Introduction to Data Science: Final Paper

Polling success in various policy fields, using the example of referendums in the Canton of Zurich in the period 2000 to 2023

David Rothschild

Lecturer: Prof. Dr. Marco Steenbergen

17.01.2023

Introduction

My research question is: "How does the approval of voters in the canton of Zurich in a referendum differ in different policy areas?

My approach

I have analysed all votes in the canton of Zurich since 2000 and assigned them to a general policy area. I assigned proposals that I could not categorise to the collective category "Other".

I differentiated between the following policy areas:

- Umwelt und Verkehr (Environment and transport): I deliberately grouped the bills on transport policy issues and environmental issues into one category, as they were often bills that concerned both sets of issues, e.g. expansion of the Limmat Valley railway (see(SRF, 2018)).

- Gesundheit (Health): This includes issues such as hospital financing or referendums on federal COVID laws.

- Soziales (Social affairs): These are policies that affect the welfare state.

- Wirtschaft (Economy): These are issues that affect the economy, such as shop opening hours.

- Demokratie (Democracy): I also include changes to the cantonal constitution.

- Kirche und Staat (Church and state) : This includes votes such as the ban on minarets or the ban on veiling, but also votes on marriage for all.

- Migration (Migration) : Votes on immigration and asylum.

- Weitere (further) : 39 of the 409 votes could not be assigned to any of the above categories, e.g. "Amendment of the Federal Act on Film Production and Film Culture".

Most proposals were put to the vote on the topics of environment, transport and social affairs.

A more detailed analysis can be found below.

My contribution

So far, there has been little research on how approval of proposals differs across policy areas.

Previous research has analysed the motives of voters or the influence of political parties in detail, see for example Milic (2010) or Kriesi (2012).
For the first time, I would like to investigate whether there are differences in the policy areas on which voters vote.
Unfortunately, the data do not provide information on whether these are initiatives or referendums.

Theory

The underlying concept is that I assume that the electorate behaves differently with respect to different proposals. In particular, polarisation can be observed in social policy (cf. Afonso and Papadopoulos, 2015).

The policy areas are categorical variables, with approval at the ballot box taking into account both the acceptance and rejection of a proposal categorically.
In addition, I consider the nominal category of how many municipalities approved a proposal and how many did not.
Another variable in this context is the approval rate in percent for the entire canton.

The characteristics that I take into account in the analysis are, as already discussed above, the policy areas and the outcome at the ballot box, as well as the behaviour of the approval of a proposal.

Methods

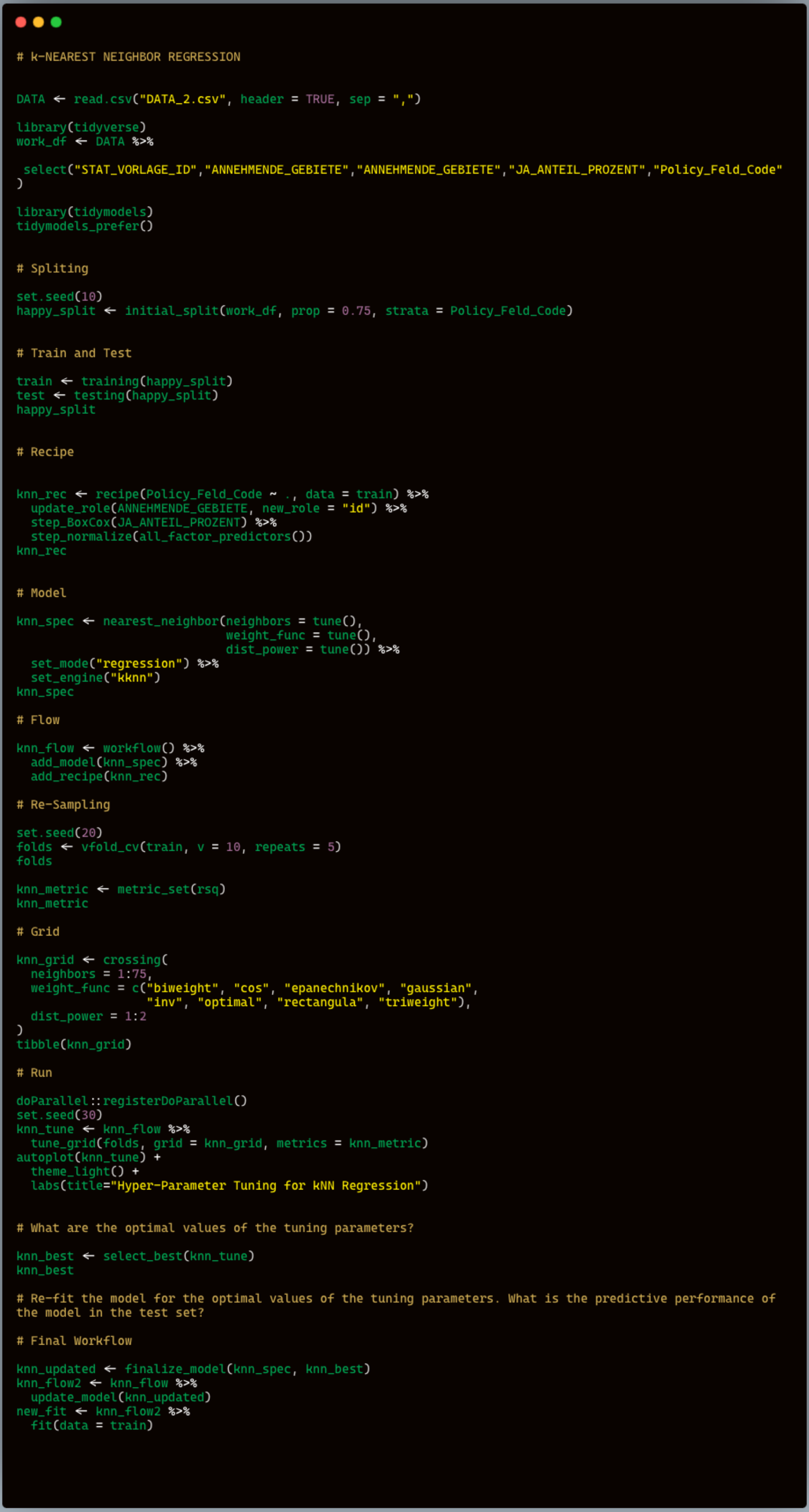
Algorithm

KNN

I decided to use a k-nearest neighbours (KNN) algorithm because I wanted to show the relationship between agreement with a model and the policy area.

Because of the categories, I found it useful to compare the results with their neighbours.

