Geo-location of German Tweets

Adrian Iordache

January 02, 2021

Abstract

In this project is presented a solution for predicting the latitude and longitude of a dataset based on nearly 30 thousand of German Tweets. The approach presented in the next few pages represents a Multi-Layer Stacking Architecture, in this particular case, a three level architecture, with input based from Term Frequency - Inverse Document Frequency features obtained on the raw tweets and the preprocessed text, for both word and character level, to stacked embeddings from transformers networks.

1 Introduction to stacking architectures

To better understand the concept of stacking we need to go back to basic averaging ensembles. A model averaging ensemble combines the predictions from multiple trained models. This approach helps for better generalization reducing the possible bias of a learning model. By increasing the number of low correlated models we can be more certain in our final predictions.

The most important limitation of standard averaging ensambles comes from the fact that each model contributes with the same amount to the ensamble prediction, regardless of the model performance.

A better variation of this approach, weighted average ensambles, weights the contribution of each model based on the performance on the validation set. This allows well-performing models to contribute more the the final prediction. The set of weights in the ensamble are often brute forced or optimized.

A further step in this generalization technique is based on replacing the weighted sum by learnable weights used to combine the predictions of the sub-model, done with any learning algorithm. This approach is called stacked generalization, or stacking for short.

In stacking, an algorithm takes the outputs of sub-models as input and attempts to learn better combinations of the input predictions to make a better output predictions.

As you can see in the image below (Figure 1), a two layer stacking architecture is based on the concept of training a set of level one models, that predicts the initial dataset and those predictions are used as input to a distinct set of level two model / models and those will generate the final prediction.

This type of architecture offers a better change for generalization reducing the biases of each individual model and it can be scaled for multiple layers.

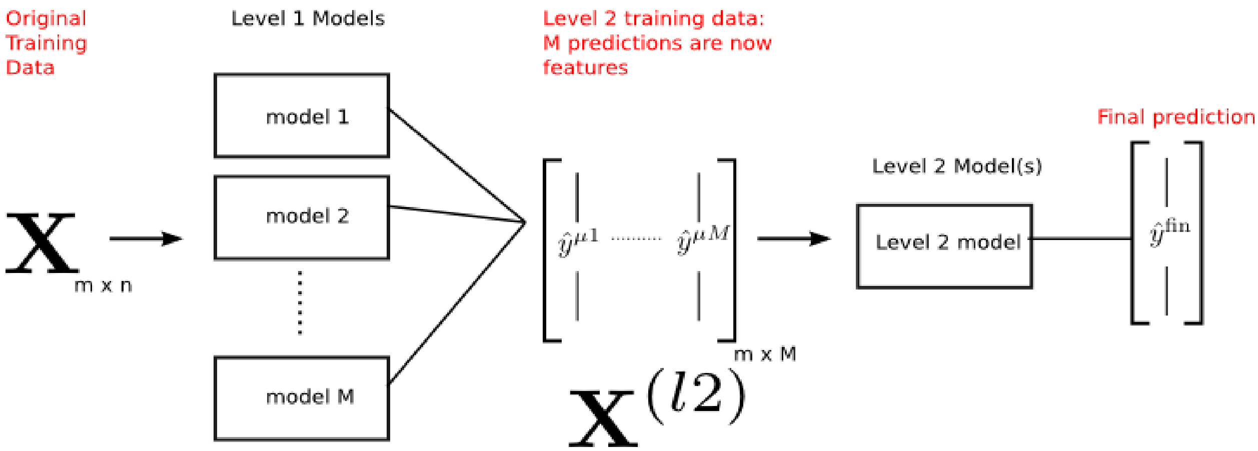

Figure 1: Standard Stacking Architecture

Besides scaling, another important aspect of stacking, that helps preventing overfitting or data leakage, is training a each model based on a cross-validation strategy and taking only the validation fold for each model. After K models will have the "out of fold" predictions ready to use as input for the next layer.

Now that we have some basic understanding of stacking architectures we can go further in presenting the solution.

2 Solution Overview

As we were provided with a training and a validation set, for a better use of all the data I decided to merge those two sets and use a cross-validation strategy. So the following results are based on the OOF MAE (Out of Fold Mean Absolute Error).

Another important aspect that needs to be mentioned, in the initial phase of this project all the frameworks used, system settings and models were seeded so we can be confident that all the experiments and results are reproducible.

As we said in the beginning of this project the solution is based on a three level stacking architecture.

2.1 Input Preprocessing

The input for the stacking architecture is based on three options:

- Term Frequency Inverse Document Frequency Features on the Raw Tweets for both Word and Character Level
- Term Frequency Inverse Document Frequency Features on the Clean Tweets for both Word and Character Level
- Stacked Word Embeddings on the Clean Tweets merged with Transformer Embeddings from BERT (Bidirectional Encoder Representations from Transformers)

For text cleaning was done: removing URLs, Emojis, HTML Tags and Punctuation Marks, tokenization and lower case for all words, removed german stopwords and lemmatization.

Embeddings are obtained from Flair NLP (A very simple framework for state-of-the-art NLP) using Document Pooling (that simply does an average over all word embeddings in the sentence) from three types of embeddings:

1. German Fast Text Embeddings
2. Flair Forward Embeddings
3. Flair Backward Embeddings

Flair embeddings represent contextual string embeddings that capture latent syntactic-semantic information that goes beyond standard word embeddings. Key differences are: (1) they are trained without any explicit notion of words and thus fundamentally model words as sequences of characters. And (2) they are contextualized by their surrounding text, meaning that the same word will have different embeddings depending on its contextual use.

The average of those embeddings will be merged with the embeddings from a german BERT model, resulting the third type of input in our stacking architecture.

