

People's Representatives - Attendance at the 19. Bundestag

Michael Minzlaff

December 2022

Executive Summary

Members of Parliament (MP, in German: Mitglieder des Bundestags, MdB) are elected to represent, and act on behalf of the people living in Germany. Their work as elected representatives is paid for by taxes raised by the federal government. It seems only fair therefore to ask how seriously they take their responsibilities.

My analysis is focused on the 19th legislative period of the Bundestag, which ran from 24 October 2017 to 26 October 2021. I show that overall attendance was high, but that there were significant differences between parties, as well as for party leaders and government ministers.

I chose to address the question whether a no-show vote can be predicted, by answering two questions in particular:

1. for any given MP and poll, is it possible to predict whether this MP is going to no-show at the poll?
2. (supplementary question) is it possible to predict which MPs are most likely to be a frequent no-show?

I found that while it is possible to make some predictions for MPs no-show behaviour, this requires costly trade-offs between precision (a combination of MP and poll correctly identified as a no-show) and recall (share of all no-shows identified). Furthermore, I also found that knowledge of which MPs are most likely to no-show (i.e. question 2) does not add sufficient predictive power to be worth the effort.

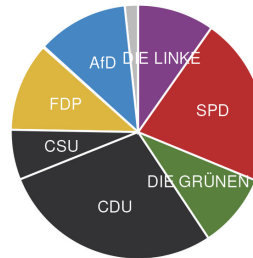
Future work in this area would benefit from a deeper investigation of MP's 2nd jobs (i.e. outside of parliament) as well as a more detailed review of MPs motivations, i.e. they may not expect a continued career in parliament.

Introduction

For this final capstone assignment of the HarvardX Data Science Professional Certificate Programme, I chose to review attendance in the German federal parliament ('Bundestag') during the recently ended 19th legislative period. The 19th legislative period of the Bundestag began on 24 October 2017 and ended with the constituent meeting of the newly elected MPs on 26 October 2021.

The 19th Bundestag was constituted of 7(6)¹ parties: CDU/CSU, SPD, FDP, Die Grünen, Die Linke, AfD.

Composition of parliament



While key debates and votes are generally covered in some depth in the media, much less visibility is given to the level of attendance / absences in parliament. Only keen observers of contemporary news reporting would notice that, quite frequently, parliamentary debates are attended only by a smaller number of MPs.

As we will see below, absences at votes are common. Many reasons may be behind this - illness, obligations as government minister, etc.. However, no further details are given as of the time of this report.

Interestingly there is also a noticeable proportion of votes that are ‘no shows’, i.e. absences without explanation from the MP.

To the best of my knowledge, no systematic analysis of the patterns of attendance in parliament has been done. This appears to be a significant omission: participating in the parliamentary debate is a key task for MPs and signals their awareness of and support for the rules and principles of parliamentary democracy.

The data for this assignment comes from [abgeordnetenwatch.de](https://www.abgeordnetenwatch.de), which was set up to offer an API to access data on politicians, members of parliament, parliamentary periods, motions and votes, and committee memberships in detail and going back 10 parliamentary periods to 2005.

The data

The data for this project is available via an API by <https://www.abgeordnetenwatch.de/api>. This API is based on HTTP GET requests to API URLs that reference specific data entities:

Entity	Description
Parliament (*)	Represent an actual parliament. The API offers access to both Federal and State / Regional parliaments
ParliamentPeriod (*)	Defines a specific parliamentary period, which can be an election or a legislature. The API covers both current parliamentary periods (e.g. for the current Federal parliamentary period that started on 18899 and ends later this year on 20387), as well as previous periods going back to 13044.
Politician (*)	Master data on politicians is stored here. The candidacies and mandates of individual politicians are stored in a separate entity, the CandidacyMandate entity, in which there is a reference to the politician.
CandidacyMandate (*)	Data related to candidacies and mandates is somewhat more complex, which is why there are two “sub-entities”: ElectoralData, and FractionMembership

¹CDU and CSU are separate parties but stand in elections - and sit in parliament as joint parliamentary groups