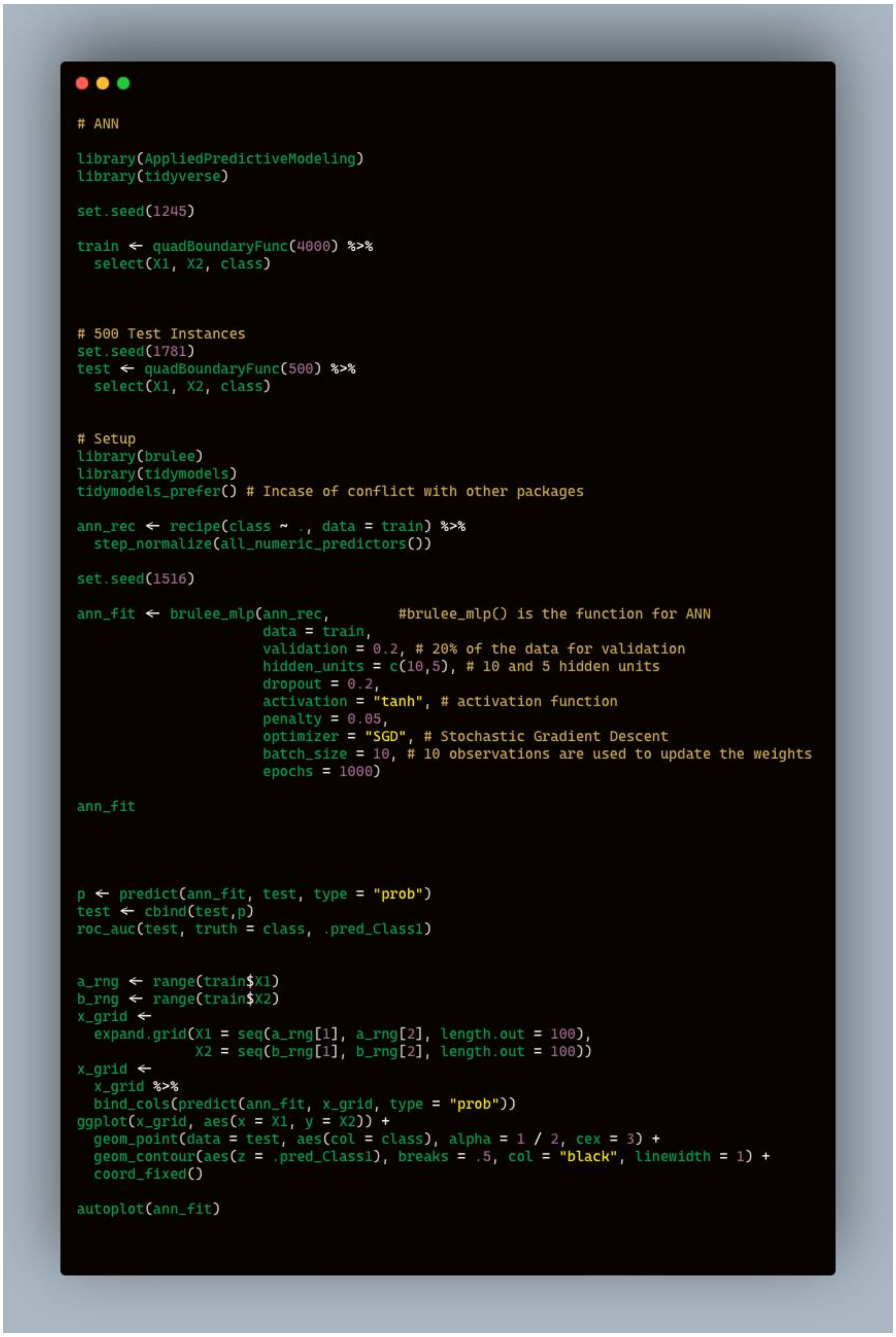
Another limitation is that no assumptions are made about the underlying data distribution, which is an advantage if you have no prior knowledge of the data set.



```r
# k-NEAREST NEIGHBOR REGRESSION


DATA ← read.csv("DATA_2.csv", header = TRUE, sep = ",")

library(tidyverse)
work_df ← DATA %>%

 select("STAT_VORLAGE_ID","ANNEHMENDE_GEBIETE","ANNEHMENDE_GEBIETE","JA_ANTEIL_PROZENT","Policy_Feld_Code"
)

library(tidymodels)
tidymodels_prefer()


# Spliting

set.seed(10)
happy_split ← initial_split(work_df, prop = 0.75, strata = Policy_Feld_Code)


# Train and Test

train ← training(happy_split)
test ← testing(happy_split)
happy_split


# Recipe


knn_rec ← recipe(Policy_Feld_Code ~ ., data = train) %>%
  update_role(ANNEHMENDE_GEBIETE, new_role = "id") %>%
  step_BoxCox(JA_ANTEIL_PROZENT) %>%
  step_normalize(all_factor_predictors())
knn_rec


# Model

knn_spec ← nearest_neighbor(neighbors = tune(),
                             weight_func = tune(),
                             dist_power = tune()) %>%
  set_mode("regression") %>%
  set_engine("kknn")
knn_spec

# Flow

knn_flow ← workflow() %>%
  add_model(knn_spec) %>%
  add_recipe(knn_rec)

# Re-Sampling

set.seed(20)
folds ← vfold_cv(train, v = 10, repeats = 5)
folds

knn_metric ← metric_set(rsq)
knn_metric

# Grid

knn_grid ← crossing(
  neighbors = 1:75,
  weight_func = c("biweight", "cos", "epanechnikov", "gaussian",
                  "inv", "optimal", "rectangula", "triweight"),
  dist_power = 1:2
)
tibble(knn_grid)

# Run

doParallel::registerDoParallel()
set.seed(30)
knn_tune ← knn_flow %>%
  tune_grid(folds, grid = knn_grid, metrics = knn_metric)
autoplot(knn_tune) +
  theme_light() +
  labs(title="Hyper-Parameter Tuning for kNN Regression")


# What are the optimal values of the tuning parameters?

knn_best ← select_best(knn_tune)
knn_best

# Re-fit the model for the optimal values of the tuning parameters. What is the predictive performance of
the model in the test set?

# Final Workflow

knn_updated ← finalize_model(knn_spec, knn_best)
knn_flow2 ← knn_flow %>%
  update_model(knn_updated)
new_fit ← knn_flow2 %>%
  fit(data = train)
```

ANN: (Artificial Neural Network)

