Meta-Learning Report

The NumerAi dataset is an enormous dataset, which keeps on changing & increasing every week, where the weeks are denoted as ‘era’ & the task is to predict the ‘target’ values, taking into account the previous data, which serves as the context for future predictions. My primary approaches were to split the dataset on the basis of ‘era’ & balance the chunks (era-wise samples, since the dataset is skewed with class 2 (0.5) being predominant) using sampling techniques (Undersampling/Oversampling).

1. My approach towards the NumerAi prediction problem is a slight modification of the standard machine learning & meta-learning paradigm. I divided the training data into training & validation sets and used them to formulate an approach that mimics the traditional meta-learning techniques to some extent. I used a handful of weak classifiers/experts (6 in total) and trained them on the training set (which mimics the behavior of the meta-training set), and generated predictions using the validation set, which eventually serves as an input to the meta-validation set. Each expert is trained individually on the training set & their performance is measured on the validation set, much like in a standard machine learning paradigm. The predictions from these experts are then stacked together to serve as the meta-validation set to the meta-model, which uses these predictions as context to map them to the target values.

Ans1: 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 each having an equal number of samples per class (performed sampling to balance the skewed dataset, which mimics the k-way n-shot task). 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).

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).

Ans 2. The Models used in my approach were standard machine learning models having different structures, which were tuned (hyperparameter tuning) to individually perform well on the training & validation sets. I tried to accommodate a diverse set of experts for my 2-level stacked ensemble, which were namely Xgboost Classifier, Random Forest Classifier, AdaBoost Classifier, Logistic Regression, Catboost Classifier, & Histogram-based Gradient Boosting Classifier. 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. The approach for this two-level stacking ensemble was from an article published in the medium[1] 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 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).

References:

[1]
https://medium.com/@brijesh_soni/stacking-to-improve-model-performance-a-comprehensive-guide-on-ensemble-learning-in-python-9ed53c93ce28

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.

Ans 3.

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].

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.

#NOTE (Limitations):
[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.

4.

```
pip install scipy
pip install numpy
pip install pandas
pip install xgboost
pip install seaborn
pip install pyarrow
pip install numerapi
pip install imblearn
pip install catboost
pip install lightgbm
pip install matplotlib
pip install cloudpickle
pip install mplcyberpunk
pip install scikit-learn
pip install torch
pip install torchsummary
pip install tqdm
pip install gdown
```

5, 7 & 9.

Provide Google Drive Link to the Training & Prediction Zip

8.

Evaluation Metrics (Corr, Accuracy, Class-wise Accuracy & F1 Scores)

10.

Drive Link for Screenshot of CorrV20 on the NumerAi platform

11.

Drive Link for Presentation (Slides, 7 min).