Combine and Conquer: Mining Social Systems for Prediction

THÈSE N^{O} 6831 (2015)

PRÉSENTÉE LE 4 DÉCEMBRE 2015

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

LABORATOIRE POUR LES COMMUNICATIONS INFORMATIQUES ET LEURS APPLICATIONS 4

PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Vincent ETTER

acceptée sur proposition du jury:

Prof. S. Süsstrunk, présidente du jury

Prof. M. Grossglauser, Prof. P. Thiran, directeurs de thèse

Prof. S. Ioannidis, rapporteur

Dr A. Sala, rapporteuse

Prof. P. Frossard, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse

2015

A toi, qui arrives pour changer nos vies...

Acknowledgements

First and foremost, thank you, Patrick and Matthias, for being two amazing advisors. During the great journey that was my PhD research, you gave me the opportunity to investigate many directions, and you always pushed me to explore further, even if it led me outside of your comfort zones: I will be forever thankful for the trust that you put in me. I also really enjoyed our weekly discussions that, most of the time, left me with more questions than answers but were always entertaining and stimulating. Your complementarity, both in terms of scientific skills and as strong personalities, makes you a powerful *combination*. I learned a lot in your company over these five years.

I had the pleasure of having Sabine Süsstrunk, Pascal Frossard, Stratis Ioannidis, and Alessandra Sala as members of my committee. Thank you for your time, your feedback, and your excellent advice.

This thesis would not have existed without the excellent work of our support team working behind the scenes. Thank you, Angela, Danielle, Holly, and Patricia, for teaching me how to take care of the coffee machine, organizing all the apéros and the wonderful outings, and making all of these administrative headaches look like tiny details. Holly, your are responsible for all the errors that are *not* in this thesis, and I cannot thank you enough for this. Thank you, Marc-André, Stéphane, and Yves, for making all of IC jealous of our IT infrastructure, and for putting up with all my weird requirements and crazy ideas.

Thank you, Martin Vetterli, for welcoming me in your lab more than once, and for being such a great source of inspiration and advice. I deeply enjoyed each of our discussions.

Thank you, Julien, Mohamed, Robin, Ivan, Juri, Farid, Lucas, Sébastien, Mathias, Igor, Kévin, Christina, Brunella, Lyudmila, Ehsan, Runwei, Emti, Young-Jun, Victor, William, Farnood, Marc, Antonin, Andrea, Vojin, and Adrian. We shared long coffee-breaks, numerous lunches in SV, beers at Satellite and at the Great, poker nights, ski outings, and even papers and trips. Thank you for all the good times, the laughs, the dry runs, the discussions on typography, and the feedback on the plots. You guys made the hard times good, and the good times better. I cannot thank you enough for making me looking forward to coming to the lab every day of the past five years.

Thank you, Jean, Tiffany, Julien, Mathilde, Christian, Gatien, Steve, Fabian, Caylene, the Animalis team, and the ISBC crew. You were there for movies, dinners (*chez Xu* or at your place), game nights, and wild parties. Thank you for reminding me that there is

a life outside of the PhD world, and that sharing it with you is what makes it awesome.

I would never have begun my PhD research without having experienced what working in an industrial research lab is like. Thank you, Ronan, for the two great internships at NEC, and for opening the path of my life that led me here today. Thank you, Zilla, for welcoming me at MSR, and for making it possible for me to continue on this path after my PhD.

Thank you, *Papa et Maman*. You were always there for me and you have always let me decide what I want to do. Thank you for giving me the opportunity to study for so long and do what I want, for letting me go to the other side of the world even though it scared you, and for being there for me each time I took a wrong turn. I am really grateful for your trust. Thank you, Didier, for being such a good big brother. You have always had my back, and I will always have yours.

Thank you, the Morel family, for welcoming me as your latest member. I really enjoyed getting to know each of you over the dinners and trips together, and discovering what it is like to be part of such a big family. Finally, thank you, Sophie. You were here for every step of this amazing journey, and I would never have made it so far without you. Thank you for your love and support, for making me want to be a better person, for always reminding me that there is more than one side to the world, and for following me in our dreams. You have given me the opportunity to embark with you on my greatest journey yet, and for that, I'll be forever thankful.

Lausanne, le 4 Décembre 2015 V. E.

Résumé

Depuis des décennies, les informaticiens développent des outils pour les aider à traiter de grands jeux de données, dont la création a été rendue triviale par les ordinateurs. A l'aube du 21ème siècle, pourtant, ces données massives ne sont plus l'exclusivité de l'informatique, mais sont au contraire devenues omniprésentes : chaque aspect de la vie humaine génère désormais quotidiennement des quantités exponentielles de données. Les outils créés par les informaticiens sont donc devenus utiles, voire essentiels, à beaucoup d'autres domaines, comme les sciences humaines, l'économie, ou la biologie. De par leur expertise dans l'utilisation de ces outils, les chercheurs en informatique sont donc souvent les premiers à les appliquer sur de telles données. Ils amènent alors un regard nouveau sur des problèmes existants dans d'autres domaines, et permettent ainsi parfois de prendre du recul et d'aborder ces problèmes sous un angle différent.

