AI Machine Learning Report

Kyryl Lebedin
rtgr66

1 Data Exploration

Statistical Outline - KNOW

This report is based on BraVl dataset. Further the scope is limited to ThingsEEG-Text subset of it, other WIKI subsets are not chosen to concentrate solely and brain-image-text relationship from ThingsEEG-Text.

To simplify examination and training (sections 1-3) ThingsEEG-Text is restricted further: unseen datasets are not used, because the text and image data in them are duplicated unlike in seen dataset ; only data from subject 10 is used; brain data channels are flattened, time is restricted to 70ms-400ms interval; PCA is applied to images reducing their number of features to 100; all the features for every dataset are scaled accordingly to enhance numerical stability during model training. Result is working dataset with 3 modalities stored in 2D arrays.

Modality	Range	Mean	Std
Brain EEG	$(-6.94,\ 6.06)$	-0.0610	0.946
Image (PCA)	$(-81.2,\ 405)$	-0.0214	4.90
Text (CLIP)	$(-5.45,\ 12.1)$	0.0136	0.701

Figure 1: Summary statistics for the working datasets.

Figure 1. table shows the value distribution in flattened working datasets. The classes are balanced with exactly 10 samples per class in every modality for every of 1654 unique classes, resulting in total 16540 samples. All modalities are tested for duplicates and missing values: there were no missing values detected, brain data and images do not contain any duplicates, while text data contains 1777 duplicate rows.

Visualization - SEE

To understand the dataset further and determine data processing strategy, main characteristics of dataset are visualized.

From section 1, text data has duplicates in it. In Figure 2. 10 samples of a same class are visualized, duplicates are colored the same (blue - means no duplicate). This class has 7 samples out of 10 duplicated. Also there are some classes that have up to 10 exact same duplicates.

Figure 3 elaborates the information from Figure 1: value distribution, range, means. The visualization is made with KDF plots for each modality. Plots represent probability density of values offering a clean picture of values distribution. The curves on the graphs represent $\pm 3\sigma$ from mean which is enough to identify potential outliers with the assumption of roughly normal distribution.

Figure 4 shows correlation matrices for every modality. Despite showing highly correlated features, they can show patterns in the data, that help to determine how features interact with each other.

Figure 5 shows feature importance of all modalities stacked together: brain, image and text respectively. The number of samples in this examples is reduced to 2000 to obtain better visualization and duplicates from text are removed. The importance are determined with help of Scikit-learn Random Forest Classifier based on gini impurity. A lot of features have low importance which means that they play insignificant role in decision making of model. Further in training model can benefit from filtering out uninformative features. Text embedding seem to capture the most semantic and discriminative information for this particular RFC model. There are still some important features among brain and image.

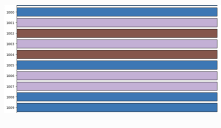

Figure 2: Value distribution

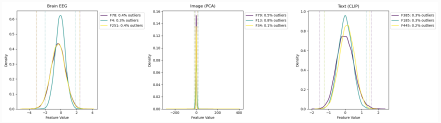

Figure 3: Value distribution

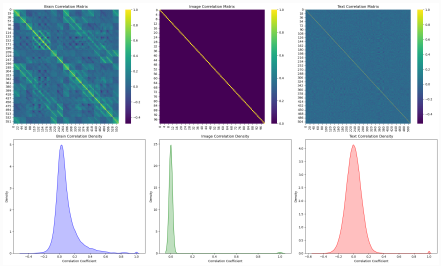

Figure 4: Correlation matrices

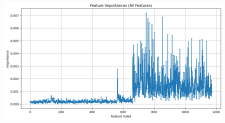

Figure 5: Importance graph

Exploration Summary - FIND

The classes in datasets are balanced and data is distributed with fairly low number of outliers as shown Figure 3. Brain and text data is fairly normal and may not require specialized outlier-handling methods, only 0.4% outliers. In image data some principal components appear tightly centered around means and some are spread wider suggesting higher variance. Despite very low percentage of text data being outlier, KDF is very long-tailed, a model sensitive to outliers (e.g. distance-base) can be badly influenced by them. This will be considered when choosing model.

The text data contains 1777 duplicate rows. Though the experiment setup for data collection allowed text duplicates, this can reduce the generalization ability of the model for the unseen data. Classes that include duplicates are dropped for further processing.

The least correlated modality is images, showing overall correlation of approximately 0. The most correlated modality is clearly brain data with mean at 0.08 (compared to text and image with 0.002 and 0.01 respectively) and a standard

deviation of 0.14 (compared to 0.01 for images and 0.1 for text). Dataset can benefit from removing features with extremely high correlations. Features with (e.g., $|r| \geq 0.9$) are dropped, resulting in removal of 33 features.