Entity	Description
ElectoralData (*)	Data such as constituency, electoral list, constituency result are stored here.
FractionMembership	Only relevant for mandates. During a mandate, the parliamentary group can be changed. In order to be able to map this, the caucus membership is maintained via this entity, in which entry and exit dates can also be maintained for each individual membership.
Committee (*)	Parliamentary committees
CommitteeMembership (*)	MP membership of parliamentary committees
Poll (*)	Data on votes called in parliament
Vote (*)	Voting data for each individual MP
Party (*)	Political parties
Fraction (*)	Parliamentary groups of each of the political parties
ElectoralList	Electoral list data. Electoral lists are always linked to a legislature or election, so there can be several electoral lists with the same name.
Constituency (*)	An MP's constituency
ElectionProgram	References to each of the parties' election manifesto
Sidejob	An MP's declared outside job
SidejobOrganization	The organisation where an MP's declared outside job is
Topic (*)	Multi-level classification of poll subjects
City	An MP's city of residence (where the MP has provided this data)
Country	An MP's country of residence (where the MP has provided this data)

The entities marked with (*) are used in my analysis.

There are two additional datasets that I draw on:

1. a file with geometric shape data: I use this to draw a map of voting districts
2. a mapping of voting districts codes and IDs

ETL approach and scripts

I created a number of R files for this project, each for specific steps in the process:

Table 2: Key R files

File	Purpose
00 Setup.R	<ul style="list-style-type: none">• Downloads a LUA filter script to facilitate multi-column R report creation
01 GetData.R	<ul style="list-style-type: none">• Retrieve data from the abgeordnetewatch API. Implements a multi-stage caching of interim files to minimise API load
02 TidyData.R	<ul style="list-style-type: none">• Reformatting and cleaning: the raw data retrieved from abgeordnetewatch is - depending on the entity - deeply nested and not directly usable for analysis. Type-casting to ensure consistency and facilitate analysis (e.g. dates)
103 Initial exploration.R	<ul style="list-style-type: none">• Exploratory data analysis (EDA): Reviews and analyses the data downloaded from abgeordnetewatch and draws out key insights
104 Predictions v2.R	<ul style="list-style-type: none">• Machine learning: Attempts to predict which MPs are most likely to have a high(-er) share of no-show votes. Attempts to predict which polls an MP will no-show.
19 Report.Rmd	<ul style="list-style-type: none">• This report

These draw on a small number of common helper files, namely:

Table 3: Supporting functionality

File	Purpose
Definitions.R	Common constants and definitions, e.g. folder names
Settings.R	Global variables and settings
getJSON.R	Retrieve (multi-page) JSON data
SupportFunctions.R	Other (additional) supporting functions
Colours.R	Standard colours to use for each of the parties and their parliamentary groups ('Fraktionen')
VotePie.R	Draw two-level donut / pie charts

Extract: data retrieval

The process of data collection is structured as follows: 01 `GetData.R` starts by retrieving the ID for the current parliamentary period of the Federal parliament (`Bundestag`), and then proceeds to download data on MPs of the current parliament (`CandidacyMandate`), details for each MP (`Politicians`), parliamentary committees and their members (`Committee` and `CommitteeMembership`), parties and fractions (`Party` and `Fraction`), and finally voting data (`Polls` and `Topic` as well as `Vote`).