Dans cette thèse, nous explorons l'application de techniques d'analyse de données et d'apprentissage automatique à plusieurs problèmes pratiques. Ces problèmes tirent leur origine de plusieurs domaines, tels que les sciences sociales, l'économie, et les sciences politiques. Nous démontrons que les techniques issues de l'informatique permettent de contribuer de manière significative à la résolution de ces problèmes. De plus, nous montrons que la combinaison de plusieurs modèles, ou de plusieurs jeux de données, joue un rôle déterminant dans la qualité des solutions que nous trouvons.

La première application que nous considérons est la mobilité humaine. Nous décrivons notre participation gagnante au Nokia Mobile Data Challenge, où notre tâche était de prédire le prochain endroit qu'un utilisateur allait visiter, étant donné son historique et son contexte actuel. Un examen méticuleux des données nous permet de mettre en évidence certaines de leurs caractéristiques, comme leur non-stationnarité ou leur rareté, qui rendent la tâche difficile. Nous présentons trois familles de modèles dont les performances varient de manière significative d'un utilisateur à l'autre, même si leurs performances globales sont similaires. Afin de tirer avantage de cette diversité, nous introduisons plusieurs stratégies de combinaison des modèles, dont les performances dépassent celles des prédicteurs individuels.

La seconde application à laquelle nous nous intéressons est la prédiction du succès de campagnes de financement participatif. Nous avons récolté des données sur Kickstarter (une des plateformes de financement participatif les plus populaires), afin de prédire si une campagne atteindra son but de financement ou non. Nous montrons qu'il est possible d'obtenir de bons résultats en considérant uniquement les informations concernant l'argent

promis aux campagnes, mais que la qualité des prédictions est améliorée si l'on combine ces informations avec des données tirées du réseau social de Kickstarter, ainsi que de Twitter. En particulier, les prédictions faites quelques heures seulement après le lancement d'une campagne sont améliorées de 4 %, atteignant ainsi une précision de 76 %.

Nous passons ensuite dans le domaine de la politique, et commençons par étudier les idéologies des politiciens. En utilisant leurs opinions, récoltées sur une plateforme d'aide au vote, nous montrons que les thèmes qui divisent le plus les politiciens sont ceux qui sont habituellement associés à une orientation gauche/droite, ou libérale/conservatrice, validant ainsi la représentation simplifiée du système politique qui est couramment utilisée. Afin d'attirer l'attention sur de potentielles mauvaises utilisations des plateformes d'aide au vote, nous créons un profil pour un candidat fictif qui récolte deux fois plus de recommandations de vote que n'importe quel candidat existant. Pour contrer ce genre d'abus, nous démontrons qu'il est possible de surveiller les politiciens une fois qu'ils ont été élus, et de potentiellement détecter les changements d'opinion, en combinant les données extraites de la plateforme d'aide au vote avec les résultats de leurs votes au Parlement.

Finalement, nous étudions les résultats de votations. Nous montrons d'abord qu'il est possible de mettre en avant des habitudes de vote typiques d'un pays, et leur évolution au cours du temps, en utilisant uniquement les résultats de votations à un niveau géographique détaillé. Ces résultats nous permettent aussi d'identifier des régions représentatives du pays, dont la connaissance du résultat est cruciale pour une bonne prédiction du résultat national d'une votation. Nous nous intéressons ensuite à la prédiction du résultat exact d'une votation dans toutes les régions (au lieu du résultat national binaire) et nous montrons que l'obtention de bonnes performances nécessite la combinaison de données concernant les régions et les votes eux-mêmes. Nous comparons l'utilisation de méthodes bayésiennes et non-bayésiennes qui combinent factorisation de matrices et régression. Nous montrons que les méthodes bayésiennes permettent de mieux estimer les hyperparamètres que des méthodes non-bayésiennes comme la validation croisée, et que, à nouveau, la combinaison de modèles et de données appropriés permet d'améliorer la qualité des prédictions. De plus, les modèles ainsi obtenus sont applicables à d'autres problèmes, produisent des prédictions robustes, et peuvent facilement être interprétés.

Mots-clés : analyse de données, apprentissage automatique, combinaison de modèles et de jeux de données, prédiction de la mobilité humaine, prédiction du succès du financement participatif, analyse de données politiques, prédiction du résultat de votations, réduction de dimensionnalité, modèles bayésiens, processus gaussien

Abstract

Computer scientists have been developing tools to deal with large quantities of data for decades, as the very nature of computers makes creating large datasets trivial. In the beginning of the 21st century, however, large datasets have become ubiquitous: data is being generated by all aspects of human life, at an exponentially increasing rate. These tools are thus becoming useful, even essential, to many other fields, such as human sciences, economics, and biology. As creators of these tools, computer scientists are often among the first to apply them to such datasets, shedding a new light on existing problems in other fields, allowing to overcome tunnel vision and make progress.

In this thesis, we explore the application of data mining and machine learning techniques to several practical problems. These problems have roots in various fields such as social science, economics, and political science. We show that computer science techniques enable us to bring significant contributions to solving them. Moreover, we show that *combining* several models or datasets related to the problem we are trying to solve is key to the quality of the solution we find.

The first application we consider is human mobility prediction. We describe our winning contribution to the Nokia Mobile Data Challenge, in which we predict the next location a user will visit based on his history and the current context. By carefully examining the data, we are able to highlight some characteristics that contribute to the difficulty of the task, such as sparsity and non-stationarity. We present three different families of models and observe that, even though their average accuracies are similar, their performances vary significantly across users. To take advantage of this diversity, we introduce several strategies to combine models, and show that the combinations outperform any individual predictor.