A lot of features have low importance which means that they play insignificant role in decision making of model. Further in training model can benefit from filtering out uninformative features. Text embedding seem to capture the most semantic and discriminative information for this particular RFC model. There are still some important features among brain and image.

2 Custom Model Implementation

Initial Custom model - IMPLEMENT

Sklearn Random Forest Classifier is chosen as a baseline model. There are several reasons to it.

RFC handles data with high dimensionality well. This is especially relevant in combined modalities dataset that includes more than 1000 features when uncompressed.

In Figure 3 Text KDF plot shows long tails. RFC relatively robust to outliers because splits are determined by thresholds rather than relying on a single global parameter (as in linear models) or measuring distances directly (as in distance-based models like k-Nearest Neighbors).

In figure 4 it was shown that text and brain data have features that correlate with each other, and though it can partially be mitigated with dropping features that correlate, RFC handles correlated features well and do not suffer from them as much as other models, because each tree in the ensemble samples features randomly at each split, reducing the chance that a single subset of correlated features will dominate.

RFC provides clear way to measure importance of features that gives great potential to improve the model. This can be useful considering the amount of unimportant features in Figure 5.

To understand the work of RFC and Decision Trees I referred to videos by **Normalized Nerd** as a foundational resource. I started with top down approach, first writing Custom_rfc class and progressively delving into the DT class and Node class. I made sure to add customization hyper-parameters, such as n_estimators, max_depth, min_sample_split, max_samples, max_featrues. I used gini impurity in my initial implementation, as cretinoid for evaluating splits. The featrues for each tree are selected randomly and bootstrap dataset are chosen randomly with replace. Build_tree is recursive function that builds each tree in the forest.

The dataset was modified by: dropping duplicates from text, removing features with correlation $|r| \geq 0.9$. The values are not scaled or normalized as RFC model is immune to it. LDA&PCA do not show any increase in accuracy. This way also dataset stores it's initial information unrestricted.

Optimal hyper-parameters values are shown in table in Table 1 below. Further hyper parameter tuning has proven that increasing values of n_estimators and max_depth further does not improve model any more and accuracy plateaus, which proves optimality.

Train/Test	n_estimators	max_depth	Criterion	min_sample_split	max_features	LDA & PCA
70/30	250	20	gini (custom) entropy (baseline)	4	sqrt	None

Table 1: Optimal hyper parameters

Custom and Baseline Comparison - COMPARE

Method	Number of Samples	Running Time (sec)	Accuracy
Baseline	20	2.8	0.912
Custom	20	212.3	0.8947
Baseline	50	5.5	0.8333
Custom	50	1353	0.8403

Table 2: Initial test

Table 2. shows performance of the models both on 20 and 50 categories. The accuracy of both models degrades slightly with increasing of categories number. This is expected as with introducing new classes that are all balanced the task for model becomes more complex. The fact that accuracy did not degrade significantly supports hypothesis that results for smaller number of categories, such as 20, describe the performance of model as well as model trained on higher number of categories.

This allows to save time while testing the model, only using 20 categories for training, with 70/30 split. The Baseline model is significantly faster than Custom, for 50 sample faster by 24,600%. This can be explained with Baseline model utilizing multiprocessing and optimization with Cython.

In Table 3, to test and compare accuracies and running times 5-Fold Cross Validation is performed. Here multiprocessing was already implemented into Custom model to save time. The experiment setup has statistical significance, proves that models perform well on the whole dataset, accounts for potential outliers in accuracy.

The final accuracies show that Baseline model is 2% more accurate with lower standard deviation, that suggests modest but consistent dominance.

Model	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**	**Accuracy (Mean ± Std)**	**Running Time (Mean ± Std)**
Baseline	0.904	0.865	0.904	0.865	0.827	**0.873 ± 0.029**	**0.484 ± 0.021**
Initial Custom (multiprocessing)	0.885	0.827	0.885	0.808	0.885	**0.858 ± 0.034**	**115.551 ± 3.467**

Table 3: Baseline vs Initial Custom (5-Fold)

Custom Model Improvement - IMPROVE

The first and obvious step is to process multiple trees at the same time by utilizing multiprocessing. This makes Custom model running time independent of number of n_estimator hyper-parameter. It is important as First Custom model creates one tree at a time, which makes it follow by: $F(t) = O(t * m * n * \log(n))$
where: t - n_estimators (number of trees), m - max_fetures, n - number of samples. This and vanilla python implementation explains $\Delta t = 1140.7$ (537.59% increase) for adding another 30 categories in Table 2. The refined model runs significantly faster, that can be seen in Table 3.

