.
I stacked these neutralized predictions from each expert to form the input for the meta-model, which learns a mapping from these predictions (use it as a context) to the target values.

Essentially, the ‘task’ in my approach is to train the weak classifiers in such a way that they create an enriched set of predictions, which, when combined together, act as a rich context to the meta-model, which can learn a mapping to the target values & be shown to generalize from this little set (1/5th of the training data) on the meta-testing set (the validation parquet).

# Approach (Cont...)

Hence, the training set mimics the role of the meta-training set, having an equal representation of classes (equal number of samples) to train the experts & to generate predictions on the held out (validation set, which again, has equal representation of classes), which when combined together (after neutralization), serves as a rich context for the meta-model (meta-validation set) to learn a mapping to the target values.

In simpler terms, the ‘task’ is to tune the weak classifiers to generate good predictions that serve as a rich source of context to the meta-model that learns to map them to the target values.
Hence, the better the predictions from each expert are, the better the meta-model will perform.

Moreover, incorporating a diverse pool of experts & increasing their number can further enhance the performance of the meta-model on the unseen meta-testing set.

# Other Approaches Tried

1. Splitting the dataset on the basis of ‘era’ & to feed them incrementally on a slew of selected experts with the predictions from the previous models acting as a context to the next.
2. Grouping sets of ‘era’ & creating separate meta-training and meta-validation sets to train the models in the model pipeline.

These approaches, although more standard in the meta-learning domain, didn’t yield great results on this dataset.

Surprisingly, the approach I finally narrowed down to seemed to generalize well from only 20% of the training parquet data (meta-validation set) on the unseen data from the validation parquet (meta-testing set).

# Models Used

Experts:

1. XgBoost Classifier
2. Random Forest Classifier
3. Adaboost Classifier with Decision Tree Classifier as Base Estimator
4. Logistic Regression
5. CatBoost Classifier
6. Histogram-based Gradient Boost Classifier

Meta-Model:

LightGBM Regressor

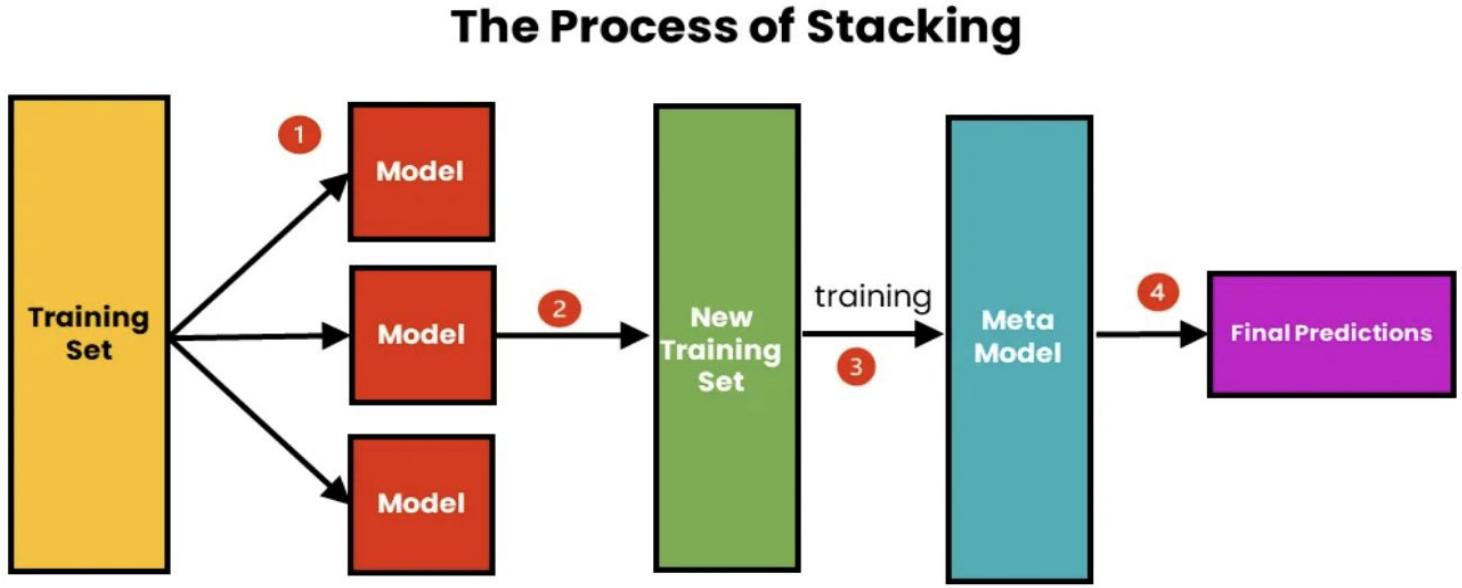


Image Source: [Link]