The second application we examine is predicting the success of crowdfunding campaigns. We collected data on Kickstarter (one of the most popular crowdfunding platforms), in order to predict whether a campaign will reach its funding goal or not. We show that we obtain good performances by simply using information about money, but that combining this information with social features extracted from Kickstarter's social graph and Twitter improves early predictions. In particular, predictions made a few hours after the beginning of a campaign are improved by 4 %, to reach an accuracy of 76 %.

Then, we move to the realms of politics, and first investigate the ideologies of politicians. Using their opinion on several aspects of politics, gathered on a voting advice application (VAA), we show that the themes that divide politicians the most are the ones that we

usually associate with left-wing/right-wing and liberal/conservative, thus validating the simplified two-dimensional view of the political system that many people use. We bring attention to the potentially malicious uses of VAAs by creating a fake candidate profile that is able to gather twice as many voting recommendations as any other. To counter this, we demonstrate that we are able to monitor politicians after they were elected, and potentially detect changes of opinion, by combining the data extracted from the VAA with the votes that they cast at the Parliament.

Finally, we study the outcome of issue votes. We first show that simply considering vote results at a fine geographical level is sufficient to highlight characteristic geographical voting patterns across a country, and their evolution over time. It also enables us to find representative regions that have a high predictive power of the national outcome of a vote. We then demonstrate that predicting the actual result of a vote in all regions (in opposition to the binary national outcome) is a much harder task that requires combining data about regions and votes themselves to obtain good performances. We compare the use of Bayesian and non-Bayesian models that combine matrix-factorization and regression. We show that Bayesian methods give better estimates of the hyperparameters than non-Bayesian methods such as cross-validation, and that, here too, combining appropriate models and datasets improves the quality of the predictions. Moreover, the resulting models generalize well to many different tasks, produce robust predictions, and are easily interpretable.

Keywords: data mining, machine learning, combining models and datasets, human mobility prediction, crowdfunding success prediction, political data analysis, vote results prediction, dimensionality reduction, Bayesian models, Gaussian processes.

Contents

1 Introduction

> They say a little knowledge is a dangerous thing, but it is not one half so bad as a lot of ignorance.
>
> *Terry Pratchett*

Data has always been at the heart of research: For centuries, scientists have conducted experiments to collect observations, in order to verify or refute their hypotheses. The 21st century is witnessing a change to this millennial way of doing science: in all aspects of human life, data is generated and collected at an exponentially increasing rate. The capabilities of computers and the advances in computer science have rendered possible the simulation of very complex systems and the processing of high volumes of data, thus enabling scientists to move from tedious and time-consuming physical experiments to rapid, highly-parallelizable software experiments. For example, biologists who used to run experiments *in vivo* (with a living creature) and *in vitro* (inside a test tube) "now commonly speak of doing them *in silico* — as simulations run on the silicon chips of a computer" [66].

Some of the contributions from computer science to the society are *direct* products of research in this field. For example, the World Wide Web, developed at CERN by Tim Berners-Lee in the late 1980s, has been instrumental in the spread of knowledge and is at the core of the development of the Information Age. More fundamentally, the formal definitions of *computation* and *computability* have deep implications in fields such as mathematics [94], philosophy [20], and biology [88].

These two advancements, the availability of data and increased computational power, have also enabled computer science to bring significant *indirect* contributions. Indeed, the application of techniques and algorithms (from computer science) to data generated by experiments in other fields has enabled developments that were previously undreamed of. Physicists, for example, rely heavily on computers to process and filter measurements

gathered in the Large Hadron Collider (LHC). As each experiment in the LHC generates 600 million events per second, they use specialized hardware and software to filter these events, recording only 200 events per second [32]. Even with such a significant reduction in the amounts of data recorded, the LHC annually produces 30 petabytes of data [31] — about 300 000 times the size of the whole English language Wikipedia — which then need to be stored and processed efficiently.

Another great success of computer science in advancing other fields is the Human Genome Project. This project began in 1990, with the goals of determining the sequence of chemical elements that make up human DNA and of understanding the roles of all genes in the human genome. Initially estimated to take 15 years, the sequencing was completed in 2003, two years ahead of schedule due to the advances in data-processing techniques and computer hardware [86].

In this thesis, we will show that many fields, such as social science, economics, and political science, can directly benefit from the algorithms developed by computer scientists. In particular, techniques from the data-mining and machine-learning communities can be adapted to solve practical problems that involve gathering, processing, and combining data at a large scale. Such applications are at the core of *why* computers were created: to enable people to solve problems that are too complex to be handled manually.

In short, the work presented in this thesis has two purposes. First, we show that applying techniques from machine learning and data mining enables us to significantly improve the solutions to problems that stem from fields such as social science, economics, and political science. Second, we propose novel applications of data-mining techniques from which practitioners of these fields can gain new insights into their data. One key characteristic of our work is that we *combine* models or datasets that are relevant to the task at hand, reinforcing the old adage saying that "the whole is greater than the sum of its parts".

1.1 Outline and Contributions

In this dissertation, we present our work on four different problems:

- The first problem we tackle is related to the mobility of people. We introduce, in Chapter 2, three families of methods for predicting the next location a user will visit, based on data recorded from his cellphone. Highlighting the diversity in the results obtained by these three families for different users, we propose five different strategies to combine these models. We submitted the predictions obtained using these combinations to the Nokia Mobile Data Challenge and won first place in the Next Place Prediction Task.
- In Chapter 3, we introduce the problem of predicting the success of crowdfunding campaigns. We present a novel dataset that we collected over a period of eight months and made available to the scientific community. We developed several

models that each outperform previous work on crowdfunding success prediction, and we show that the combination of financial data with social features significantly improves early predictions.