To avoid unnecessary API requests, 01 `GetData.R` uses the `getJSON()` helper function (provided in `GetJSON.R`) that requires a filename, a directory path, and an API URL. It uses these inputs to then check whether the relevant data has already been downloaded (in which case it would open the relevant file and return its content), or retrieve the API result from that URL and store it in a file. Either way, `getJSON()` returns the requested data. In cases where the API response indicates that more results are available than have been returned (paged data), `getJSON()` repeats the API call with `&page=` and a page number appended to each call until all results have been obtained. Finally, 01 `GetData.R` writes the collected data to storage as serialised R objects.

```
# Get a reference to the federal parliament ('Bundestag')
print('Fetching reference to current parliamentary period\n')
parliamentsAPI <- c(name = 'parliaments.JSON',
                    url = 'https://www.abgeordnetenwatch.de/api/v2/parliaments')
parliaments <- getJSON(parliamentsAPI['name'],
                      parliamentsAPI['url'],
                      path = folderJSON)
BundestagID <- parliaments[parliaments["label"] == "Bundestag",]$id
currentBundestagID <- parliaments[parliaments["id"] == BundestagID,$current_project$id

# Get a list of all parliamentary periods for the federal parliament available
# via this API
print('Getting list of earlier parliamentary periods\n')
PeriodsAPI <- c(
  name = 'periods.JSON',
  url = paste0(
    'https://www.abgeordnetenwatch.de/api/v2/parliament-periods?parliament=',
    BundestagID
  )
)
Periods <- getJSON(PeriodsAPI['name'],
                  PeriodsAPI['url'],
                  path = folderJSON)
save(Periods, file = paste0(folderRData, 'Periods-RAW.RData'))

previousBundestagID <- Periods[Periods$type == 'legislature',] %>%
  slice(2) %>%
  pull('id')

# Get a list of all Members of Parliament in the previous legislative period
print('Downloading list of MPs\n')
MPsAPI <- c(
  name = 'mps.JSON',
  url = paste0(
    'https://www.abgeordnetenwatch.de/api/v2/candidacies-mandates?parliament_period=',
    previousBundestagID
  )
)
```

```
MPs <- getJSON(MPsAPI['name'],
               MPsAPI['url'],
               path = folderJSON)
save(MPs, file = paste0(folderRData, 'MPs-RAW.RData'))
```

Transform: data cleaning

Data cleaning primarily takes place in 02 TidyData.R. The script first retrieves the downloaded raw API data from file and unpacks and unnests all columns that contain a data.frame or a list. The script then converts columns containing dates to Date format, and all columns containing IDs to numeric.

Following this, the script creates a few additional columns (such as age, derived from year of birth) that will be needed in subsequent analyses.

Finally, I create stripped-down ‘skeleton’ versions of certain datasets that contain only the core columns required for linking between entities. While this appeared useful at the beginning, I ended up making less use of it than I thought.

The tidied and stripped-down data frames are saved back to disk as RData files. The code snippet below illustrates this process using the example of the `CandidacyMandate` entity.

```
load(file = paste0(folderRData, 'MPs-RAW.RData'))
MPs <- MPs %>% unpack(c(parliament_period, politician, electoral_data),
                    names_sep = '.')
MPs <- MPs %>% unpack(c(electoral_data.electoral_list,
                      electoral_data.constituency),
                    names_sep = '.')
MPs <- MPs %>% unnest('fraction_memberships', names_sep = ".", keep_empty = TRUE)
MPs <- MPs %>% unpack('fraction_memberships.fraction', names_sep = ".")
MPs$start_date <- as.Date(MPs$start_date)
MPs$end_date <- as.Date(MPs$end_date)
MPs <- MPs %>% mutate_if((str_detect(names(.), "id$")), as.numeric)
save(MPs, file = paste0(folderRData, 'MPs.RData'))

skMPs <- MPs %>% select(id, label, politician.id, politician.label,
                      electoral_data.id, electoral_data.electoral_list.id,
                      electoral_data.constituency.id, fraction_memberships.id,
                      fraction_memberships.fraction.id, start_date, end_date)
save(skMPs, file = paste0(folderRData, 'skMPs.RData'))
rm(list = c('MPs', 'skMPs'))
```

Exploratory analysis

Members of Parliament (MP, in German: Mitglied des Bundestags, MdB) are elected to represent and act on behalf of the people living in Germany. It seems fair to ask how seriously they take their responsibilities.

In this analysis, we need to exclude those persons who are no longer in parliament, e.g. because of illness or death. A Google search with their name usually provides context for readers who are curious for more background, but this is not relevant here.

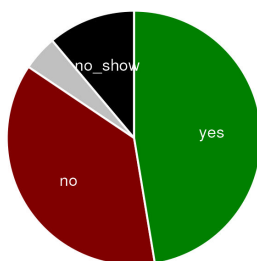
```
inactiveMPs <- skMPs %>%
  filter(!is.na(end_date)) %>%
  select('id')
```

I investigate how attendance of MPs remaining active for the entire legislative period has developed over the course of the 19th parliament. I then also look at absenteeism by gender, age, tenure, party membership, MP's 'security' in their seat (i.e. popular support in their constituency), and topic of each motion voted on by parliament.