# Motivation behind using them

- The motivation behind choosing these models was their performance on similar kinds of data (large, multi-class, skewed dataset).

- Some of the classifier models were proven performers on the NumerAi dataset, while others showed good performance when applied to the NumerAi problem.

- The motivation behind adding tree-based classifiers was their good training (sometimes, they overfit the training set), which can serve as a good base for generating predictions (input) for the meta-model.

- Although I tried out different models as the base experts, I picked those models that performed well on this type of data while ensuring diversity in the model structure.

- The LightGBM Regressor model was chosen as the Meta-Model, considering its high usage & proven performance on the NumerAi dataset.

# Inspiration/Novelty of Work

The approach for this two-level stacking ensemble was from an article published in the medium that surprisingly worked well for the NumerAi dataset.

Furthermore, the addition of partial feature neutralization (neutralizing the predictions of the base experts but not training them again) boosted the performance of the meta-model (on the meta-testing set, the correlation went up from 0.009 to 0.0181 in one of the runs (saved on GitHub)receiving a more nuanced, high correlation-free input data (predictions) that, when combined together, formed a rich context source for the meta-model to learn a mapping that generalizes well on the unseen data (the meta-testing set).

# Other Approaches Tried

1. Used complex Deep Learning models such as ResNets, Inception Blocks, VGGs, Attention Models (LSTMs, GRUs, etc.), & Custom Deep Learning Models with different convolutional & pooling layers, in isolation as well as in an incremental/stacked fashion. Tried ensemble on deep learning architectures but did not yield great results (computationally expensive, too).
2. Tried Splitting the data into eras & sampled them (tried both oversampling and undersampling within eras), and applied a slew of experts (including Tabpfn along with some of the models specified above), in both the context-independent & context-dependent (forwarding of context to the next model, i.e., traditional stacking), and combined their predictions to create a meta-validation set for the meta-model (LightGBM).

Most of my experiments revolved around my current approach of the 2-level stacked ensemble but tried different architectures/models (shared in the papers within the Meta-Learning course) with different modeling of the NumerAi problem (splitting into single eras or combining them in chunks) and context/context-free training of base experts (i.e. traditional stacking within the experts) and later stacking the neutralized predictions into the meta-model.

Although these approaches were promising, they did not yield higher gains than some simpler modeling approaches; hence, I simplified these complex modeling approaches to the current, relatively simpler one.

# 2-level Stacking Ensemble (Pictorially)

Images taken from here (example depiction)

| Dataset | Description |
|---|---|
| Training Data | Used to train the base models |
| Hold-Out Validation Data | Used to train the meta-model |

Split the training dataset into two parts

| Model | Algorithm |
|---|---|
| Base Model 1 | Decision Tree |
| Base Model 2 | Neural Network |
| Base Model 3 | Support Vector Machine |

Train several base models on the Training Data

| Model | Base Model 1 | Base Model 2 | Base Model 3 |
|---|---|---|---|
| Prediction | Prediction 1 | Prediction 2 | Prediction 3 |

Make predictions using the base models on the Hold-Out Validation Data

| Model | Algorithm |
|---|---|
| Meta-Model | Logistic Regression |

Train the meta-model on the Hold-Out Validation Data.

# Training Workflow

The training data (the training parquet) was split into training and validation sets with an 80:20 ratio.

The training & validation sets were separately sampled (undersampled) to balance them.

The training set was used to train each of the experts & the validation set was used to generate predictions from each of the trained experts.

The predictions from the trained experts, made on the validation set, were neutralized that acted as input to the meta-model.
Since the experts were not re-trained (by using the neutralized predictions as context), the feature neutralization only incorporated the removal of the influence of highly correlated features to the target as these predictions are only being used as a context for the meta-model, thus removing the need for the addition of the removed factor back into the predictions of the meta-model.