- In Chapter 4, we detail our data-driven study of the political landscape of Switzerland. We analyze the political opinions of citizens, political candidates, and elected members of the parliament, and we show that the usual left/right and liberal/conservative simplification of the political space is indeed the most efficient way of summarizing the political opinions of Swiss people. We are the first to highlight potential misuses of the data from voting advice applications. For example, we propose a method for taking advantage of this data to optimize the answers of a candidate in order to obtain more recommendations. We also present a technique for detecting candidates who falsely advertise their opinion on the voting advice application and then vote differently once elected.
- Finally, we study the outcome of issue votes in Chapter 5. We introduce a new dataset of hundreds of votes, with detailed geographical results and side information about both votes and regions. We analyze voting patterns over a large time scale and show that such an analysis enables political scientists to highlight interesting behaviors. We also make a methodological contribution by detailing the construction of a model for predicting the outcome of a vote in each region. We show how using a Bayesian approach enables us to properly tune the model's hyperparameters and to significantly improve the prediction results.

1.2 Models

The four problems we described above can be solved using several methods. Indeed, computer scientists have been developing models and algorithms for decades in order to solve practical problems of different complexities. When presented with a new problem, one of the main difficulties we face is in selecting which method to use. Of course, there is no golden rule, and several techniques often give similar results for a given application.

We list below the models we use in this thesis. We do not claim to give a detailed and comprehensive description of these models, rather we provide the main idea behind each of them and we list their main advantages and limitations. Table 1.1 summarizes the list of models and in which chapters they are used. In Section 1.2.3, we give more details about the reasons behind our choice of models.

1.2.1 Evolution of a System over Time

The first two methods we introduce are suited for modeling the evolution of a system over time. They apply to a wide variety of problems, from modeling physical systems to describing the relationship of symbols in DNA.

	Chapter 2	Chapter 3	Chapter 4	Chapter 5
Markov chain	✓	✓	–	–
Dynamic Bayesian network	✓	–	–	–
k-nearest neighbors	–	✓	–	–
Logistic regression	–	–	✓	–
Artificial neural network	✓	–	–	–
Support vector machine	–	✓	–	–
Gradient-boosted decision tree	✓	–	–	✓

Table 1.1 – Summary of the different methods used in this thesis, and the chapter in which they are mentioned.

Markov Chain

A first-order Markov chain (MC) is a mathematical model that can be defined by a *state space* $\mathcal{X}$, that enumerates the different states the system can be in, and a *transition matrix* $\boldsymbol{P}$, where $P_{ij} \in [0, 1]$ represents the probability of the system going from state $i \in \mathcal{X}$ to state $j \in \mathcal{X}$.

The key property of MCs is that their *future* evolution is independent from their *past*, given their *present* state. In other words, if we represent the state of the system at time n as a random variable $X_n \in \mathcal{X}$, the sequence of random variables $X_1, X_2, \ldots, X_n, X_{n+1}$ is a MC if and only if

$$\mathbb{P}(X_{n+1} = i \mid X_1 = k, \ldots, X_n = j) = \mathbb{P}(X_{n+1} = i \mid X_n = j) = P_{ij}.$$

MCs provide an intuitive framework for modeling data that evolves over time. Their training is efficient, and they require only the transition matrix $\boldsymbol{P}$ to be stored. Their main limitations are that their state space is discrete and that their expressiveness is directly tied to the size of their state space. They can be extended to higher orders, where the probability of the next state depends on the k past states (instead of only the current one). However, the storage cost and training data requirements of these higher-order MCs grow exponentially with their order k.

Dynamic Bayesian Network

A dynamic Bayesian network (DBN) [84] is a probabilistic model that can be used to express the conditional dependencies of random variables over adjacent time steps. It is a very powerful tool that enables the modeling of complex systems in which several random variables are linked and evolve over discrete time steps. Markov chains, presented above, can be seen as a special case of DBNs.

The strength of DBNs lies in their flexibility: they can be tailored to describe many practical problems in a very intuitive way. Once the various variables have been defined, a DBN enables us, for example, to perform inference on the future state of the system, based on some observed current and past data. The main drawback is that the feasibility of this inference depends on the structure of the DBN. In many practical applications, it is not possible to perform an exact inference and complex approximate-inference algorithms are required.

1.2.2 Classification

In some contexts, modeling the evolution of a system over time can be seen as a more general classification task: Given some features, such as its current state and possibly some side information, what will be the next state of the system? In this case, the classes are simply the set of possible states. Deciding between two (or more) possible outcomes is also at the heart of many other practical problems: Is this tumor benign or malignant? Is this e-mail ham or spam? Is this a photo of a cat, a dog, or a human? The classification algorithms we describe below are designed to solve such tasks, but they differ in their requirements and complexity.