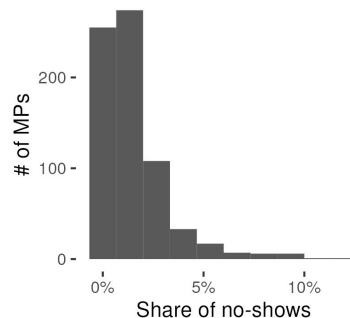
Overall attendance

Overall attendance at parliamentary debates is high, with - on average 89% participation in votes. This average hides significant differences between MPs, parties, and over time. Looking at the frequency distribution of no-shows, let's focus on those MPs that have a no-show share larger than 5%.

Voting behaviour



Frequency of no-shows



Drill-down into attendance by MP

1. As becomes immediately apparent, government ministers have a higher share of missed votes than most MPs:

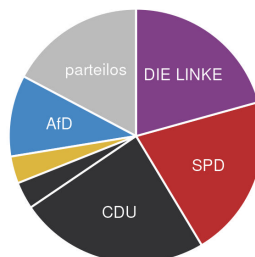
Table 4: Top MPs by absence

Name	Occupation	Party	Share of polls missed
Sylvia Gabelmann	MdB	DIE LINKE	12%
Angela Merkel	Bundeskanzlerin	CDU	11%
Uwe Kamann	MdB, Unternehmer	parteilos	10%
Helge Braun	Bundesminister für besondere Aufgaben und Chef des Bundeskanzleramts, MdB	CDU	10%
Pia Zimmermann	MdB	DIE LINKE	10%
Sahra Wagenknecht	MdB	DIE LINKE	9%

The top 'no-shows' include (appropriately?) the Head of Government, Chancellor Angela Merkel and Dr. Helge Braun (Head of the Chancellery and Federal Minister for Special Affairs), but also highlight a pattern that we will see in more detail below, i.e. that MPs not affiliated with any party ('parteilos') or with parties on the left or right fringes of the political spectrum are more likely to be no-shows than MPs from mainstream parties.

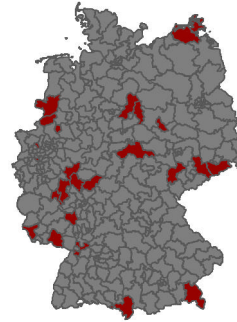
2. More than half of all no-show MPs are from fringe parties on either the left or the right of the political spectrum, or from MPs no longer affiliated with any party. Let's look at their characteristics in a bit more detail.


No-shows by party affiliation



3. I wanted to see *where* these MPs' constituencies were - whether there was any discernable regional pattern. And, to be honest, I also wanted to find out how to create maps in R. It turns out to be a lot easier than I thought. However, as the graphic on the right shows, there didn't seem to be any obvious pattern.

MP constituencies



 No-show constituency

```
# QUESTION Where are they from?
load(paste0(folderRData, 'constituencies.RData'))
shpConstituencies <- st_read(paste0(folderTemp,
                                     'Geometrie_Wahlkreise_19DBT_VG250.shp'))

nsConstituencies <- shpConstituencies %>%
  left_join(constituencies, by=c('WKR_NR' = 'number')) %>%
  left_join(nsMPs, by=c('id' = 'electoral_data.constituency.id')) %>%
  select(c('politician.label', 'geometry', 'isNS'))

nsConstituencies[is.na(nsConstituencies$isNS), 'isNS'] <- FALSE

gMap <- nsConstituencies %>% ggplot() +
  geom_sf(aes(fill=isNS), size=0.1) +
  coord_sf(label_axes='') +
  scale_fill_manual('No-show MP constituency',
                    values=colHighlight2,
                    limits=c(TRUE),
                    labels=c('No-show constituency')) +
  ggtitle('MP constituencies') +
  theme(panel.background = element_rect(fill = "transparent", colour = NA),
        plot.background = element_rect(fill = "transparent", colour = NA),
        legend.position='bottom',
        legend.title=element_blank())

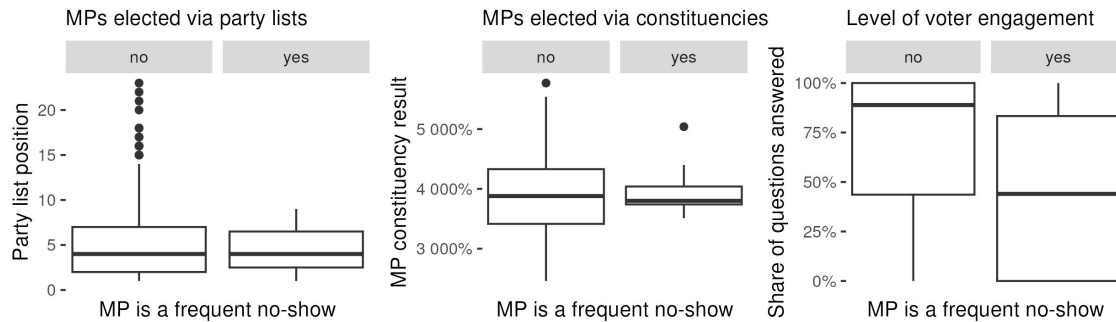
ggsave(plot=gMap, width=7, height=7, units='cm', dpi='print', path=folderImages,
        filename='Constituency map.jpg')
```