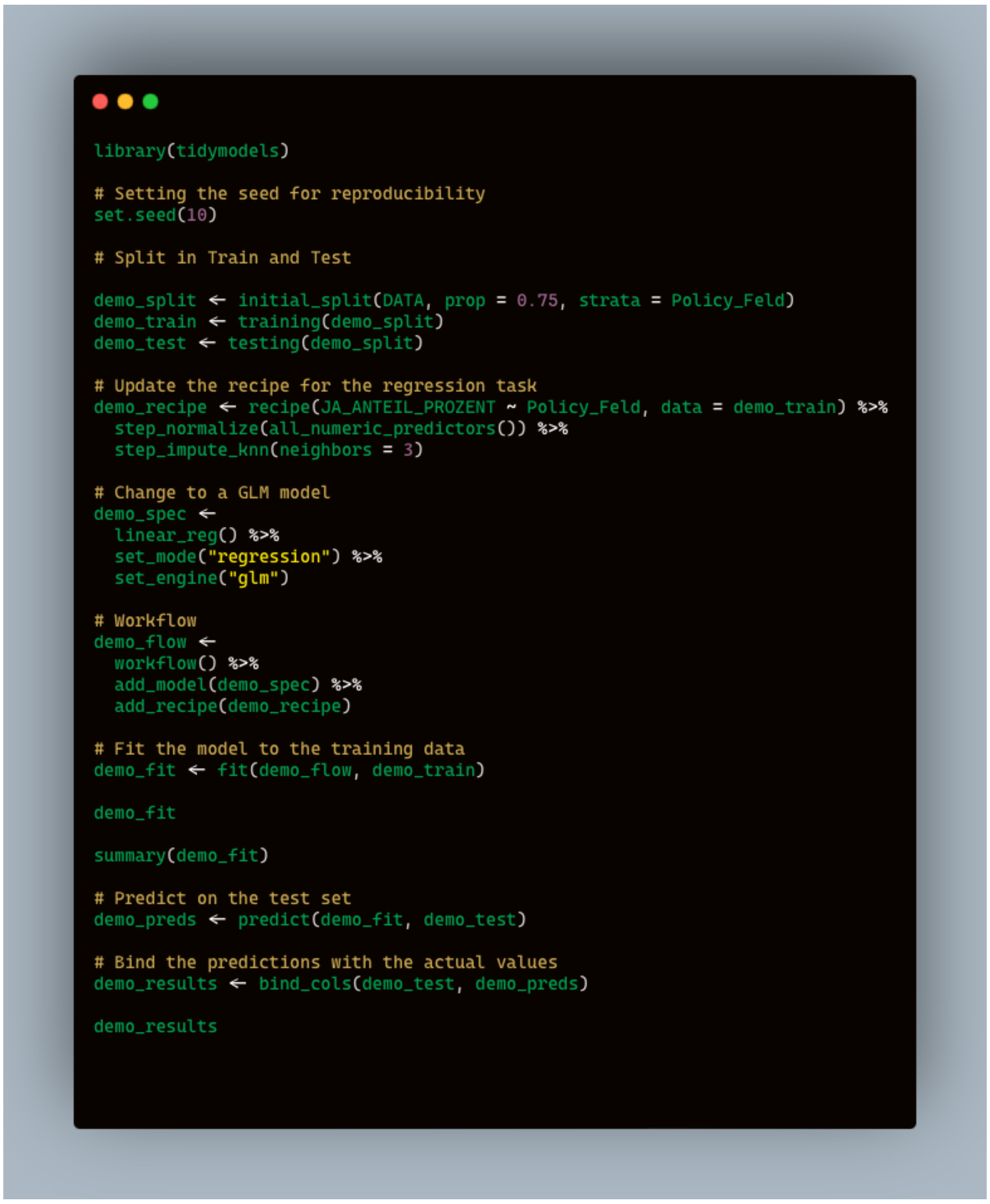
I used ANN because I suspected a link between the policy area and voting behaviour, especially because of the non-linear relationship.
However, the biggest problem I see is that many data points are needed to get valid and reliable results.



```r
# ANN

library(AppliedPredictiveModeling)
library(tidyverse)

set.seed(1245)

train <- quadBoundaryFunc(4000) %>%
  select(X1, X2, class)


# 500 Test Instances
set.seed(1781)
test <- quadBoundaryFunc(500) %>%
  select(X1, X2, class)


# Setup
library(brulee)
library(tidymodels)
tidymodels_prefer() # Incase of conflict with other packages

ann_rec <- recipe(class ~ ., data = train) %>%
  step_normalize(all_numeric_predictors())

set.seed(1516)

ann_fit <- brulee_mlp(ann_rec,        #brulee_mlp() is the function for ANN
                      data = train,
                      validation = 0.2, # 20% of the data for validation
                      hidden_units = c(10,5), # 10 and 5 hidden units
                      dropout = 0.2,
                      activation = "tanh", # activation function
                      penalty = 0.05,
                      optimizer = "SGD", # Stochastic Gradient Descent
                      batch_size = 10, # 10 observations are used to update the weights
                      epochs = 1000)

ann_fit



p <- predict(ann_fit, test, type = "prob")
test <- cbind(test,p)
roc_auc(test, truth = class, .pred_Class1)


a_rng <- range(train$X1)
b_rng <- range(train$X2)
x_grid <-
  expand.grid(X1 = seq(a_rng[1], a_rng[2], length.out = 100),
              X2 = seq(b_rng[1], b_rng[2], length.out = 100))
x_grid <-
  x_grid %>%
  bind_cols(predict(ann_fit, x_grid, type = "prob"))
ggplot(x_grid, aes(x = X1, y = X2)) +
  geom_point(data = test, aes(col = class), alpha = 1 / 2, cex = 3) +
  geom_contour(aes(z = .pred_Class1), breaks = .5, col = "black", linewidth = 1) +
  coord_fixed()

autoplot(ann_fit)
```

I also used a GLM model.

One of the main advantages of GLM models is that they are suitable for small data sets and are relatively easy to interpret.
The main disadvantages are that it is difficult to visualise the complexity and that they have limited predictive power.



```r
library(tidymodels)

# Setting the seed for reproducibility
set.seed(10)

# Split in Train and Test

demo_split <- initial_split(DATA, prop = 0.75, strata = Policy_Feld)
demo_train <- training(demo_split)
demo_test <- testing(demo_split)

# Update the recipe for the regression task
demo_recipe <- recipe(JA_ANTEIL_PROZENT ~ Policy_Feld, data = demo_train) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_impute_knn(neighbors = 3)

# Change to a GLM model
demo_spec <-
  linear_reg() %>%
  set_mode("regression") %>%
  set_engine("glm")

# Workflow
demo_flow <-
  workflow() %>%
  add_model(demo_spec) %>%
  add_recipe(demo_recipe)

# Fit the model to the training data
demo_fit <- fit(demo_flow, demo_train)

demo_fit

summary(demo_fit)

# Predict on the test set
demo_preds <- predict(demo_fit, demo_test)

# Bind the predictions with the actual values
demo_results <- bind_cols(demo_test, demo_preds)

demo_results
```

Tuning process

In the following I will explain the tuning parameters I used for KNN and ANN.

KNN:

I ran a k-fold cross validation. The training is divided into 10 parts (folds) and the algorithm is repeated 5 times.

I then set the metric to the squared correlation coefficient (R-squared).

Then grids of hyperparameters are created to be tested in the kNN algorithm. The hyperparameters are the number of neighbours, the weighting function (weight_func) and the distance power (dist_power).

In the actual tuning step, I performed the hyperparameter optimisation. The kNN algorithm is tested on each hyperparameter set of the grid and the metric (R-squared) is calculated.

ANN :

I did the following with the Artificial Neural Network:

I used brulee mlp to run a Multilayer Perceptron (MLP) and chose the following tuning parameters:

Hidden_units: The network has two hidden layers, the first with 5 units (or neurons) and the second with 2 units.

Dropout: A drop out rate of 10%.

For the activation of the neurons, I have chosen an activation function for the neurons. You've chosen "relu" (rectified linear unit), which is a commonly used activation function in neural networks. It introduces non-linearity into the model, allowing it to learn more complex patterns.

For the epoch, I set it to 1000, so that the learning algorithm runs through the data set 1000 times.

The rate schedule is set to cyclic, so I have cyclic learning.

With a step size of 4, step_size = 4 means that the learning rate will cycle every 4 epochs.

I did not tune the GLM model.

Data:

Descriptive Statistics

As a basis for my analysis, I used the data set: "Abstimmungsarchiv des Kantons Zürich ab 1831"(Statistisches Amt des Kantons Zürich, 2023).

There are two versions of this dataset: "Abstimmungsarchiv des Kantons Zürich, Ergebnisse Kanton Zürich" and "Abstimmungsarchiv des Kantons Zürich, Ergebnisse pro Gemeinde".

I first tried to work with the second dataset, which contains information on votes in every political municipality in the canton of Zurich. With over 5.1 million entries, the dataset was clearly too large to analyse. Even if I had focused on fewer columns and votes from 2000 onwards, I would still have had data from 160 municipalities for each of the 409 votes.
The computing power available to me on a laptop was not sufficient to produce results.
I therefore concentrated on the smaller dataset "Voting archive of the canton of Zurich, results of the canton of Zurich" and, in order to make the analysis more meaningful, only included votes from the year 2000 onwards.