Logistic Regression

Logistic regression (LR) [65] is one of the simplest classification techniques. It extends the idea of linear regression to binary classification, by passing the output of the regression through a logistic function $\sigma(x)$ (illustrated in Figure 1.1) that maps any value $x \in \mathbb{R}$ to the interval $[0, 1]$:

$$\sigma(x) = \frac{1}{1 + \exp(-x)}.$$

If we encode the binary classes as 0 and 1, the output of LR can be interpreted as the probability of the input being of class 1. In other words, given an N-dimensional sample $\boldsymbol{x} \in \mathbb{R}^N$ and its class $y \in \{0, 1\}$, we have that

$$\mathbb{P}(y = 1 \mid \boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{\beta}\boldsymbol{x})},$$

where $\boldsymbol{\beta} \in \mathbb{R}^N$ are the parameters of the model.

LR can be trained and evaluated efficiently, especially for high-dimensional datasets. Its main limitation is that it only performs well with linearly-separable datasets, thus rendering it unusable for many complex problems that require more flexible decision regions.

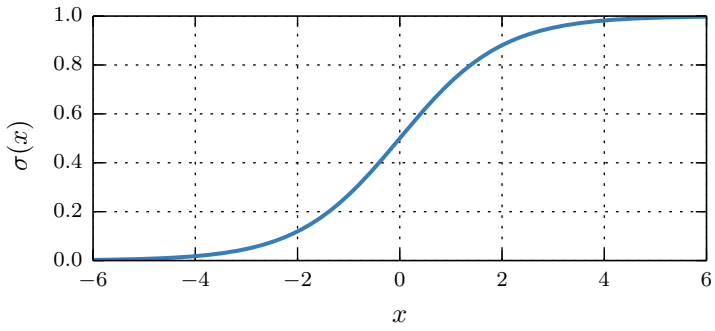


Figure 1.1 – Logistic function $\sigma(x)$.

k-Nearest Neighbors

k-nearest neighbors (k-NN) [38] is a simple algorithm that can be used for classification and regression. When used for classification, it takes as input a list of training samples, along with their assigned labels. To classify a new sample, k-NN simply computes the distance of this sample to all known training samples and outputs the class most common among the k closest ones. The metric used to compute the distance can be chosen freely, a typical choice being the l_2-norm.

The main advantage of k-NN is that it is simple and very intuitive. However, its main drawback is that it strongly suffers from the *curse of dimensionality* [28]: in high-dimensional spaces, the nearest and the farthest neighbors tend to be at similar distances, resulting in poor classification results. Moreover, it is computationally expensive: The classification of each new sample requires its distance to all training samples to be computed, and thus all training samples to be kept in memory. Even though some more advanced techniques (such as k-d trees [27]) can be used to find the k closest samples efficiently, the computational costs of k-NN prevents its use with datasets of large size.

Artificial Neural Network

An artificial neural network (ANN) [81] is a classification algorithm inspired by the structure of the brain. It passes its input through a sequence of *layers* that each apply a linear transformation and that are interconnected by *transfer functions*. A two-layer ANN with a suitable transfer function can approximate any continuous function [64]. The shape and transfer function of the output layer can be tailored for various applications, for example, to perform multi-label classification.

The main advantages of ANNs are that they can be trained relatively simply and be evaluated rapidly (as they mostly rely on matrix multiplications), and that their generalization power is directly controllable through their structure. They can also easily handle multi-label classification problems. Their main drawback is that there is no guarantee that their training will find a global optimum.

Support Vector Machine

A support vector machine (SVM) [37] is a popular binary classification algorithm. The intuition behind a SVM is simple: It maps training samples to a high-dimensional space and finds the hyperplane separating the two classes of samples in that space with the largest "margin". This hyperplane is built by selecting training samples that are close to the decision threshold: the *support vectors*.

The main advantage of SVMs is that their training is stable, because it always converges to a global optimum. Moreover, they can be very efficient to evaluate, as they only require the support vectors for the computation of the decision threshold. Their main drawbacks are that their training can be slow for large datasets and that their complexity can only be indirectly controlled using hyperparameters. Moreover, they are not easily applicable to multi-label problems.

Gradient-Boosted Decision Tree

A simple way of building a classifier is to enumerate the rules that qualify a sample to be a member of a particular class. These rules can be expressed as decision trees, where nodes represent decisions made on some features of the sample, and leaves represent the classes we associate with the sample. Gradient-boosted decision trees (GBDTs) [49] combine such decision trees sequentially, such that each decision tree focuses on the training samples misclassified by the previous trees.

GBDTs can be evaluated efficiently and they naturally handle multi-label problems. Moreover, for small sizes, the resulting model can be easily interpreted. Their main drawbacks are that their hyperparameters are difficult to tune, rendering GBDTs prone to overfitting, and that they are not robust (a small perturbation of the training data can result in very different models).

1.2.3 Our Choices

As we already mentioned, most of the models described above can be applied to each of the problems we study in this thesis. Even though these models differ in their requirements or in their capabilities, we still have to decide which one to use. We describe below the reasons behind our choice of algorithms for each application.

In Chapter 2, we build models to predict the next location a user will visit, based on his current context. MCs and DBNs are obvious choices to model such a problem: many of the state-of-the-art methods for mobility prediction use some variant of MCs. We thus include a simple first-order MC as a baseline, and design a problem-specific DBN as one

of our three models. We cannot use high-order methods that take more than the current location into account, as we only have the current context as input.

Because a user usually visits several places, the task of predicting his next move can also be seen as a multi-label classification problem, where the label to predict is the next location. We thus use ANNs and GBDTs for this task, as they can naturally handle multi-class problems. Moreover, the output of ANNs and GBDTs being usually dissimilar (GBDTs produce sharp and non-smooth decision regions while ANNs are more stable with respect to their inputs), we take advantage of this diversity by combining the resulting models.