4. Next, I took a look at the power base of each of these MPs. My hypothesis was that, given a stronger position within the party or their constituency, MPs would be more likely to have a higher share of ‘no-shows’ for two reasons:

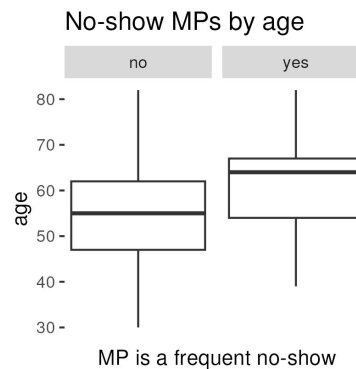
- There should be higher demand on their time from party offices
- There should be less pressure to show engagement

As the images below illustrate, there is only limited support for my hypothesis: stronger constituency results *tend to* be associated with a higher share of no-shows, but overall there are only few differences between the more and less active MPs along this dimension.

No-show MPs by strength of their power base in party & constituency and engagement with the public



5. Finally, let’s look at their demographics (age, gender). Again there seem to be only minor differences between those MPs who attend most polls, and those who are more likely to be absent: 3.7% of male and 4.9% female MPs are (more) frequently absent at polls than their peers.



Attendance by topic

In this section, I ask what, if anything, we can learn about the polls that have the highest share of no-shows (i.e. topics, dates, # of polls on the day).

To start, let’s identify those polls that have the highest share of no-shows.