2.2 Level One Models - High Variance Layer

In any stacking architecture is very important that the models in the first layer (or layers) to be as "different" as possible, but still maintaining high performances. From that variance we can obtain an extended point of view for the problem we are trying to solve.

The method I used for measuring the level of "difference" between two models is based on the level of correlation.

To ensure that the level one maintains variance I preferred to analyze each possible candidate for entering level one based only on the MAE on the validation set and if the results are better than an empiric threshold chosen, the model will be added to the level.

Now that we have a number of "good enough" models for the first level we can apply feature selection to choose which models will remain.

For the feature selection part I used a custom metric based on a weighted average between the Out of Fold MAE of each model and the level of correlation between the selected model and the rest of the models in level. This way we can retain only a chosen percentage of models that have high variance and low OOF MAE. This chosen percentage will be later a hyper-parameter for bayesian optimization.

The layer one will contain 57 models based hyper-parameters changes.

Models used in level one:

- LGBMRegressor with different n_estimators
- RandomForestRegressor with different n_estimators
- Ridge with different alphas (regularization term)
- SVR with different C values
- XGBRegressor with different n_estimators and learning rates

Each model will be trained for different ngram ranges for both word or character level if Term Frequency Inverse Document Frequency Features are used.

Models that did not enter the level one:

- LARS Lasso
- Extremely Randomized Trees
- Bayesian Regression
- Adaptive Boosting
- Neural Networks
- Kernel Ridge Regression

2.3 Level Two Models - Specialization Layer

At this point of the solution we want to reduce variance in favor of better scores, in this case, for reducing out of fold error.

This layer will contain only the best regressors for a small range of hyper-parameters, this way we can be sure that a small percentage of variance is retain and this will give a better chance for generalization.

Models in level two layer:

- 5 * SVR with different C values
- 4 * Histogram-based Gradient Boosting Regression Tree
- 5 * Categorical Boosting Regressor (CatBoostRegressors)
- 1 * RandomForrestRegressor
- 1 * Bagging Regressor with NuSVR for 10 estimators

2.4 Level Three Models - Final Voting Layer

Usually in stacking architectures the last layer is represented by a simple regressor or the best regressor in the archtecture. But in this case, from my experiments, the best model for the final layer was a voting system based on the best regressors in the second layer.

So for the meta-learner was choose a Voting Regressor with uniform weights

- SVR
- HistGradientBoostingRegressor
- CatBoostRegressors

The final layer hyper-parameters were searched using Bayesian Optimization.

2.5 Searching for hyper-parameters - Bayesian Optimization

For searching hyper-parameters I used Bayesian Optimization, the main reason for using this type of approach was that in contrast to other hyper-optimization searching procedures, like GridSearch, RandomSearch that do not use knowledge from the previous experiments, the bayesian search uses a Expected Improvement function that guides the search, obtaining better results in larger hyper-parameters spaces in much less time.

For using this type a pipeline we need to establish:

1. Initial Bounds or the hyper-parameters spaces
2. Number of initial rounds (random iterations from the space of hyper-parameters)
3. Number of bayesian rounds (number of trials conditioned by previous experiments)

For this solution we used 128 initial points and 512 bayesian iterations over the next hyper-parameter spaces

- (SVR) C = (0.1, 20)
- (CatBoostRegressor) n_estimators: (100, 2000)
- (HistGradientBoostingRegressor) learning rate: (0.01, 1)
- (CatBoostRegressor) learning rate: (0.01, 1)
- (HistGradientBoostingRegressor) max_leaf_nodes: (4, 64)
- (Feature Selection based on Custom Metric) keep_percentage: (0.5, 1)

Best Parameters from Bayesian Optimization:

- (SVR) C = 14
- (CatBoostRegressor) n_estimators: 1424
- (HistGradientBoostingRegressor) learning rate: 0.06
- (CatBoostRegressor) learning rate: 0.01
- (HistGradientBoostingRegressor) max_leaf_nodes: 23
- (Feature Selection based on Custom Metric) keep_percentage: 0.96

Out of Fold Mean Absolute Error: 0.4827

This Score was my best OOF Error and actually my best submission on the private leaderboard.

But for my best chosen submission I used an experimental trick. I logged all my iterations of bayesian optimization, OOF Error, parameters and each iteration predicted the test set and I used those to average my final submission.

This was my best chosen submission.

For a better understanding, a diagram of the solution will be presented below (Figure 2).