In Chapter 3, we build models to predict the success of crowdfunding campaigns. The first two models rely on the amounts of money pledged to campaigns for their predictions. We take two approaches: one model uses the whole series of amounts, from the beginning of the campaign to the current amount, whereas the second model only uses the current amount. We want the first model to compare campaigns based on the shape of their trajectory as a whole, and not only on a few values. Therefore, we use a k-NN, as it gives equal weights to all input features. As the second model basically needs to capture the step-by-step evolution of campaigns, we use a first-order MC.

The other two models we develop rely on features extracted from the social network of Kickstarter and from Twitter. SVMs, ANNs, and GBDTs could all be used, but we choose to use SVMs, as the dimensionality of the input data and the number of models we need to train are reasonably small. Indeed, SVMs are more computationally costly to train, but their training is stable and always converges to a global optimum, facilitating the choice of hyper-parameters through cross-validation.

In Chapter 4, we predict the votes politicians will cast at the parliament based on their opinions expressed on a voting advice application. As we need to train a separate model for each vote and for each politician, and to choose the best hyperparameters for each of these models though cross-validation, we restrict ourselves to algorithms that are efficient to train and evaluate : LR and GBDT. As we found no improvement when using the non-linear method in our experiments, we use LR for simplicity and efficiency.

Finally, in Chapter 5, we predict the binary national outcome of a vote using the result of this vote in one small administrative region. The very nature of the problem makes GBDTs good candidates for this task, as they naturally capture the thresholding that occurs when computing the binary outcome of a vote from a proportion of "yes". We also compared simpler methods, such as LR, but found that GBDTs gave better results.

2 Where Will They Go?

> Coming back to where you started is not the same as never leaving.
>
> *Terry Pratchett*

Mobility is a central aspect of our life; the locations we visit reflect our tastes and lifestyle and shape our social relationships. Now, with smartphones and other connected devices being almost ubiquitous, our location is tracked, recorded, and analyzed at all times. Using this data, researchers and companies have been studying the mobility of people for more than a decade, to describe the intrinsic characteristics of mobility patterns, but also to better understand the behavior and relationships of people [59, 41, 87].

A better understanding of the mechanics and motivations behind human mobility is of high interest to many, from service providers to public administrations. Companies can use mobility data to enhance their services, by strengthening their infrastructure in locations with a high probability of visit, but also to better target promotions and advertisements. Similarly, urban planners can rely on estimates of frequentation based on mobility models to better design and size new urban developments. Mobility modeling is also critical in the study and prediction of the diffusion of infectious diseases [26, 122].

2.1 Problem: Human Mobility Prediction

In this chapter, we describe our winning contribution to *Next Place Prediction Task* of the Nokia Mobile Data Challenge (NMDC). Our task was "to predict the next destination of a user given the current context, by building user-specific models that learn from their mobility history and then applying these models to the current context to predict where the users go next" [77]. A context is described by the data collected from the mobile phone of the user (date, location of the user, cell tower id, WLAN, phone calls, etc.)

We present the dataset, in Section 2.2, and identify some of its characteristics that are at the root of its unpredictability. For instance, some users change their home location during the observation period. We develop techniques that enable us to detect and adapt to these changes.

We introduce, in Section 2.4, an artificial neural network tailored for mobility prediction, as well as other predictors based on graphical models and decision trees. These predictors exhibit similar average prediction accuracies. For each user, however, we observed a high performance variability between the models. In order to take advantage of this variability, in Section 2.5, we finally combine the models by using different blending strategies. We show that the combined models achieve a higher accuracy than any individual model.

2.2 Dataset

The NMDC was "a large-scale research initiative aimed at generating innovations around smartphone-based research, as well as community-based evaluation of related mobile data analysis methodologies" [77]. It was organized by Nokia and took place from January 2012 to April 2012. It featured an open track, in which participants were able to propose their own problem to study, and three dedicated tracks, each defining a specific problem for teams to solve: semantic place prediction, next-place prediction and demographic attributes prediction.

At the heart of this challenge was the dataset gathered during the Lausanne Data Collection Campaign (LDCC) [71]. This dataset consists of a rich set of features (locations, phone calls, text messages, application usage, etc.) recorded from the smartphones of 170 participants, over periods of time ranging from a few weeks to almost two years. This data was collected in a privacy-preserving manner, allowing for meaningful statistics to be gathered while the anonymity of participants was protected.

Each task had its own subset of the LDCC data. The Next-Place Prediction Task, the focus in this chapter, was assigned a subset of 80 users. For each user, the last 50 days of data were kept as a test set for the evaluation of each team's submissions, and the rest were used as training data.

For privacy reasons, all identifiers (phone numbers, WLAN SSIDs, contact names, etc.) were encrypted; but more importantly, physical locations were not released. Instead, for each user, the organizers of the NMDC first identified *places* — corresponding to circular areas with a 100-meter radius — by using both GPS and WLAN data. Then, they represented each place by a unique identifier. Consequently, a sequence of geographic coordinates is represented as a sequence of place identifiers.