When tuning hyper-parameters Baseline model showed better performance when calculating impurity with entropy rather than gini. Initial baseline model only has gini method as it is easier to implement. In Improved Custom entropy is implemented. Log2 is computed faster than exponent.

The major improvement in Custom model is filtering features based on their importance. The importance calculation is added to class implementation, summing up the impurity change for every feature in every tree across the forest. The hyper-parameter importance threshold is then defined by randomized CV.

3 Result Analysis and Visualization

Statistical outline - PERFORMANCE

Table 4 performs the 5-Fold CV for all 3 models with optimal hyper-parameters. The importance threshold for improved model is $I = 0.0007$. For each fold table shows accuracies and macro-averages of precision, recall and f1. Weighted and micro- averages are not used as data is perfectly balanced with 10 samples per category.

Despite values do not differ significantly in absolute values, 5-Fold CV proves the improvement of the model. The data could have easily been reverse engineered by not applying data processing to Baseline and Initial custom model (duplicate drop, correlation removal) and applying to "Improved" to show higher increase in accuracy. Though the report focuses more on showing quality model improvement based on already optimally processed data, rather than creating good conditions for one model and bad for another to obtain higher absolute difference.

While initial Custom performs worse than Baseline, Improved Custom is better than Baseline in every statistic, matching accuracy with lower std, higher precision, recall, f1. Consistently high values for every statistic across all models suggests that model predicts equally well and consistent for every category.

Model	Fold 1				Fold 2				Fold 3				Fold 4				Fold 5				Summary (Mean ± Std)				
	Acc.	Prec.	Recall	F1	Acc.	Prec.	Recall	F1	Acc.	Prec.	Recall	F1	Acc.	Prec.	Recall	F1	Acc.	Prec.	Recall	F1	Acc.	Prec.	Recall	F1	Train Time
Baseline	0.904	0.917	0.885	0.880	0.865	0.812	0.796	0.788	0.904	0.913	0.917	0.902	0.865	0.887	0.919	0.878	0.827	0.728	0.794	0.742	**0.873 ± 0.029**	**0.851 ± 0.072**	**0.862 ± 0.056**	**0.838 ± 0.062**	**0.491 ± 0.015**
Custom	0.827	0.819	0.823	0.788	0.846	0.833	0.833	0.813	0.827	0.858	0.797	0.793	0.827	0.849	0.867	0.828	0.846	0.827	0.851	0.814	**0.835 ± 0.009**	**0.837 ± 0.014**	**0.834 ± 0.024**	**0.807 ± 0.015**	**87.436 ± 4.251**
Custom Improved	0.885	0.906	0.865	0.866	0.846	0.826	0.843	0.822	0.865	0.889	0.865	0.864	0.904	0.913	0.947	0.908	0.865	0.856	0.845	0.829	**0.873 ± 0.020**	**0.878 ± 0.033**	**0.873 ± 0.038**	**0.858 ± 0.031**	**56.526 ± 1.477**

Table 4: Baseline vs Custom vs Custom Improved (5-Fold), I=0.0007

Visual Comparison- VISUALIZATION

In Figure 8 and 10 importance graphs are provided for Baseline and Custom models. Though both models leverage Text data more than other, Custom model shows vastly higher relative importance of image and brain data in prediction. On the other side Baseline scikit model predicts labels almost solely based on text data, ignoring brain and image.

On Figures 6-8 the confusion matrices are shown for fold 2 in CV for each model. The cluster can be spotted, where all the models get confused $x \in [18 - 28]$ $y \in [37 - 32]$. Though for other folds, this cluster does not appear, this can be tested into Notebook code. Comparing CM for Custom models, it can be seen that improved model, based on filtering unimportant information struggle less with values from upper triangle.

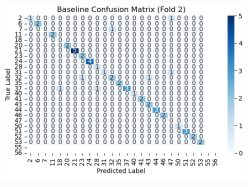

Figure 6: CM Baseline

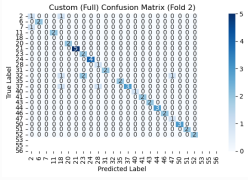

Figure 7: CM Custom

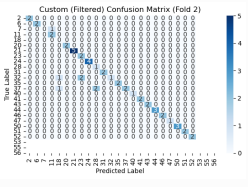

Figure 8: CM Custom Improved

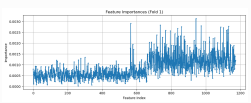

Figure 9: Importance Custom

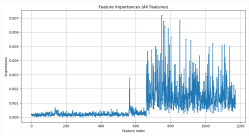

Figure 10: Importance Baseline

Improvements analysis- ABLATION

Here is the summary of all the ideas used to overcome the problems from section 1.FIND.