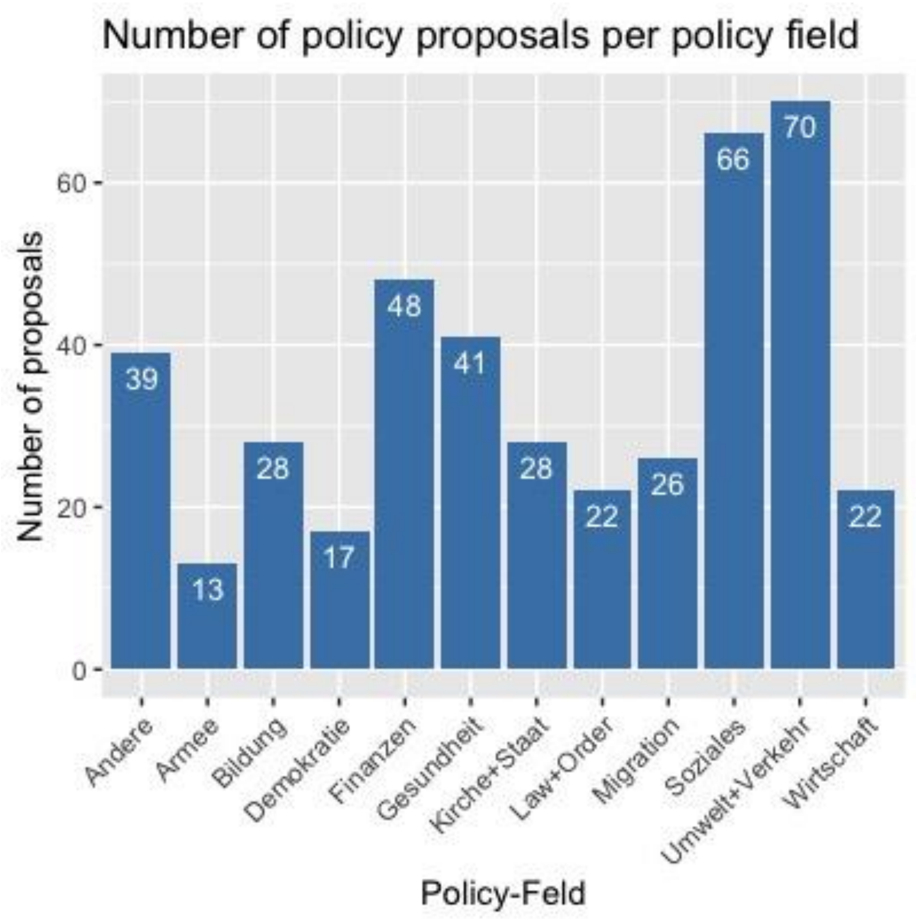
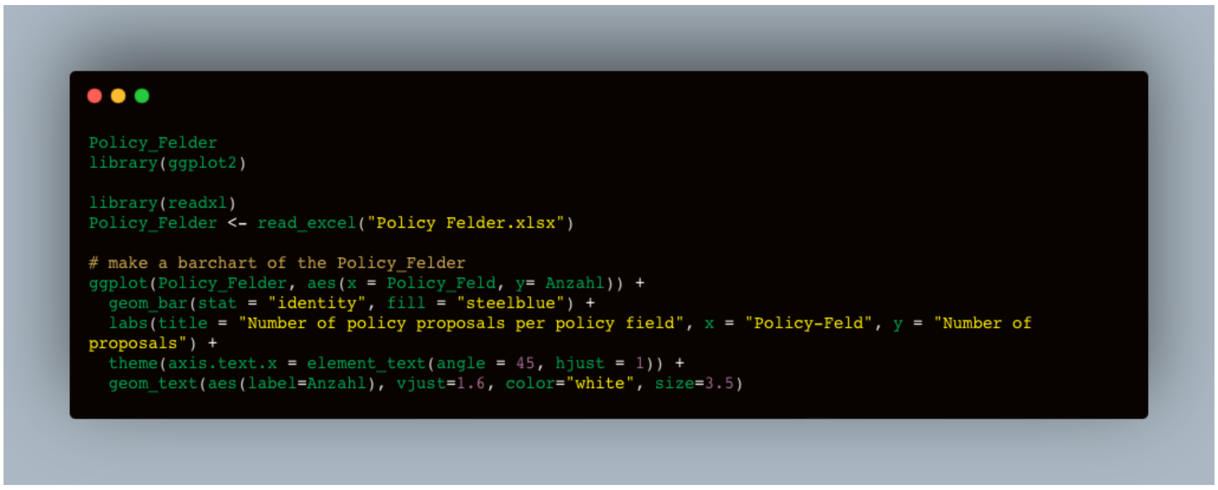


Fig. 1 Verteilung der Abstimmungen nach Politikfeld



```
Policy_Felder
library(ggplot2)

library(readxl)
Policy_Felder <- read_excel("Policy Felder.xlsx")

# make a barchart of the Policy_Felder
ggplot(Policy_Felder, aes(x = Policy_Feld, y= Anzahl)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(title = "Number of policy proposals per policy field", x = "Policy-Feld", y = "Number of
proposals") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  geom_text(aes(label=Anzahl), vjust=1.6, color="white", size=3.5)
```

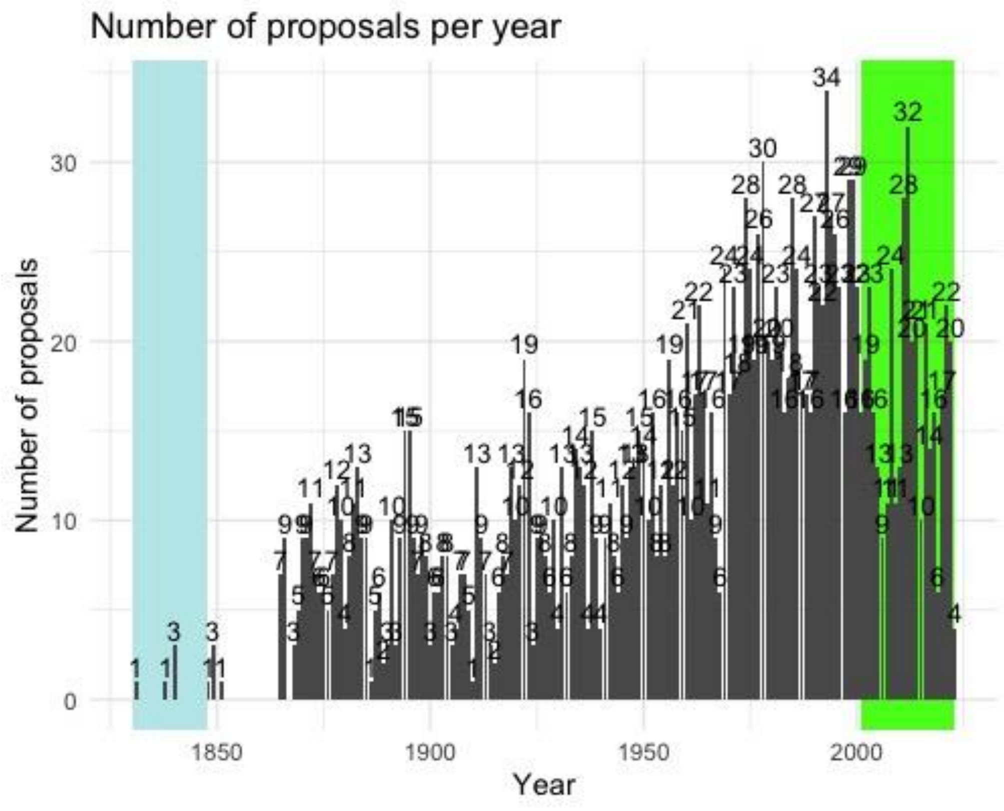
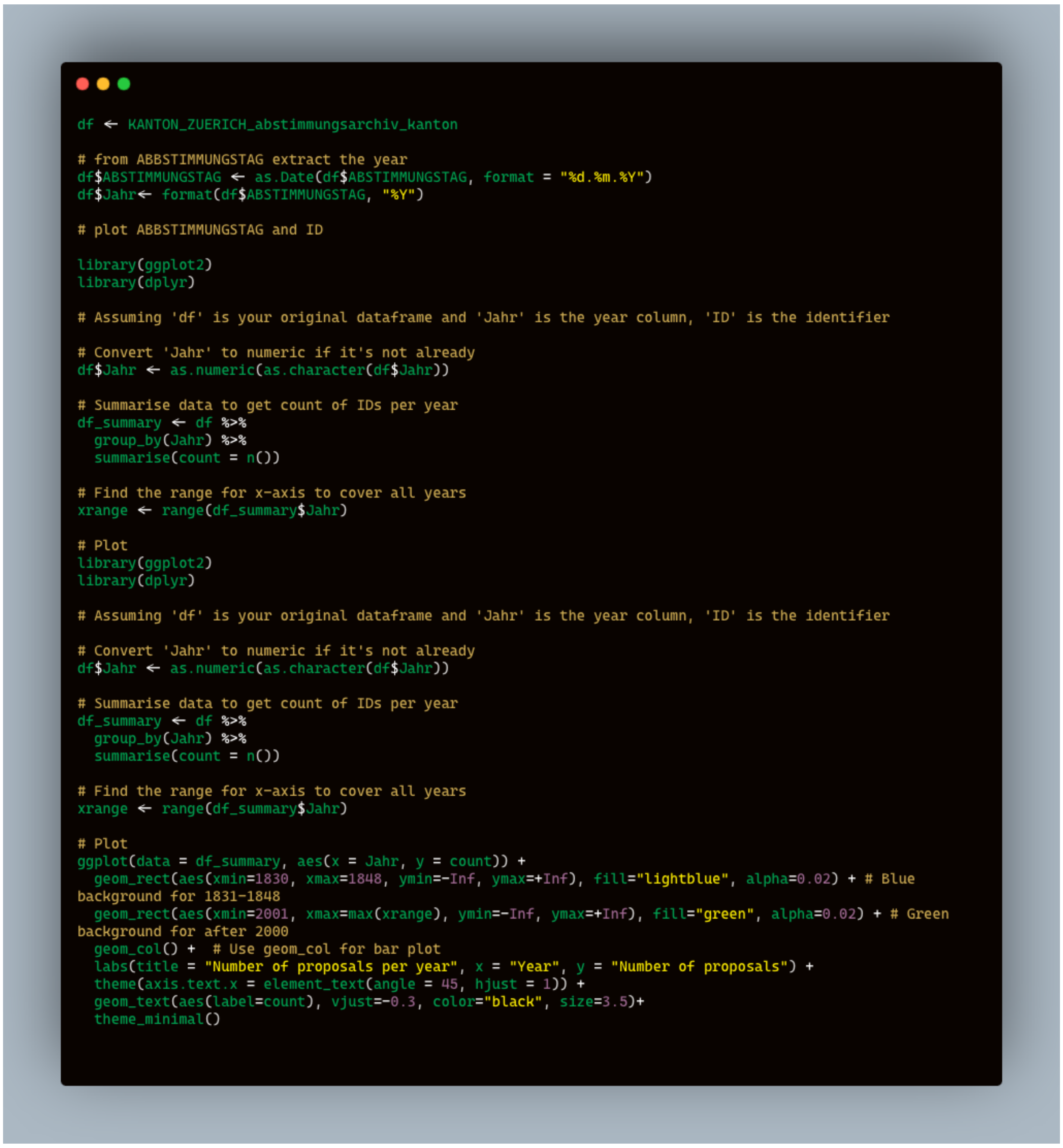