The user's visits to these places are the data used for the prediction task. Each *visit* is defined by starting and ending times, and the visited place. In addition, several types

of data are available: accelerometer, application usage, GSM, WLAN, media plays, etc. The complete list can be found in the dataset description [77]. Given a visit and all the associated data characterizing it, our task was to predict the next place visited by the user after he left his current location.

At the end of the challenge, each participating team was allowed to submit five different sets of predictions, corresponding to visits from the undisclosed part of the LDCC data. Then, the organizers of NMDC evaluated the prediction accuracy of each participating team's submissions.

2.2.1 Constraints

We present below two major constraints (imposed by the rules of the NMDC) that restricted the range of methods we could use and that made our task more challenging:

User Specificity. To prevent cross-referencing people and places between users, sensitive data is user-specific: The identifiers are encrypted using different keys, and places are defined and numbered for each user independently. Moreover, the rules of the challenge explicitly forbid all participants to try and reverse this process, or make some links between users. We are therefore not allowed to build joint models over the user population, i.e., to learn from one user to make a prediction about another. For this reason, we build user-specific predictors, and consider each user independently.

Memoryless Predictors. As explained above, the input for the Next-Place Prediction Task is the *current* visit, along with all additional data recorded from the user's phone during that time. For the training phase, to train our models we have access to the whole sequence of visits of each user, enabling us, for example, to study the time intervals between visits. When being evaluated on the undisclosed part of the dataset, however, we are only given the *current* visit of a user, but not his *history*, i.e., the sequence of previous visits. If we were given the history, we could develop higher-order predictors that not only take into account the current place but also the sequence of places visited just before. Indeed, such information is very useful: If a user is currently at a transportation hub, e.g., a bus station, knowing whether he was home or at work just before greatly helps in predicting his next move. Because this information is not available to us in this challenge, we limit ourselves to *memoryless* predictors, i.e., methods that take into account only the current context, without any knowledge of the past.

2.2.2 Characteristics

Before developing our models, we study the characteristics of individual users, to gain insights into the specificity of their behaviors. For an example of the various types of mobility patterns encountered across users, we show in Figure 2.1 an intuitive representation of the mobility traces of three users selected from the dataset. The figure depicts a user's behavior over one year as a matrix, where each column is a day of the year and each line an interval of one hour. We map each place to a color, and leave blank the intervals of time during which we have no information about the user's location.

A few users, for example User 143 shown in Figure 2.1(a), have a very regular behavior, which seems to support the results (such as those presented by Song et al. [108]) claiming that human mobility is very predictable. However, similarly to User 1 shown in Figure 2.1(c), the majority of users shows no clear regular pattern in their behavior. Of course, a lack of visual regularity does not imply that there is no underlying structure in a user's mobility. We will see in Section 2.4.3 that we can still predict the behavior of such users with reasonable accuracy.

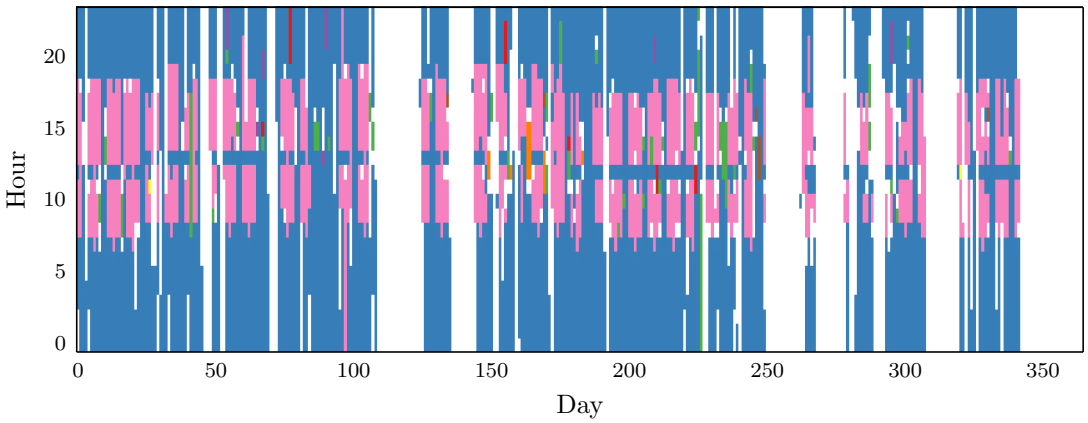
Interestingly, some users have a rather regular behavior, but with a significant change at some point during the data collection period. For example, User 13 (shown in Figure 2.1(b)) seems to have moved, with about two thirds of his nights spent at one place, and the last third of his nights spent at a different location. Such a non-stationarity in the behavior of users could lead to significant errors in the prediction of movements. Indeed, a change that happens late in the dataset would lead to the undisclosed test set beginning very differently from the training data.