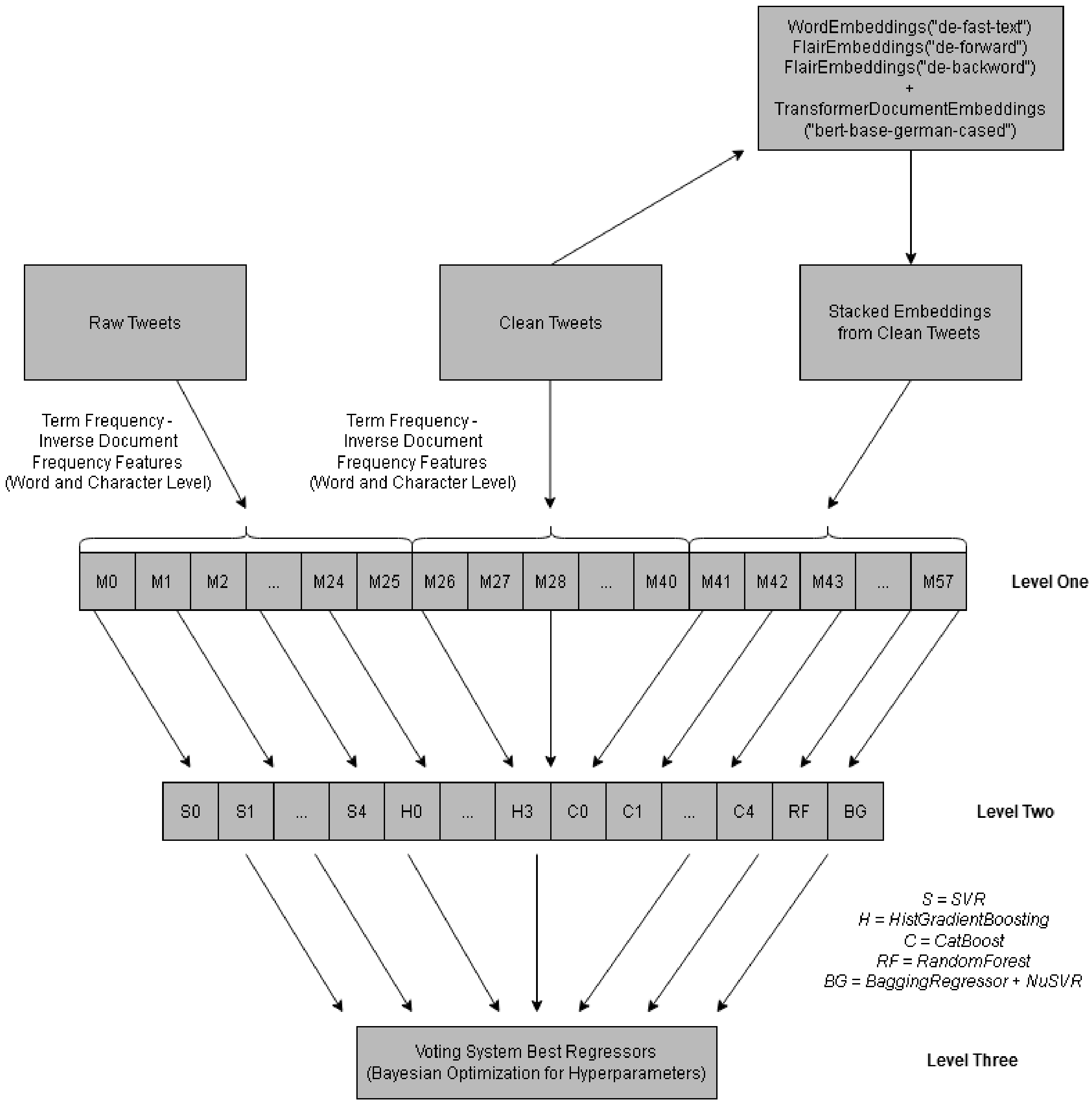

Figure 2: Solution Architecture

2.5.1 Second Chosen Submission

A variation of the solution presented earlier, but using for second level only:

- 5 * SVR with different C values
- 4 * Histogram-based Gradient Boosting Regression Tree

3 Cross-Validation Strategy

As we said in the beginning of this project, for a better use of the data I decided to use cross-validation strategies, so all of my labeled data will be used in developing the models.

Usually in stacking architectures it's recommended to use the same folds between layers, to prevent possible data leakage and from that OOF error would not be relevant.

Taking into consideration the fact that each models will see the data for the first time, I choose to change the cross-validation strategy between level one and level two, but for models in the same layer the chosen folds will stay the same.

3.1 Level One - Repeated K-Folds

In Level One I used Repeated K-Folds with 5 Folds for 5 Repetitions, the reason for that is based on the following deduction.

As I previously said, we need the models in the first layer to contain as much variance as possible, but the variance is necessary between the models, not between the folds of the same model.

So to reduce variance between folds, I trained 5 Folds for 5 Repetitions and at the end I averaged the axis of repetitions for the test predictions and out of fold predictions that will be used in the second layer.

3.2 Level Two and Three - Stratified K-Folds

Stratified K-Folds is usually very used in Classification problems with imbalanced data, the reason for that is based on the fact that we want to train and evaluate each model on data that has the same distribution as the initial dataset.

The problem here being a double regression, might not be so usable here, but after we look at the label distribution for each target (Figure 3 and 4), we can observe some skewed regions and those might represent a problem if we want a representative OOF Error.

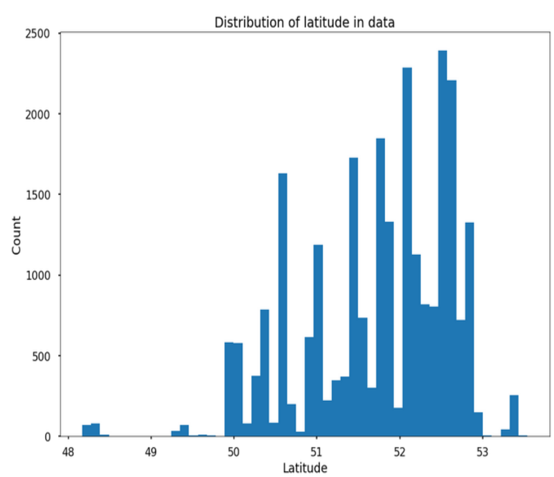

Figure 3: Distribution of latitude coordinates

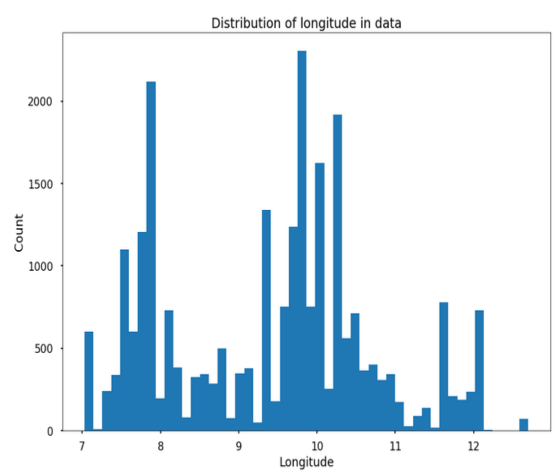

Figure 4: Distribution of longitude coordinates

After this observation, I decided to transform the continuous space of latitude and longitude into a discrete space based on bounds extracted from distributions.