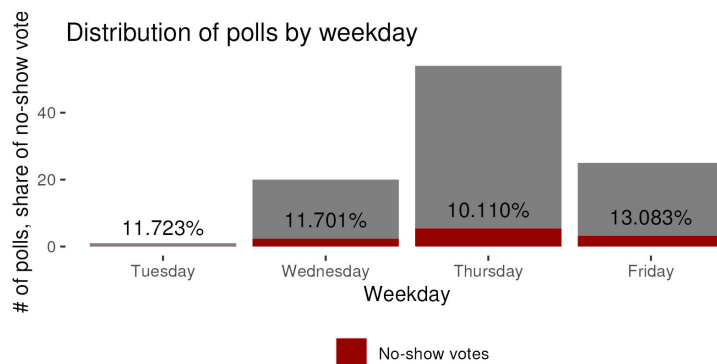
As we have seen above, the average share of no-show votes across all polls is about NA. I therefore now focus on polls where the share of no-show votes is above this threshold:

```
skActiveVotes <- activeVotes %>%
  select(c('id', 'mandate.id', 'poll.id', 'vote'))

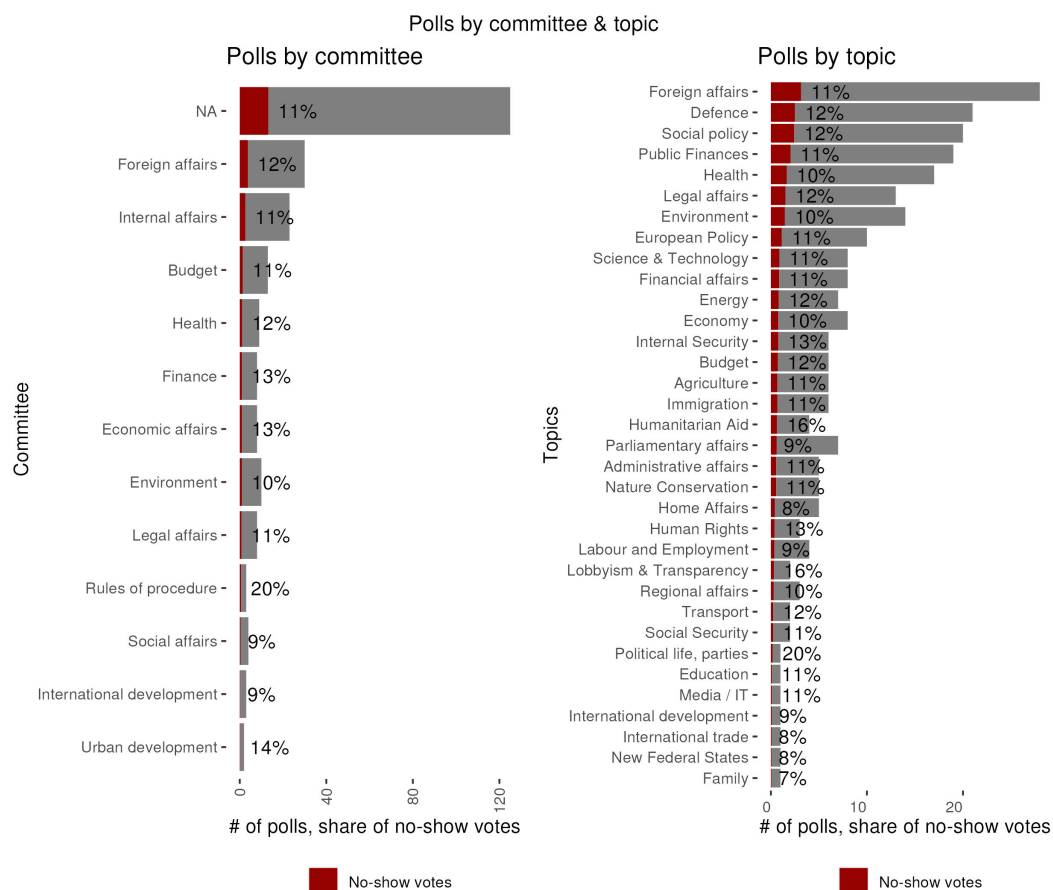
nsVotes <- skActiveVotes %>%
  group_by(poll.id, vote) %>% summarise(n=n()) %>% ungroup() %>%
  group_by(poll.id) %>% mutate(N=sum(n), s=n/N) %>%
  filter(vote == 'no_show') %>% select(c('poll.id', 'n', 's'))

nsVoteIDs <- nsVotes %>% filter(s > dictKeyStats['AvgNSSharePerPoll']) %>%
  select('poll.id')
```

6. The majority of polls is held on Thursdays and Fridays, i.e. at the end of the working week. However, Thursdays are the weekday with the highest share of no-show votes. This could at least in part be because Friday is likely to be seen as the start of the weekend by some MPs, and with a lower probability of polls scheduled for Fridays, some MPs ‘start’ their weekends on Thursdays already. Note that there does not seem to be a strong effect of the # of polls per day on MPs attendance.



7. Last but not least, I also investigated whether certain topics might result in a higher share of no-show votes. As the graph (below) shows, procedural affairs and foreign affairs are the areas most affected.



With this, I conclude the exploratory analysis. The following section of this paper reports on my approach and results predicting no-show behaviour.

Predicting no-show behaviour

I chose to focus on predicting no-show behaviour in the most recently completed parliamentary period. Following the exploratory analysis above, I expect difficulties in getting strong prediction results. I therefore decided to split the task into two questions:

1. predict for each MP whether they are likely to be a frequent no-show at polls
2. predict for each MP and each poll whether the MP is likely to vote no-show

Step 1 would - that, at least was my plan - give me a valuable additional data point to work with for answering the second question. Unfortunately it turned out that such a no-show flag does not add sufficient additional data, and I ultimately discarded this approach in deriving my final result.