Fig. 2 Number of votes cast in one year.

Blue: before the creation of the federal state in 1848 Green: observation period



```r
df ← KANTON_ZUERICH_abstimmungsarchiv_kanton

# from ABBSTIMMUNGSTAG extract the year
df$ABSTIMMUNGSTAG ← as.Date(df$ABSTIMMUNGSTAG, format = "%d.%m.%Y")
df$Jahr← format(df$ABSTIMMUNGSTAG, "%Y")

# plot ABBSTIMMUNGSTAG and ID

library(ggplot2)
library(dplyr)

# Assuming 'df' is your original dataframe and 'Jahr' is the year column, 'ID' is the identifier

# Convert 'Jahr' to numeric if it's not already
df$Jahr ← as.numeric(as.character(df$Jahr))

# Summarise data to get count of IDs per year
df_summary ← df %>%
  group_by(Jahr) %>%
  summarise(count = n())

# Find the range for x-axis to cover all years
xrange ← range(df_summary$Jahr)

# Plot
library(ggplot2)
library(dplyr)

# Assuming 'df' is your original dataframe and 'Jahr' is the year column, 'ID' is the identifier

# Convert 'Jahr' to numeric if it's not already
df$Jahr ← as.numeric(as.character(df$Jahr))

# Summarise data to get count of IDs per year
df_summary ← df %>%
  group_by(Jahr) %>%
  summarise(count = n())

# Find the range for x-axis to cover all years
xrange ← range(df_summary$Jahr)

# Plot
ggplot(data = df_summary, aes(x = Jahr, y = count)) +
  geom_rect(aes(xmin=1830, xmax=1848, ymin=-Inf, ymax=+Inf), fill="lightblue", alpha=0.02) + # Blue
background for 1831-1848
  geom_rect(aes(xmin=2001, xmax=max(xrange), ymin=-Inf, ymax=+Inf), fill="green", alpha=0.02) + # Green
background for after 2000
  geom_col() +  # Use geom_col for bar plot
  labs(title = "Number of proposals per year", x = "Year", y = "Number of proposals") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  geom_text(aes(label=count), vjust=-0.3, color="black", size=3.5)+
  theme_minimal()
```

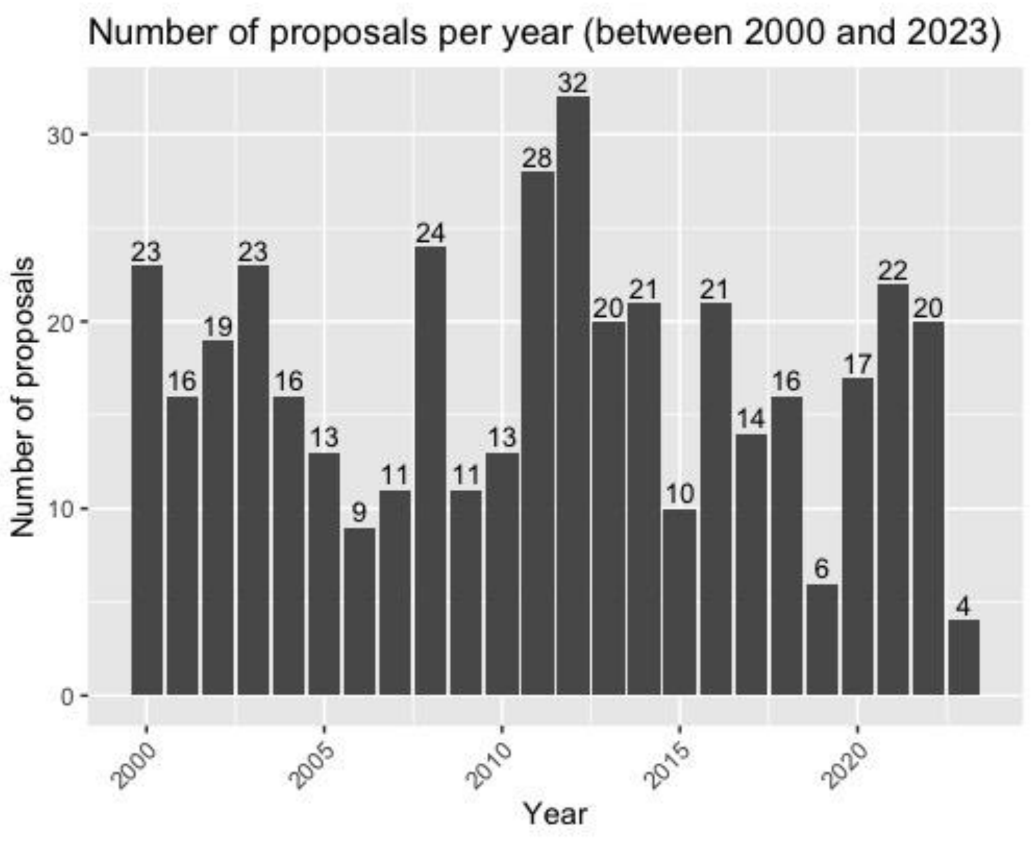


Fig. 3 Number of votes in the observation period

Pre-processing

Before I started working with the dataset, I first did a holistic coding of the voting templates as described above.
(The R code can be found in the appendix).