Latitude Bounds: $[48, 49, 50, 51, 52, 53]$

Longitude Bounds: $[7, 8, 9, 10, 11, 12]$

Those intervals in discrete form will be used for stratification of the folds. This methodology helped improve OOF Error by 0.003.

4 Out of Fold Results

To present our OOF Error for each layer will use a standard average of all model errors for both, latitude and longitude.

This might not be representative because the whole purpose of stacking architectures is to find a better combining functions than standard averages or weighted averages, but will give us just a slightly idea about the differences between levels and the improvement resulted from stacking.

	MAE latitude	MSE latitude	MAE longitude	MSE longitude	MAE	MSE
Level One	0.5285	0.4758	0.7084	0.8686	0.6185	0.6722
Level Two	0.4496	0.3721	0.5373	0.6070	0.4934	0.4896
Level Three	0.4410	0.3641	0.5245	0.5977	0.4827	0.4809

5 Leaderboard Results - 2nd Place Solution

- Best Chosen Submission $\longrightarrow$ Public LB: 0.47058, Private LB: 0.47490
- Second Chosen Submission $\longrightarrow$ Public LB: 0.47061, Private LB: 0.47578
- Best Public Submission $\longrightarrow$ Public LB: 0.47113, Private LB: 0.47462
- Best Private Submission $\longrightarrow$ Public LB: 0.46900, Private LB: 0.47782

6 Implementation and Development

This section is meant to present the structure of the project and each individual script.

- utils.py $\longrightarrow$ Main utility script used for importing modules, setting constants, seeding, reading, defining other utility functions for the rest of the project (will be imported by most of the other scripts)
- translator.py $\longrightarrow$ Contains wrappers over various translations frameworks to be used in later experiments

- blending.py ⟶ Based on the level of correlation blend official submissions for better results
- adversarial_validation.py ⟶ Script for Adversarial Validation to make sure the test splitting is done right (used especially for embeddings)
- baseline.py ⟶ Initial script for baseline submissions, using StackingRegressor from sklearn in two level stacking architecture
- text_cleaning.py ⟶ Used for cleaning the input tweets
- embeddings.py ⟶ Wrappers over various NLP frameworks used to generate embeddings
- embeddings_analysis.py ⟶ Simple training script for validating different types of embeddings
- stacking_embeddings.py ⟶ Utility script used for merging embeddings
- feature_selection.py ⟶ Methods for feature selection between stacking levels or test features
- bayesian_optimization.py ⟶ Script for Bayesian Optimization for hyper-parameter tuning
- model_analysis.py ⟶ Training script used for model evaluation, will be used to establish which models will enter level one
- stacking_level_one.py ⟶ Adding model after analysis to level one features
- stacking_level_two.py ⟶ Adding model after analysis to level two features
- level_three.py ⟶ Final script, training the final layer of the architecture

Another three notebooks were used: one for EDA, one for accelerating embeddings on GPU and the last one for training SVR, Ridge and RandomForest on GPU using a dedicated library.

7 Conclusions

In this project we have developed a solution for predicting geographic coordinates of german tweets based on basic knowledge on Natural Language Processing with more advanced techniques of Machine Learning and Stacking Architectures for reducing biases of each individual model, maintaining a balance between the level of variance in each level and the mean absolute error.