To facilitate the analysis, I created a function `doTestMethod()` which I call repeatedly with different parameters to split, train, and test a dataset, and that calculates precision + recall performance. This function takes a number of inputs:

- a dataframe with the data to run predictions on
- a list of column names that contain the predictors
- a character string with the name of the column that is to be predicted
- the name of the machine learning model to use
- the name of the sampling method to use
- a version tag number
- an *optional* list of parameters that is passed through to the training function

I **apply** this function to a list containing parameter sets for sampling and training. The output - a list of precision & recall results - is then stored for assessment and use in this report.

By default, `doTestMethod()` stores the results of previous runs, and will draw on these if called again.

Predicting whether any given MP is likely to be a frequent no-show

Building on the exploratory analysis of the previous section, I flagged MPs with a share of no-show votes equal to or larger than 5% as `isNSMP`.

The list of MPs was split into two equally sized training and testing sets. Given the small number of flagged MPs, sampling was critical to ensure balanced datasets for both training and testing.

My analysis showed that `up` worked best for this problem, and that no significant benefit was to be gained from switching to either `rose` or `smote` (in fact, using either caused problems with training in most cases). Unsurprisingly, downsampling did not perform as well since this resulted in a very small overall training / testing datasets.

I applied a range of different ML algorithms to the problem, and assessed both precision (the share of correctly identified no-show MPs in the total list of MPs predicted by the algorithm) and recall (the share of true no-show MPs identified by the algorithm).

As the table below illustrates, results were not great:

Table 5: Predicting which MPs are most likely to vote no-show

Training	glm	rda	bayesglm	lda	knn	cforest	treebag
Sampling	up	up	up	up	up	up	up
Precision	10.4%	10.4%	13.0%	12.2%	5.8%	9.4%	21.4%
Recall	35.7%	57.1%	50.0%	64.3%	64.3%	35.7%	21.4%

Predicting whether any given MP is likely to no-show for a given poll

I tested whether predicting a no-show vote (`isNSVote`) was feasible both with and without reference to the `isNSMP` flag that identifies those MPs most likely to be a no-show. My hope was that this flag would help in

predicting no-show behaviour.

Unlike the MP dataset used so far, the poll and voting dataset includes a number of categorical predictors. I ‘one hot’ encoded the relevant columns of the dataframe before running the training task.

1. Prediction without the `isNS` flag

As in the previous section, I again tested a number of ML algorithms on this problem. For the sake of limiting computing time, I however chose a reduced set of methods: `glm`, `lda`, and `bayesglm`, all with `up` as sampling algorithm. The table below summarises the results of this analysis:

Table 6: Predicting no-show voting behaviour

Training	glm	lda	bayesglm
Sampling	up	up	up
Precision	31.2%	31.0%	NA
Recall	67.6%	68.0%	NA

2. Prediction including reference to the `isNSMP` flag

Surprisingly, adding the `isNSMP` flag for calling out those MPs most likely to no-show at polls did not improve prediction performance. I hypothesise that the number of MPs identified with this flag is too small (or, in other words: the number of no-show votes attributed to the `isNSMP` MPs is too small) to make a big difference.

Table 7: Predicting no-show voting behaviour, with a flag for top no-show MPs

Training	glm	lda	bayesglm
Sampling	up	up	up
Precision	31.2%	31.0%	NA
Recall	67.6%	68.0%	NA

Summary and Conclusions

My analysis shows an overall good level of engagement by Members of Parliament in the decision making processes. There are three groups of MPs who are conspicuous in being absent more frequently: senior members of the government, MPs not affiliated with any party, and MPs from the left and right fringes of the political spectrum. While it is possible to speculate what motivation (or lack thereof) drives this behaviour, this is outside the scope of this project.

The analysis also shows that it is possible to predict no-show behaviour with some accuracy (in particular in light of the very low prevalence of no-show at polls), albeit at a significant cost in precision: in other words, a high rate of false positives.

Next steps

Next steps, in terms of this project, would be to drill into more detail of the performance of each of the ML algorithms to better understand what opportunities might exist to improve their predictive power with this dataset and - in parallel - to develop an intuition to help predict which algorithm(s) are most likely to deliver a strong result.

In parallel, it would be interesting to extend this analysis to probe the motivations of MPs for their absences at polls. Sadly, no data on this is (currently?) provided via `abgeordnetenwatch`.