Beginning:

Collecting the data from twitter:

After some research we found a file containing all twitter account names for members of the German Bundestag with their respective party. This file had some formatting errors which we had to clean up before using it.

For downloading the required data from twitter we decided to use the opensource framework selenium in combination with google chrome as web browser. This setup allowed us to download as many tweets as we wanted with all of the metadata visible on Twitter.

Because the twitter page is not loaded all at once but continuously as you scroll down, we had to imitate a real user scrolling down the page. For implementing this, we used selenium to scroll to the end of the current view and wait a certain amount of time for the page to load new content. To find the ideal time to wait we had to assess the trade-off between increasing the runtime and the risk of losing some tweets because they weren’t loaded in that short amount of time. We still noticed that we didn’t get all of the tweets in the first run so we decided to do another run in a different network and combine the results.

(Selenium Part 3): Here you can see our setup in action while it is scrolling down the page and saving the data.

Here is an overview of the two runs which shows doing a second run increased the total number of tweets by 5%. Because this value is so low, we assume that we got nearly all the data.

For every politician we created an own CSV-file containing the username, post date, text of the tweet, the number of replies, retweets and likes. From these files we created several numpy arrays for training and testing, in which we deleted all retweets to guarantee that all texts were written by the politician.

(Parties): In our first step we wanted to train a model to detect which political party the sender of the tweet belongs to. The model therefore has to process the text of a tweet and predict to which of the following seven parties the tweet belongs to: ['CDU', 'LINKE', 'FDP', 'GRÜNE','SPD', 'CSU', 'AFD'] .

(Keras\_Tuner\_Convusion): To tune the hyperparameters for our model we decided to use the keras tuner. The result we got was 16500 neurons for the hidden layer and a learning rate of 0.001. We then used these hyperparameters to fully train a model, which reached an accuracy of about 55%. This is quite a high value considering the relatively small differences between certain parties. To visualize the difficulties the model had with classifying the tweets, we used it to predict the party for every tweet in our test dataset. Out of these results we created the following confusion matrix. It shows that the model has the highest accuracy for identifying tweets belonging to the AfD party. Tweets from the party “Bündnis/90 die Grünen” however seem harder to classify as the model confuses a relatively large percentage with the SPD and LINKE. It also shows that the model is biased against the CSU which we already expected since there are way less tweets from the CSU in our dataset than from the other parties.

(LKR\_Explenation\_Big5\_start): We were also curious how our model would respond to tweets by members of the LKR party which it had never seen before. The result is very interesting as the model can’t clearly decide to which party the LKR-tweets belong to. Looking up which political orientation the LKR has, we found out that it characterizes itself as a party of the center which matches the prediction of our model.

In the next part of our project we wanted to use our dataset to analyse the Big5 personality traits of the politicians and examine the differences. We were also interested in whether there is a noticeable difference between the parties.

We decided to train a model using a data set containing English essays with their big5 personality trait scores. We used that English dataset because we couldn’t find any German dataset containing psychology data about texts. To carry out the big5analysis on our German tweets we therefore had to translate our data into English. Because these translated texts could have less information, we had to check if the big5 analysis is also possible on machine translated data. For this purpose we needed some sort of bi-directional translation between German and English. The first thing we thought of was training the transformer model we discussed during the meetings on translated TED talk transcripts.

Furthermore we had to create our own tokenizer for the German language since the TensorFlow tutorial only provided a Portuguese and English tokenizer.