I decided to first create a subset of the data from the dataset "Voting Archive of the Canton of Zurich, Results Canton of Zurich".

In a first step, I filtered the data so that I only had entries after 01.01.2000.

In a second step, I removed variables that were not useful for the analysis, so that only the following variables remained for the evaluation:

"ABSTIMMUNGSTAG" Election Day

"VORLAGE_KURZBEZ" Short description of the bill

"STAT_VORLAGE_ID" ID oft he bill

"ABSTIMMUNGSART_BEZ" : Cantonal or federal

"ANNEHMENDE_GEBIETE" : How many municipalities have accepted the bill.

"ABLEHNENDE_GEBIETE": How many municipalities have rejected the billl?

"JA_ANTEIL_PROZENT" : Approval in per cent across the canton

"Policy_Feld" : My coding as described above

Policy_Feld_Code: numerical values

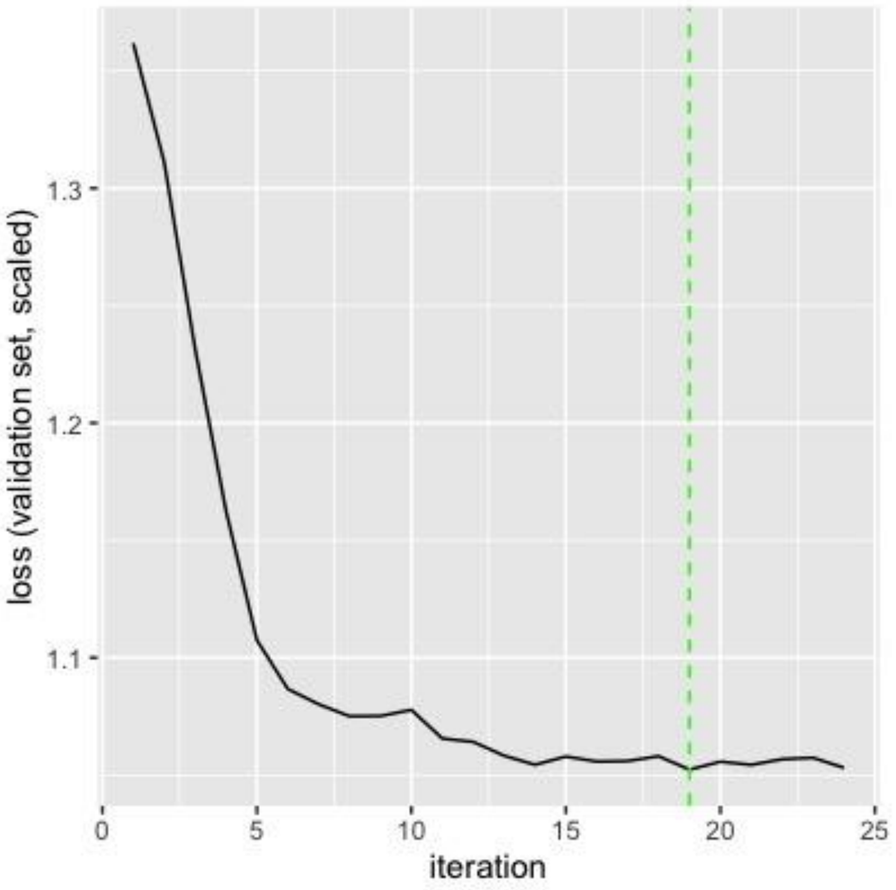
Performance

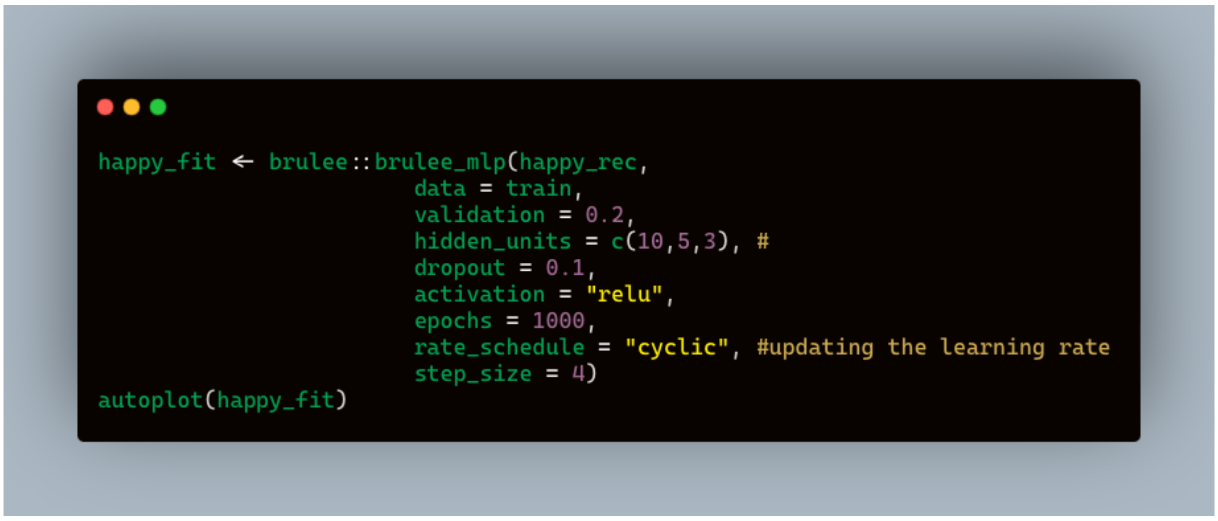
In the following I present the performance of the three approaches.

KNN:

The knn_test_metrics indicate that the k-nearest_neighbours model has a moderate level of prediction error and low explanatory power: an RMSE of 3.77 and MAE of 3.29 suggest average prediction errors of these magnitudes, while a low R² of 0.0206 implies that the model explains only about 2% of the variance in the data. These metrics suggest that the model may need improvement or a different approach to achieve better accuracy.

ANN:





```
happy_fit <- brulee::brulee_mlp(happy_rec,
                        data = train,
                        validation = 0.2,
                        hidden_units = c(10,5,3), #
                        dropout = 0.1,
                        activation = "relu",
                        epochs = 1000,
                        rate_schedule = "cyclic", #updating the learning rate
                        step_size = 4)
autoplot(happy_fit)
```

The graph shows the loss of the validation set as a function of the number of iterations (epochs). The loss value decreases and stabilises with the number of iterations, indicating that the model is starting to stabilise and possibly avoid overfitting. The green dashed line represents a point at which the learning rate is updated when a cyclic schedule is used.

results of parameter tuning

Parameter Choices:

Activation function: The use of ReLU (Rectified Linear Unit) activation functions is beneficial for its ability to

Hidden units (c(10,5,3)): The architecture of your model includes three hidden layers of 10, 5 and 3 units respectively. This configuration suggests a strategy of gradually reducing the complexity of the representations, which must strike a balance between being complex enough to capture data patterns and simple enough to avoid overfitting.

Model parameters (117): The relatively small number of parameters suggests a simpler model, which may be advantageous for generalisation but may lack capacity for more complex data patterns.

Samples (326) and features (3): The high ratio of samples to features is generally positive for training stable models.

Numerical score: This is used because the model is designed for a regression task.

Weight Decay (0.001): This regularisation technique helps prevent overfitting by penalising large weights, contributing to a more robust model.

Dropout rate (0.1): Implementing dropout helps prevent overfitting by randomly discarding a fraction of the inputs, which encourages the model to learn more robust features.