First of all due to heavy tails in text data and outliers the RFC model is chosen because tree-based methods are generally more robust to outliers. It uses split threshold for each decision tree rather than metrics like Euclidean distance.

Although duplicated were allowed during data collection, the classes with text duplicates were dropped during data processing for model. This reduces the risk of over-fitting to repeated samples, increasing model's confidence to generalize for unseen data.

During data processing the features with with correlation $|r| \geq 0.9$ are dropped, This makes model focus on genuinely informative features, which proved to be more beneficial through 5-fold CV. It makes training time is made faster as well. On the other side dimensionality reduction technics only make model to perform worse.

It was spotted that data has a lot of features that do not contribute to classification, looking at importance graphs Figures 5, 9, 10. Implementing importance filtering mechanism was the main idea for improvement in Custom implementation. The performance of improved model, above in this section, shows good evidence to support the hypothesis made in first section: "the model will benefit a lot from removing unimportant features". Also it is shown in VISUALIZATION through importance graphs Figures 9-10 that custom implementation leverages every modality and not just focuses on text data. In summary, the final model performs better than baseline on same dataset and utilizes inter-modal structure of dataset better.

4 Paradigm Design and Data Splitting

Federated paradigm - PARADIGM

The Federated learning paradigm is particularly relevant in our era of rapid technological progress. Soon with the help of companies like Neuralink and Synchron the brain activity scans such as EEG can become as sensitive and personal as photos from private gallery. Maybe one day in the future one would tick the required box, "allow analysis of user EEG" just like currently do with "personal data". Federated learning paradigm can ensure secure handling of such data.

Hypothesizing about the future and possible applications can show how Federated paradigm can support real world use cases. The industries can highly benefit from being able to customize their products based on customers' brain data. Processing such data in its nature raises many ethical considerations, such as consent, data ownership, risk of misuse. The Federated learning could mitigate the risks by keeping and processing the data in decentralized way, keeping data on users devices. Only summary parameters are shared and aggregated, meaning raw EEG signals never leave user's domain. In the future Federated approach for such data can offer critical balance between innovation and data protection.

Model change - ADJUSTMENT

The adjust this report model, data handling pipelines can be separated into two distinct RFC models: one focusing on brain EEG, another on Text and Image combined as they pose less privacy risk. This can model the process how in real world user's brain data, collected from chip could provide additional information into ML classification tasks. Then summary information from both pipelines can be combined to determine the class of test samples.

Focusing on lower amount of modalities can allow tailoring models to handle specific modality better, producing better result. Although, the information gained from 2 sub-models need to be weighted accordingly, because as seen in section 3.VISUALIZATION some modalities can determine the class better than others, based on importance graphs. This would fairly account for brain information gain, to maximize the overall efficiency.

Summary - REFLECTION

The new model is built. As suggested in subsection above it is ensemble of Random Forests that evaluates brain and text+image separately. It used Custom Improved model as sub-model. It is further refereed as RBC or Random Biome Classifier. Two models evaluate classes and then based on these two prediction sets final set is built by choosing randomly between two based on probability weight of each sub-model. Two sub-models are trained in parallel, as it would happen in real world.

Train/test split is picked 70/30 as before to maximize the accuracy. The dataset is preprocessed in the same way (remove correlation and duplicates). New optimal hyper-parameters are deduced for sub-models. Brain sub-model has weight 0.1 and text+image 0.9. Table 5 shows the average accuracy and std for 5 runs with different random state. The average accuracies for brain sub-model is 0.2, text+image 0.8.

Summarizing, the performance of RBC is lower than any implemented before, but the difference is only 15%. The lower accuracy can be explained by inefficiency of data or model for brain prediction. Although, RBC allows to introduce user brain data protection and proportionally account for brain features in combination with brain + text. This can be efficient approach introducing the constraints of Federated Paradigm, and potentially can be used in real world applications.

Model	Run 1	Run 2	Run 3	Run 4	Run 5	Mean ± Std
Custom Federated	0.7340	0.7689	0.7378	0.7519	0.7688	**0.7523 ± 0.0148**

Table 5: Custom Federated accuracy across 5 random runs

References

[1] Normalized Nerd. *Understanding Random Forests and Decision Trees*. YouTube video, 2025. Available at: `https://www.youtube.com/watch?v=sgQAhG5Q7iY&ab_channel=NormalizedNerd`.