Appendix A Level One Models

id	model	parameters	oof_error	observation	label	text
0	LGBM	n_estimators = 30	0.5435	None	latitude	final_text
0	LGBM	n_estimators = 30	0.725	None	longitude	final_text
1	LGBM	n_estimators = 30	0.5178	ngram_range = (1, 7), analyzer = 'char_wb'	latitude	final_text
1	LGBM	n_estimators = 30	0.6767	ngram_range = (1, 7), analyzer = 'char_wb'	longitude	final_text
2	RandomForestRegressor	n_estimators = 30	0.5226	None	latitude	final_text
2	RandomForestRegressor	n_estimators = 30	0.6651	None	longitude	final_text
3	RandomForestRegressor	n_estimators = 50	0.5223	None	latitude	final_text
3	RandomForestRegressor	n_estimators = 50	0.6646	None	longitude	final_text
4	RandomForestRegressor	n_estimators = 100	0.5221	None	latitude	final_text
4	RandomForestRegressor	n_estimators = 100	0.6636	None	longitude	final_text
5	LGBMRegressor	n_estimators = 50	0.5268	None	latitude	final_text
5	LGBMRegressor	n_estimators = 50	0.6897	None	longitude	final_text
6	LGBMRegressor	n_estimators = 100	0.5155	None	latitude	final_text
6	LGBMRegressor	n_estimators = 100	0.6701	None	longitude	final_text
7	LGBMRegressor	n_estimators = 100	0.4915	ngram_range = (1, 10), analyzer = 'char_wb'	latitude	final_text
7	LGBMRegressor	n_estimators = 100	0.6258	ngram_range = (1, 10), analyzer = 'char_wb'	longitude	final_text
8	LGBMRegressor	n_estimators = 100	0.4913	ngram_range = (1, 15), analyzer = 'char_wb'	latitude	final_text
8	LGBMRegressor	n_estimators = 100	0.6257	ngram_range = (1, 15), analyzer = 'char_wb'	longitude	final_text
9	LGBMRegressor	n_estimators = 300	0.4866	ngram_range = (1, 10), analyzer = 'char_wb'	latitude	final_text
9	LGBMRegressor	n_estimators = 300	0.6175	ngram_range = (1, 10), analyzer = 'char_wb'	longitude	final_text
10	LGBMRegressor	n_estimators = 500	0.4874	ngram_range = (1, 10), analyzer = 'char_wb'	latitude	final_text
10	LGBMRegressor	n_estimators = 500	0.6177	ngram_range = (1, 10), analyzer = 'char_wb'	longitude	final_text
11	LGBMRegressor	n_estimators = 100	0.5236	ngram_range = (1, 3), analyzer = 'word'	latitude	final_text
11	LGBMRegressor	n_estimators = 100	0.6842	ngram_range = (1, 3), analyzer = 'word'	longitude	final_text
12	LGBMRegressor	n_estimators = 300	0.5561	ngram_range = (1, 5), analyzer = 'word'	latitude	final_text
12	LGBMRegressor	n_estimators = 300	0.753	ngram_range = (1, 5), analyzer = 'word'	longitude	final_text
13	Ridge	alpha = 0.1	0.5237	ngram_range = (1, 5), analyzer = 'char_wb'	latitude	final_text
13	Ridge	alpha = 0.1	0.6959	ngram_range = (1, 5), analyzer = 'char_wb'	longitude	final_text
14	Ridge	alpha = 0.1	0.5249	ngram_range = (1, 7), analyzer = 'char_wb'	latitude	final_text
14	Ridge	alpha = 0.1	0.6975	ngram_range = (1, 7), analyzer = 'char_wb'	longitude	final_text
15	Ridge	alpha = 0.1	0.5262	ngram_range = (1, 10), analyzer = 'char_wb'	latitude	final_text
15	Ridge	alpha = 0.1	0.6990000000000001	ngram_range = (1, 10), analyzer = 'char_wb'	longitude	final_text
16	Ridge	alpha = 0.5	0.5026	ngram_range = (1, 5), analyzer = 'char_wb'	latitude	final_text
16	Ridge	alpha = 0.5	0.6656	ngram_range = (1, 5), analyzer = 'char_wb'	longitude	final_text

id	model	parameters	oof_error	observation	label	text
17	Ridge	alpha = 0.5	0.5054	ngram_range = (1, 7), analyzer = 'char_wb'	latitude	final_text
17	Ridge	alpha = 0.5	0.6693	ngram_range = (1, 7), analyzer = 'char_wb'	longitude	final_text
18	Ridge	alpha = 0.5	0.5081	ngram_range = (1, 10), analyzer = 'char_wb'	latitude	final_text
18	Ridge	alpha = 0.5	0.6731	ngram_range = (1, 10), analyzer = 'char_wb'	longitude	final_text
19	Ridge	alpha = 1	0.5034	ngram_range = (1, 5), analyzer = 'char_wb'	latitude	final_text
19	Ridge	alpha = 1	0.6671	ngram_range = (1, 5), analyzer = 'char_wb'	longitude	final_text
20	Ridge	alpha = 1	0.506	ngram_range = (1, 7), analyzer = 'char_wb'	latitude	final_text
20	Ridge	alpha = 1	0.6706	ngram_range = (1, 7), analyzer = 'char_wb'	longitude	final_text
21	Ridge	alpha = 1	0.5087	ngram_range = (1, 10), analyzer = 'char_wb'	latitude	final_text
21	Ridge	alpha = 1	0.6744	ngram_range = (1, 10), analyzer = 'char_wb'	longitude	final_text
22	SVR	C = 0.5	0.5217	ngram_range = (1, 10), analyzer = 'char_wb'	latitude	final_text
22	SVR	C = 0.5	0.7170000000000001	ngram_range = (1, 10), analyzer = 'char_wb'	longitude	final_text
23	SVR	C = 0.5	0.5152	ngram_range = (1, 3), analyzer = 'char_wb'	latitude	final_text
23	SVR	C = 0.5	0.7019	ngram_range = (1, 3), analyzer = 'char_wb'	longitude	final_text
24	SVR	C = 0.1	0.5507	ngram_range = (1, 3), analyzer = 'char_wb'	latitude	final_text
24	SVR	C = 0.1	0.7815	ngram_range = (1, 3), analyzer = 'char_wb'	longitude	final_text
25	SVR	C = 1	0.51	ngram_range = (1, 3), analyzer = 'char_wb'	latitude	final_text
25	SVR	C = 1	0.6873	ngram_range = (1, 3), analyzer = 'char_wb'	longitude	final_text
26	XGBRegressor	learning_rate = 0.1, n_estimators = 500	0.5062	ngram_range = (1, 5), analyzer = 'char_wb'	latitude	original
26	XGBRegressor	learning_rate = 0.1, n_estimators = 500	0.6683	ngram_range = (1, 5), analyzer = 'char_wb'	longitude	original
27	XGBRegressor	learning_rate = 0.1, n_estimators = 1000	0.4993	ngram_range = (1, 7), analyzer = 'char_wb'	latitude	original
27	XGBRegressor	learning_rate = 0.1, n_estimators = 1000	0.6559999999999999	ngram_range = (1, 7), analyzer = 'char_wb'	longitude	original
28	XGBRegressor	learning_rate = 0.1, n_estimators = 1000	0.5002	ngram_range = (1, 10), analyzer = 'char_wb'	latitude	original
28	XGBRegressor	learning_rate = 0.1, n_estimators = 1000	0.6567	ngram_range = (1, 10), analyzer = 'char_wb'	longitude	original
29	XGBRegressor	learning_rate = 0.1, n_estimators = 1000	0.5125	ngram_range = (1, 3), analyzer = 'char_wb'	latitude	original
29	XGBRegressor	learning_rate = 0.1, n_estimators = 1000	0.6796	ngram_range = (1, 3), analyzer = 'char_wb'	longitude	original
30	XGBRegressor	learning_rate = 0.1, n_estimators = 1000	0.5227	ngram_range = (1, 1), analyzer = 'word'	latitude	original
30	XGBRegressor	learning_rate = 0.1, n_estimators = 1000	0.6914	ngram_range = (1, 1), analyzer = 'word'	longitude	original
31	XGBRegressor	learning_rate = 0.1, n_estimators = 1000	0.5059	ngram_range = (3, 7), analyzer = 'char_wb'	latitude	original
31	XGBRegressor	learning_rate = 0.1, n_estimators = 1000	0.6689	ngram_range = (3, 7), analyzer = 'char_wb'	longitude	original
32	XGBRegressor	learning_rate = 0.1, n_estimators = 1000	0.5447	ngram_range = (1, 3), analyzer = 'word'	latitude	original
32	XGBRegressor	learning_rate = 0.1, n_estimators = 1000	0.7398	ngram_range = (1, 3), analyzer = 'word'	longitude	original
33	Ridge	alpha = 3	0.5436	ngram_range = (1, 3), analyzer = 'char_wb'	latitude	original
33	Ridge	alpha = 3	0.7369	ngram_range = (1, 3), analyzer = 'char_wb'	longitude	original