Batch Size (261): A large batch size, close to the total number of samples, means less stochastic noise during training, but may slow down convergence.

Schedule: Cyclic (step size = 4): A cyclic learning rate schedule may be effective.

Scaled validation loss after 19 epochs (1.05): The validation set loss is an important indicator of model performance. The appropriateness of a 1.05 loss value depends on the problem context and the scale of the outcome variable.

Performance on the Test Set

Qualitative assessment: the validation loss is an indication of the performance of the test set, a loss of 1.05 could indicate moderate performance.

Satisfactory performance: The determination of whether this performance is satisfactory depends on the specific application, baseline comparisons and project requirements. If this model shows improvement over previous iterations, or meets the project's benchmarks, then it could be considered satisfactory.

GLM

The results of the parameter tuning show that the choices made for the tuning of the parameters were aimed at optimising the predictive accuracy of the model. The chosen metrics and their respective estimates indicate the performance of the model:

The RMSE (Root Mean Square Error) of 17.0 indicates the standard deviation of the prediction errors, showing on average how much the predictions deviate from the actual values. A lower RMSE is generally preferred, so a value of 17.0 indicates moderate prediction accuracy.

The R-squared (R^2) of 0.0175 measures the proportion of variance in the dependent variable that can be predicted by the independent variables. This low value suggests that the model explains only a small proportion of the variability in the response data.

The MAE (Mean Absolute Error) of 14.1 reflects the average size of the errors in a set of predictions without considering their direction. Similar to the RMSE, a lower MAE is better and 14.1 indicates a moderate level of prediction error.

Overall, these metrics suggest that while the model has some predictive capability, its accuracy and explanatory power are somewhat limited, and further tuning or a different modelling approach may be required to improve performance on the test set.

Interpretation

First, I would like to discuss the GLM model,
I am aware that the R^2 of 0.01 is extremely low and therefore difficult to interpret.

Looking at the direction of the coefficients, we can see that voting is more likely to be rejected in the following policy areas:

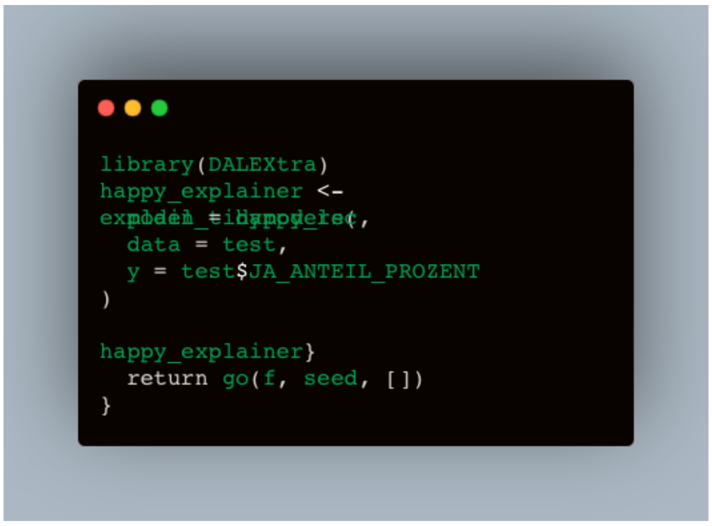
Education, Democracy, Social Affairs, Environment and Transport.

In the policy areas:

Finance, Health, Law and Order, Migration, Security and Military, and Economy, people are more likely to vote yes.

I also looked at the ANN results.

I also used the explain_tidymodels function from DALEXtra with DalExtra to see how the policy areas differ in terms of acceptance.



```
library(DALEXtra)
happy_explainer <-
explain_tidymodels(
  model = happy_rec,
  data = test,
  y = test$JA_ANTEIL_PROZENT
)

happy_explainer}
  return go(f, seed, [])
}
```

It can be seen that a policy in the area of law and order (Policy_Field_Code 11) was accepted in all municipalities analysed and received an overall approval rating of 76.93%.

A military policy (Policy_Field_Code 9) received only 55.17% support.

Conclusion

These approaches can be used to show that there are tendencies as to which policy areas receive more and which less support.

However, my sample was not large enough to obtain good results, especially with ANN.

In a follow-up study, it would certainly be interesting to carry out such an investigation on a larger sample, e.g. all votes in Switzerland, in order to obtain better results.
A better coding of the policy areas would also be necessary.

A note on transparency:

- I used ChatGPT and Copilot for the analysis.
- I created the Snipet code with Carbon (https://carbon.now.sh).

References

Afonso, A., & Papadopoulos, Y. (2015). How the Populist Radical Right Transformed Swiss Welfare Politics: From Compromises to Polarization. *Swiss Political Science Review*, *21*(4), 617–635. https://doi.org/10.1111/spsr.12182

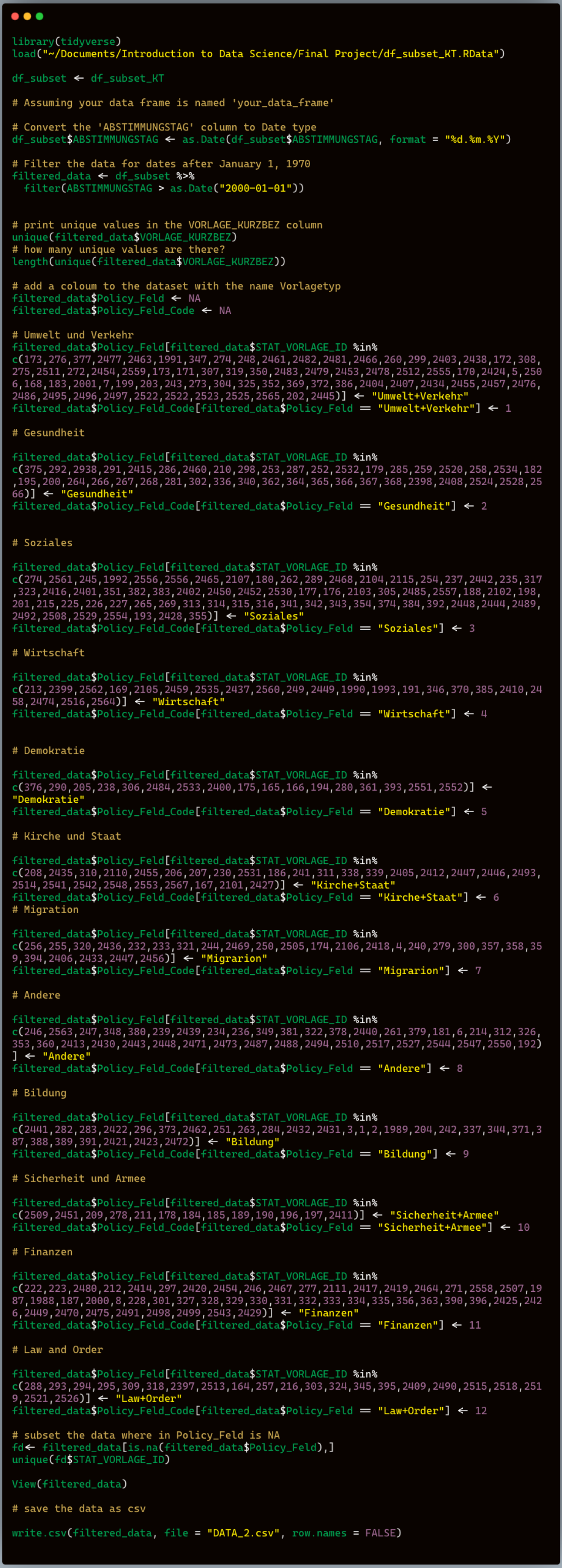
Kriesi, H. (2012). Personalization of national election campaigns. *Party Politics*, *18*(6), 825–844. https://doi.org/10.1177/1354068810389643