Moreover, it is tricky to find out by what factor the predictions must be changed (added into the predictions) as different experts had different features correlated to their predictions & the meta-model, being trained on the predictions from these models, further complicates the feature neutralization process[1].

# Training Workflow

The meta-model used these predictions (on 20% of initial training parquet data), learned a mapping to the target values & performed well on the unseen data (the validation parquet), indicating good generalization.

Thus, the training set (80% of the training parquet data) acted as the meta-training set, the predictions on the validation set formed a part of the meta-validation set, and the data from the validation parquet acted as the meta-testing set.

**Limitation:**

[1] Although it might be possible to identify the correct/logical factor to add to the predictions of the meta-model, by which complete feature neutralization might be done, I found it particularly challenging to find a logically & intuitively correct factor that can be added to perform full feature neutralization. Hence, I found the removal of the influence of highly correlated features with respect to the target in the base experts to be an intuitively correct approach to model the problem.

The Meta-model predicted majorly 3 classes, leaving out the extreme class values (class 0 & 4), which may improve on adding more diverse classifiers.

# Evaluation Metrics

**Overall Accuracy:**

```
acc = accuracy_score(rounded_predictions, test_df_y_resampled)
print("Accuracy on Validation Set: ", acc)
```

```
Accuracy on Validation Set:  0.4350211396660943
```

**Pearson's Correlation:**

Pearson's Correlation

```
pearson_corr, _ = stats.pearsonr(rounded_predictions, test_df_y_resampled)
print("Pearson Correlation:", pearson_corr)
```

```
Pearson Correlation: 0.017730612069269 46
```

# Evaluation Metrics

**NumerAi Correlation:**

```
actual_corr = numerai_corr(rounded_predictions, test_df_y_resampled)
actual_corr
```

```
0.018142158486232806
```

**Class-wise Accuracy & F1 Scores**

```
Accuracy for class 0: 0.0000
Accuracy for class 1: 0.1673
Accuracy for class 2: 0.6938
Accuracy for class 3: 0.2702
Accuracy for class 4: 0.0000
```

```
F1 Score for class 0: 0.0000
F1 Score for class 1: 0.1929
F1 Score for class 2: 0.6251
F1 Score for class 3: 0.2448
F1 Score for class 4: 0.0000
```

**Source:** Link

# Learnings from the Project

- Simple Benchmark models (specially the Tree-based ones) perform relatively well in comparison to deep neural networks.

- Sampling of data helps in the overall performance of the model, since the data being skewed in nature.

- Undersampling works better than oversampling, as oversampling adds synthetic samples to a not so clean dataset, which creates issues in model training & yield poor performance.

- Training the ensemble on individual eras & stacking their performance does not yield great results.

- Feature Neutralization may or may not enhance the performance of the model, depending on the correlation threshold & proportion used for scaling.

- In a general trend, complexifying the model architecture leads to a drop in performance, specially when using strong classifier networks together. For example, using Tabpfn as one of the experts along with the other experts did not give me desired results as the meta model classified every sample as class 2 (0.5), even after feature neutralization.

- Strong classifier models work well in isolation, than when combined together. The simple LightGBM model gives a correlation of 0.03 (approx) but using deep neural networks along with attention based models gives a correlation as low as 0.002

# Learning in context of MTL

- Learned about modeling a problem as a meta-learning problem by creating meta-training, meta-validation & meta-testing sets.
- Learned about different types of models, such as Tabpfn, MotherNet, CNPs, etc.
- Learned about the role of context in enhancing the prediction of models.
- Learned about different types of modeling techniques such as Incremental Learning, Stacking & complex ensembles.
- Learned about neutralizing the predictions of a model using feature neutralization that removes the influence of highly correlated features with respect to the target.
- Learned about the role of chunking/grouping together of samples on the basis of some underlying feature/target values, which may pay dividends in meta-training.
- One striking observation was the meta-model from my ensemble being able to generalise well with only 20% of the training data, but dropped in performance when trained on the entire training data, i.e., without splitting it into training & validation sets.

# THANK YOU