id	model	parameters	oof_error	observation	label	text
34	Ridge	alpha = 3	0.5248	ngram_range = (1, 5), analyzer = 'char_wb'	latitude	original
34	Ridge	alpha = 3	0.701	ngram_range = (1, 5), analyzer = 'char_wb'	longitude	original
35	Ridge	alpha = 3	0.528	ngram_range = (1, 7), analyzer = 'char_wb'	latitude	original
35	Ridge	alpha = 3	0.7052	ngram_range = (1, 7), analyzer = 'char_wb'	longitude	original
36	Ridge	alpha = 3	0.5309	ngram_range = (1, 10), analyzer = 'char_wb'	latitude	original
36	Ridge	alpha = 3	0.7094	ngram_range = (1, 10), analyzer = 'char_wb'	longitude	original
37	Ridge	alpha = 5	0.5591	ngram_range = (1, 3), analyzer = 'char_wb'	latitude	original
37	Ridge	alpha = 5	0.7634	ngram_range = (1, 3), analyzer = 'char_wb'	longitude	original
38	Ridge	alpha = 5	0.5421	ngram_range = (1, 5), analyzer = 'char_wb'	latitude	original
38	Ridge	alpha = 5	0.7298	ngram_range = (1, 5), analyzer = 'char_wb'	longitude	original
39	Ridge	alpha = 5	0.5463	ngram_range = (1, 7), analyzer = 'char_wb'	latitude	original
39	Ridge	alpha = 5	0.7357	ngram_range = (1, 7), analyzer = 'char_wb'	longitude	original
40	Ridge	alpha = 5	0.5495	ngram_range = (1, 10), analyzer = 'char_wb'	latitude	original
40	Ridge	alpha = 5	0.7408	ngram_range = (1, 10), analyzer = 'char_wb'	longitude	original
41	LGBMRegressor	n_estimators = 100	0.5479	stacked + transformers (version-7)	latitude	embeddings
41	LGBMRegressor	n_estimators = 100	0.7367	stacked + transformers (version-7)	longitude	embeddings
42	LGBMRegressor	n_estimators = 200	0.5378	stacked + transformers (version-7)	latitude	embeddings
42	LGBMRegressor	n_estimators = 200	0.7193	stacked + transformers (version-7)	longitude	embeddings
43	LGBMRegressor	n_estimators = 300	0.5343	stacked + transformers (version-7)	latitude	embeddings
43	LGBMRegressor	n_estimators = 300	0.7134	stacked + transformers (version-7)	longitude	embeddings
44	SVR	C = 1, kernel = 'rbf'	0.5844	stacked + transformers (version-7)	latitude	embeddings
44	SVR	C = 1, kernel = 'rbf'	0.8515	stacked + transformers (version-7)	longitude	embeddings
45	SVR	C = 5, kernel = 'rbf'	0.556	stacked + transformers (version-7)	latitude	embeddings
45	SVR	C = 5, kernel = 'rbf'	0.7821	stacked + transformers (version-7)	longitude	embeddings
46	SVR	C = 10, kernel = 'rbf'	0.5485	stacked + transformers (version-7)	latitude	embeddings
46	SVR	C = 10, kernel = 'rbf'	0.7628	stacked + transformers (version-7)	longitude	embeddings
47	SVR	C = 20, kernel = 'rbf'	0.5453	stacked + transformers (version-7)	latitude	embeddings
47	SVR	C = 20, kernel = 'rbf'	0.7513	stacked + transformers (version-7)	longitude	embeddings
48	SVR	C = 10, kernel = poly, degree = 2	0.5488	stacked + transformers (version-7)	latitude	embeddings
48	SVR	C = 10, kernel = poly, degree = 2	0.7644	stacked + transformers (version-7)	longitude	embeddings
49	SVR	C = 20, kernel = poly, degree = 2	0.545	stacked + transformers (version-7)	latitude	embeddings
49	SVR	C = 20, kernel = poly, degree = 2	0.752	stacked + transformers (version-7)	longitude	embeddings
50	SVR	C = 10, kernel = poly, degree = 3	0.5466	stacked + transformers (version-7)	latitude	embeddings
50	SVR	C = 10, kernel = poly, degree = 3	0.7555	stacked + transformers (version-7)	longitude	embeddings