Milic, T. (2010). Steuern die Parteien das Volk? Der Einfluss der Parteien auf die inhaltliche Argumentation ihrer Anhängerschaften bei Schweizer Sachabstimmungen. *Zeitschrift Für Politikwissenschaft*, *20*(1), 3–45. https://doi.org/10.5771/1430-6387-2010-1-3

SRF. (2018, September 4). *Streit um die Limmattalbahn*. https://www.srf.ch/news/schweiz/abstimmung-kanton-zuerich-streit-um-die-limmattalbahn

Statistisches Amt des Kantons Zürich. (2023, June 21). *Abstimmungsarchiv des Kantons Zürich ab 1831*. https://www.zh.ch/de/politik-staat/statistik-daten/datenkatalog.html#/datasets/714@statistisches-amt-kanton-zuerich

Appendix



```r
library(tidyverse)
load("~/Documents/Introduction to Data Science/Final Project/df_subset_KT.RData")

df_subset ← df_subset_KT

# Assuming your data frame is named 'your_data_frame'

# Convert the 'ABSTIMMUNGSTAG' column to Date type
df_subset$ABSTIMMUNGSTAG ← as.Date(df_subset$ABSTIMMUNGSTAG, format = "%d.%m.%Y")

# Filter the data for dates after January 1, 1970
filtered_data ← df_subset %>%
  filter(ABSTIMMUNGSTAG > as.Date("2000-01-01"))


# print unique values in the VORLAGE_KURZBEZ column
unique(filtered_data$VORLAGE_KURZBEZ)
# how many unique values are there?
length(unique(filtered_data$VORLAGE_KURZBEZ))

# add a coloum to the dataset with the name Vorlagetyp
filtered_data$Policy_Feld ← NA
filtered_data$Policy_Feld_Code ← NA

# Umwelt und Verkehr
filtered_data$Policy_Feld[filtered_data$STAT_VORLAGE_ID %in%
c(173,276,377,2477,2463,1991,347,274,248,2461,2482,2481,2466,260,299,2403,2438,172,308,
275,2511,272,2454,2559,173,171,307,319,350,2483,2479,2453,2478,2512,2555,170,2424,5,250
6,168,183,2001,7,199,203,243,273,304,325,352,369,372,386,2404,2407,2434,2455,2457,2476,
2486,2495,2496,2497,2522,2522,2523,2525,2565,202,2445)] ← "Umwelt+Verkehr"
filtered_data$Policy_Feld_Code[filtered_data$Policy_Feld == "Umwelt+Verkehr"] ← 1

# Gesundheit

filtered_data$Policy_Feld[filtered_data$STAT_VORLAGE_ID %in%
c(375,292,2938,291,2415,286,2460,210,298,253,287,252,2532,179,285,259,2520,258,2534,182
,195,200,264,266,267,268,281,302,336,340,362,364,365,366,367,368,2398,2408,2524,2528,25
66)] ← "Gesundheit"
filtered_data$Policy_Feld_Code[filtered_data$Policy_Feld == "Gesundheit"] ← 2


# Soziales

filtered_data$Policy_Feld[filtered_data$STAT_VORLAGE_ID %in%
c(274,2561,245,1992,2556,2556,2465,2107,180,262,289,2468,2104,2115,254,237,2442,235,317
,323,2416,2401,351,382,383,2402,2450,2452,2530,177,176,2103,305,2485,2557,188,2102,198,
201,215,225,226,227,265,269,313,314,315,316,341,342,343,354,374,384,392,2448,2444,2489,
2492,2508,2529,2554,193,2428,355)] ← "Soziales"
filtered_data$Policy_Feld_Code[filtered_data$Policy_Feld == "Soziales"] ← 3

# Wirtschaft

filtered_data$Policy_Feld[filtered_data$STAT_VORLAGE_ID %in%
c(213,2399,2562,169,2105,2459,2535,2437,2560,249,2449,1990,1993,191,346,370,385,2410,24
58,2474,2516,2564)] ← "Wirtschaft"
filtered_data$Policy_Feld_Code[filtered_data$Policy_Feld == "Wirtschaft"] ← 4


# Demokratie

filtered_data$Policy_Feld[filtered_data$STAT_VORLAGE_ID %in%
c(376,290,205,238,306,2484,2533,2400,175,165,166,194,280,361,393,2551,2552)] ←
"Demokratie"
filtered_data$Policy_Feld_Code[filtered_data$Policy_Feld == "Demokratie"] ← 5

# Kirche und Staat

filtered_data$Policy_Feld[filtered_data$STAT_VORLAGE_ID %in%
c(208,2435,310,2110,2455,206,207,230,2531,186,241,311,338,339,2405,2412,2447,2446,2493,
2514,2541,2542,2548,2553,2567,167,2101,2427)] ← "Kirche+Staat"
filtered_data$Policy_Feld_Code[filtered_data$Policy_Feld == "Kirche+Staat"] ← 6
# Migration

filtered_data$Policy_Feld[filtered_data$STAT_VORLAGE_ID %in%
c(256,255,320,2436,232,233,321,244,2469,250,2505,174,2106,2418,4,240,279,300,357,358,35
9,394,2406,2433,2447,2456)] ← "Migrarion"
filtered_data$Policy_Feld_Code[filtered_data$Policy_Feld == "Migrarion"] ← 7

# Andere

filtered_data$Policy_Feld[filtered_data$STAT_VORLAGE_ID %in%
c(246,2563,247,348,380,239,2439,234,236,349,381,322,378,2440,261,379,181,6,214,312,326,
353,360,2413,2430,2443,2448,2471,2473,2487,2488,2494,2510,2517,2527,2544,2547,2550,192)
] ← "Andere"
filtered_data$Policy_Feld_Code[filtered_data$Policy_Feld == "Andere"] ← 8

# Bildung

filtered_data$Policy_Feld[filtered_data$STAT_VORLAGE_ID %in%
c(2441,282,283,2422,296,373,2462,251,263,284,2432,2431,3,1,2,1989,204,242,337,344,371,3
87,388,389,391,2421,2423,2472)] ← "Bildung"
filtered_data$Policy_Feld_Code[filtered_data$Policy_Feld == "Bildung"] ← 9

# Sicherheit und Armee

filtered_data$Policy_Feld[filtered_data$STAT_VORLAGE_ID %in%
c(2509,2451,209,278,211,178,184,185,189,190,196,197,2411)] ← "Sicherheit+Armee"
filtered_data$Policy_Feld_Code[filtered_data$Policy_Feld == "Sicherheit+Armee"] ← 10

# Finanzen

filtered_data$Policy_Feld[filtered_data$STAT_VORLAGE_ID %in%
c(222,223,2480,212,2414,297,2420,2454,246,2467,277,2111,2417,2419,2464,271,2558,2507,19
87,1988,187,2000,8,228,301,327,328,329,330,331,332,333,334,335,356,363,390,396,2425,242
6,2449,2470,2475,2491,2498,2499,2543,2429)] ← "Finanzen"
filtered_data$Policy_Feld_Code[filtered_data$Policy_Feld == "Finanzen"] ← 11

# Law and Order

filtered_data$Policy_Feld[filtered_data$STAT_VORLAGE_ID %in%
c(288,293,294,295,309,318,2397,2513,164,257,216,303,324,345,395,2409,2490,2515,2518,251
9,2521,2526)] ← "Law+Order"
filtered_data$Policy_Feld_Code[filtered_data$Policy_Feld == "Law+Order"] ← 12

# subset the data where in Policy_Feld is NA
fd← filtered_data[is.na(filtered_data$Policy_Feld),]
unique(fd$STAT_VORLAGE_ID)

View(filtered_data)

# save the data as csv

write.csv(filtered_data, file = "DATA_2.csv", row.names = FALSE)
```