# PROJECT 1 PML

Ghadamiyan Lida, Class 407 AI 03.01.2021

## Estimate the latitude and longitude of German Tweets

This project consists in trainig a regressor for predicting the latitude and longitude of german tweets. The final score is calculated using mean absolute error.

## 1 Choosing the model

For my final submissions I chose to use Ridge Regressor and KNN regressor after multiple attempts. I tried with the following regressors: RandomForest, KernelRidge, MLP, XgBoost, DecissionTrees and SVM with default hyperparameters on the validation data. Based on the results I retained Ridge, KNN and RandomForest.

I considered the latitude and longitude independent one from the other and predict each of them with its own model in order to be able to keep the best approach.

## 2 Implementation

I treated the data set as data frames using the pandas library. I loaded the data with pd.read_csv and header set to 'None' because the data contains only the values. The validation data needed latin-1 encoding for some special characters.

For feature extraction I considered the following scikit-learn tutorial

https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html. I created a vocabulary containg just unigrams by finding the occurrence of every word using CountVectorizer. Then, I used TfidfTransformer to obtain the term frequencies by dividing the occurrences by the vocabulary length.

The documents were lowered and the punctuation and words shorter than two characters were removed with CountVect. I chose not to remove the emojis as I considered them useful for differentiating users. For example, people from a location next to a river or a lake might use the beach related emojis more than others in more urban areas.

I trained the model on the training data and evaluated it on the validation data. I applied the same procedure for feature extraction to each file (train, validation and test). For the Kaggle submissions I trained the model on a data set formed by concatenating the train and validation data. This data set was also used for the grid search.

## 3 Model evaluation

The mean absolute error represents the average distance from the predicted values to the real ones and the mean squared error is the average squared distances. If the distance is bigger, the MSE will increase more than MAE would do for the same error.

The MAE results on the validation data using tunned hyperparameters are:

| Regressor | Latitude | Longitude |
|---|---|---|
| Ridge | 0.5221330793452222 | 0.6974568486309981 |
| KNN | 0.5659368074168913 | 0.7626482204097159 |
| RandomForest | 0.7086745163062681 | 1.047419516152995 |

Considering these results I chose Ridge for both latitude and longitude for my first submission.

The overall MAE for both latitude and longitude is the mean of the MAE for each of them.

The MSE results on the validation data using tunned hyperparameters are:

| Regressor | Latitude | Longitude |
|---|---|---|
| Ridge | 0.46333504055507857 | 0.8202959928306802 |
| KNN | 0.550244300486255 | 0.9688341478833623 |
| RandomForest | 0.7968375490950509 | 1.61135700202054 |

Regarding these tables it is obvious that Ridge is better at both Latitude and Longitude, and KNN is the second best. This means that the coordinates will be predicted with one model. I chose to continue with two separate models because using Multioutput Regressor gave the same results but with more lines of code. Another reason is that I want to have the freedom of choosing different parameters for latitude and longitude if it will be necessary.

The increased differences between the MSE and MAE for KNN and RandomForest show that the errors are larger. So, the Ridge Regressor seems to be the best choice.

## 4 Hyperparameters tunning

For the hyperparameters tunning I created a pipeline with the Countvectorizer, tfitfTransformer and the chosen Model. I ran a gridsearch using the negated mean absolute error for metric. The negation comes from the convention conform to which higher values are better. Another parameter of the gridsearch is cross-validation generator, that shuffles the data randomly and then splits it in K folds and use k-1 of them for training and 1 for testing until all of them were used for testing. The cross validation technique helps to avoid overfitting because all the folds are used for testing while the score is kept and the model is discarded. In the end it summarizes the performance of the model using the metric given, in this case the mae.

After performing the grid search, the best parameters for the Ridge Regressor were the following:

```
Best parameters 1st model :  {'Model_LR1__alpha': 1, 'cvect__ngram_range': ↩
    (1, 1), 'tfidf_transformer__norm': 'l2', 'tfidf_transformer__use_idf': ↩
    False}
Best parameters 2nd model :  {'Model_LR1__alpha': 1, 'cvect__ngram_range': ↩
    (1, 1), 'tfidf_transformer__norm': 'l2', 'tfidf_transformer__use_idf': ↩
    False}
```

The same procedure was applied for all the models, resulting in different parameteres.

The best score returned by the grid search for the first latitude is 0.5136244050757868 and for the longitude 0.6806272277821972. These results are slightly better that the one obtained on the validation data, but it is clear that the final submission score will be close to this. So, the final submission would be close to (0.51+0.68)/2 = 0.595.

The score at the end of the competition was 0.597, so the strategy functioned.

## 5 Conclusion

I believe that the most important thing in this competition was to know how the model should behave and what is its average score, in order to be able to tell if the shown score on 30 percent of the testing data is relevant or not.

I tried to follow this reasoning, and what I expected for each model was close to the final score. It also helped me find the best model, in order to be able to select the best two final submissions.