id	model	parameters	oof_error	observation	label	text
51	SVR	C = 20, kernel = poly, degree = 3	0.5464	stacked + transformers (version-7)	latitude	embeddings
51	SVR	C = 20, kernel = poly, degree = 3	0.7497	stacked + transformers (version-7)	longitude	embeddings
52	SVR	C = 10, kernel = poly, degree = 4	0.547	stacked + transformers (version-7)	latitude	embeddings
52	SVR	C = 10, kernel = poly, degree = 4	0.7529	stacked + transformers (version-7)	longitude	embeddings
53	SVR	C = 20, kernel = poly, degree = 4	0.5499	stacked + transformers (version-7)	latitude	embeddings
53	SVR	C = 20, kernel = poly, degree = 4	0.7528	stacked + transformers (version-7)	longitude	embeddings
54	Ridge	alpha = 1	0.5509	stacked + transformers (version-7)	latitude	embeddings
54	Ridge	alpha = 1	0.7528	stacked + transformers (version-7)	longitude	embeddings
55	Ridge	alpha = 5	0.5521	stacked + transformers (version-7)	latitude	embeddings
55	Ridge	alpha = 5	0.759	stacked + transformers (version-7)	longitude	embeddings
56	Ridge	alpha = 10	0.5551	stacked + transformers (version-7)	latitude	embeddings
56	Ridge	alpha = 10	0.7664	stacked + transformers (version-7)	longitude	embeddings

Appendix B Level Two Models

id	model	parameters	oof_error	feature_selection	label
0	SVR	C = 1	0.446	All Features -> 57 Features, MinMaxScaler	latitude
0	SVR	C = 1	0.534	All Features -> 57 Features, MinMaxScaler	longitude
1	SVR	C = 0.1	0.4515	All Features -> 57 Features, MinMaxScaler	latitude
1	SVR	C = 0.1	0.5453	All Features -> 57 Features, MinMaxScaler	longitude
2	SVR	C = 0.5	0.4467	All Features -> 57 Features, MinMaxScaler	latitude
2	SVR	C = 0.5	0.5361	All Features -> 57 Features, MinMaxScaler	longitude
3	SVR	C = 5	0.4475	All Features -> 57 Features, MinMaxScaler	latitude
3	SVR	C = 5	0.5325	All Features -> 57 Features, MinMaxScaler	longitude
4	SVR	C = 10	0.4497	All Features -> 57 Features, MinMaxScaler	latitude
4	SVR	C = 10	0.5338	All Features -> 57 Features, MinMaxScaler	longitude
5	HistGradientBoostingRegressor	loss = 'least_absolute_deviation', learning_rate = 0.05, max_iter = 150	0.4488	All Features -> 57 Features, MinMaxScaler	latitude
5	HistGradientBoostingRegressor	loss = 'least_absolute_deviation', learning_rate = 0.05, max_iter = 150	0.536	All Features -> 57 Features, MinMaxScaler	longitude
6	HistGradientBoostingRegressor	loss = 'least_absolute_deviation', learning_rate = 0.1, max_iter = 150	0.4482	All Features -> 57 Features, MinMaxScaler	latitude
6	HistGradientBoostingRegressor	loss = 'least_absolute_deviation', learning_rate = 0.1, max_iter = 150	0.5356	All Features -> 57 Features, MinMaxScaler	longitude
7	HistGradientBoostingRegressor	loss = 'least_absolute_deviation', learning_rate = 0.15, max_iter = 150	0.4497	All Features -> 57 Features, MinMaxScaler	latitude
7	HistGradientBoostingRegressor	loss = 'least_absolute_deviation', learning_rate = 0.15, max_iter = 150	0.5379999999999999	All Features -> 57 Features, MinMaxScaler	longitude
8	HistGradientBoostingRegressor	loss = 'least_absolute_deviation', learning_rate = 0.2, max_iter = 150	0.4518	All Features -> 57 Features, MinMaxScaler	latitude
8	HistGradientBoostingRegressor	loss = 'least_absolute_deviation', learning_rate = 0.2, max_iter = 150	0.54	All Features -> 57 Features, MinMaxScaler	longitude
9	CatBoostRegressor	n_estimators = 900, learning_rate = 0.01, random_state = SEED, silent = True, loss_function = 'MAPE'	0.4511	All Features -> 57 Features	latitude
9	CatBoostRegressor	n_estimators = 900, learning_rate = 0.01, random_state = SEED, silent = True, loss_function = 'MAPE'	0.5429	All Features -> 57 Features	longitude
10	CatBoostRegressor	n_estimators = 900, learning_rate = 0.03, random_state = SEED, silent = True,	0.4475	All Features -> 57 Features	latitude

id	model	parameters	oof_error	feature_selection	label
		loss_function = 'MAPE'			
10	CatBoostRegressor	n_estimators = 900, learning_rate = 0.03, random_state = SEED, silent = True, loss_function = 'MAPE'	0.5347	All Features -> 57 Features	longitude
11	CatBoostRegressor	n_estimators = 900, learning_rate = 0.05, random_state = SEED, silent = True, loss_function = 'MAPE'	0.448	All Features -> 57 Features	latitude
11	CatBoostRegressor	n_estimators = 900, learning_rate = 0.05, random_state = SEED, silent = True, loss_function = 'MAPE'	0.5336	All Features -> 57 Features	longitude
12	CatBoostRegressor	n_estimators = 900, learning_rate = 0.1, random_state = SEED, silent = True, loss_function = 'MAPE'	0.4497	All Features -> 57 Features	latitude
12	CatBoostRegressor	n_estimators = 900, learning_rate = 0.1, random_state = SEED, silent = True, loss_function = 'MAPE'	0.5361	All Features -> 57 Features	longitude
13	CatBoostRegressor	n_estimators = 900, learning_rate = 0.15, random_state = SEED, silent = True, loss_function = 'MAPE'	0.4538	All Features -> 57 Features	latitude
13	CatBoostRegressor	n_estimators = 900, learning_rate = 0.15, random_state = SEED, silent = True, loss_function = 'MAPE'	0.5408	All Features -> 57 Features	longitude
14	RandomForestRegressor	n_estimators = 1000, max_features = 0.7, max_samples = 0.7	0.4546	All Features -> 57 Features	latitude
14	RandomForestRegressor	n_estimators = 1000, max_features = 0.7, max_samples = 0.7	0.5457	All Features -> 57 Features	longitude
15	BaggingRegressor(NuSVR(C = 10, nu = 0.8))	max_features = 0.8, n_estimators = 10	0.4491	All Features -> 57 Features	latitude
15	BaggingRegressor(NuSVR(C = 10, nu = 0.8))	max_features = 0.8, n_estimators = 10	0.5333	All Features -> 57 Features	longitude

Appendix C Submissions Logger

id	train_error_latitude	valid_error_latitude	train_error_longitude	valid_error_longitude	train_error	valid_error	test_error_public	description
1	0.6267	0.6371	0.9431	0.9317	0.7849	0.7844	0.7106	Baseline, just encoding (Multilingual Universal Sentence Encoder V4) and linear stacking on level one with LGBM Meta
2	0.3212	0.4978	0.3846	0.6214	0.3529	0.5596	0.5333	TfidfVectorizer + Level 1 -> LGBM Added + Meta XGB
3	0.3471	0.4881	0.2858	0.5931	0.31645	0.5406	0.5152	Same + Level 1: SVR + RF + ExtraTrees
4	0.2674	0.4906	0.3275	0.6008	0.29745	0.5457	0.5264	Added Lemmatization and German Stopwords
5	0.2377	0.4955	0.2642	0.6063	0.25095	0.5509	0.5379	Added Encodings (USE v4) to standard input
6	0.2607	0.4741	0.3086	0.5822	0.2847	0.5281	0.5144	ngram_range -> (1, 5) and analyzer = 'char_wb'
7	0.2394	0.4734	0.2972	0.5706	0.2683	0.5221	0.5181	removed all emojis, ngram_range -> (1, 7) and analyzer = 'char_wb'
8	0	0.4644	0	0.5644	0	0.5144	0.5002	25 * Ensamble Forrest Models + XGB Meta... Baseline
9	0	0.4531	0	0.5425	0	0.4978	0.4911	Replace XGB with HistGradientBoostingRegressor as meta-learner
10	0	0.4475	0	0.5373	0	0.4924	0.4845	Stacking SVC, RF, XGB with HistGradientBoostingRegressor + StandardScaler for SVC
11	0	0.4435	0	0.5299	0	0.4867	0.4732	Adding 15 level one learners on original data for variance, baseline
12	0	Nan	0	Nan	0	Nan	0.4723	Bayesian Optimization Mean Stage 1
13	0	0.4422	0	0.5254	0	0.4838	0.4706	Up to 57 estimators to level one, and separated level two with 5 SVR, 4 Hist and VotingRegressor as head
14	0	0.4412	0	0.5247	0	0.4829	0.4715	5 SVR, 4 Hist, 5 Cats, 1 RF, 1 BaggingNuSVR and VotingRegressor as head + finetunning
15	0	0.4411	0	0.5245	0	0.4828	0.4711	Bayesian Optimization for parameters
16	0	Nan	0	Nan	0	Nan	0.4705	Bayesian Optimization Mean Stage 2

Appendix D Submission Blendings Logger

id	test_error_public	description
1	0.492	submission_3 * 0.5 + submission_6 * 0.5
2	0.4926	submission_3 * 0.4 + submission_6 * 0.6
3	0.4932	submission_3 * 0.6 + submission_6 * 0.4
4	0.4915	submission_3 * 0.5 + submission_6 * 0.25 + submission_7 * 0.25
5	0.4919	submission_3 * 0.5 + submission_6 * 0.2 + submission_7 * 0.3
6	0.4912	submission_3 * 0.5 + submission_6 * 0.3 + submission_7 * 0.2
7	0.4896	submission_8 * 0.5 + submission_3 * 0.3 + submission_6 * 0.1 + submission_7 * 0.1
8	0.4909	submission_8 * 0.5 + submission_3 * 0.5
9	0.4836	submission_9 * 0.6 + submission_3 * 0.2 + submission_6 * 0.1 + submission_7 * 0.1
10	0.4795	submission_10 * 0.6 + submission_3 * 0.2 + submission_6 * 0.1 + submission_7 * 0.1
11	0.4796	submission_10 * 0.5 + submission_9 * 0.1 + submission_3 * 0.2 + submission_6 * 0.1 + submission_7 * 0.1
12	0.4721	submission_11 * 0.6 + submission_3 * 0.2 + submission_6 * 0.1 + submission_7 * 0.1
13	0.4705	submission_12 * 0.7 + submission_3 * 0.2 + submission_6 * 0.05 + submission_7 * 0.05
14	0.4691	submission_13 * 0.7 + submission_3 * 0.2 + submission_6 * 0.05 + submission_7 * 0.05
15	0.4701	submission_14 * 0.7 + submission_3 * 0.2 + submission_6 * 0.05 + submission_7 * 0.05
16	Na	Submission_16 * 0.7 + submission_3 * 0.2 + submission_6 * 0.05 + submission_7 * 0.05
17	0.4711	Submission_16 * 0.7 + submission_3 * 0.1 + submission_6 * 0.1 + submission_7 * 0.1
18	0.4705	Submission_16 * 0.7 + submission_3 * 0.3