These observations highlight the following salient characteristics of the data that we believe are critical to the prediction task:

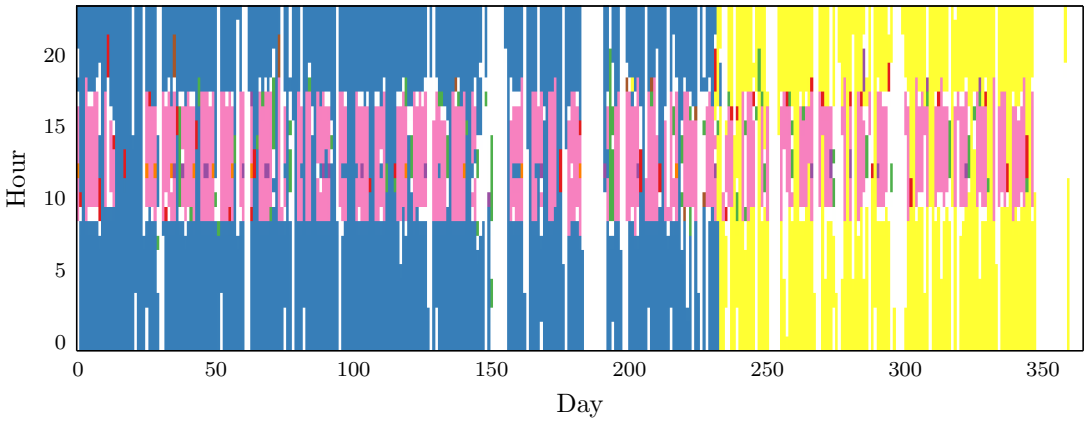
Non-Stationarity. We often observe a significant change in users' habits over time, as illustrated in Figure 2.1(b). The fact that some users change their home or work location right at the end of the observation period complicates the prediction task. To overcome this, we implement home-change detection mechanisms, as described in Section 2.3.2. Moreover, to get a realistic estimation of our predictors' performances, we keep the last part of the dataset as testing data, as explained in Section 2.3.3.

Data Gaps. We experience, for some users, periods (ranging from a few hours up to a few months) with no information about their behavior. Moreover, as shown in Figure 2.1(c), these gaps are sometimes followed by a change of mobility habits. To limit the effect of such transitions, our predictors take into account the possibility that we have missed some data between two detected visits, by including the directness of a visit (described in Section 2.3.1) as an input feature.

(a) User 143: regular mobility pattern



(b) User 13: regular mobility pattern with a change of home



(c) User 1: irregular mobility pattern with data gaps and non-stationarities

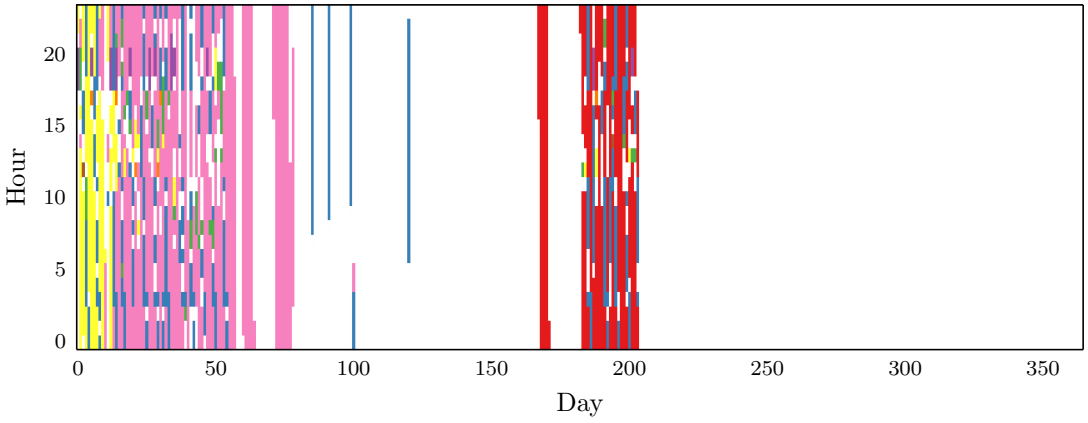


Figure 2.1 – Typical behaviors of users over one year, shown as matrices where each column is one day of the year and each line an interval of one hour. We map each place to a color and leave blank the intervals of time during which we have no information about the user's location. We show (a) a very regular user, (b) a home change, and (c) data gaps and non-stationarities.

Definition	Domain	Explanation
L	$\mathbb{N}$	Number of distinct places
$\mathcal{L}$	$\{1, \ldots, \mathrm{L}\}$	Set of visited places
k	$\mathbb{N}$	Time resolution
$X(n)$	$\mathcal{L}$	Place of the visit
$T_s(n)$	$\mathbb{N}$	Absolute starting time
$H_s^k(n)$	$\{1, \ldots, k\}$	Quantized starting hour
$D_s(n) = \mathrm{day}(T_s(n))$	$\{1, \ldots, 7\}$	Starting day
$W_s(n) = \mathrm{weekday}(T_s(n))$	$\{0, 1\}$	Indicates whether the visit starts on a weekday
$T_e(n)$	$\mathbb{N}$	Absolute ending time
$H_e^k(n)$	$\{1, \ldots, k\}$	Quantized ending hour
$D_e(n) = \mathrm{day}(T_e(n))$	$\{1, \ldots, 7\}$	Ending day
$W_e(n) = \mathrm{weekday}(T_e(n))$	$\{0, 1\}$	Indicates whether the visit ends on a weekday
$U(n)$	$\{0, 1\}$	Indicates whether there might be an unobserved place between $X(n)$ and $X(n+1)$
$C(n)$	$\{0, 1\}$	Indicates whether the user charged his phone during the visit

Table 2.1 – Definition and domain of the variables relative to a user, as well as those describing his nth visit.

Sparsity. The period of observation for some users is too short (less than 15 days) to reflect faithfully their mobility patterns. We overcome this lack of data by allowing for coarser segmentations of the day, using the time resolution parameter described in Section 2.3.1. We also limit the complexity of our predictors, so that they do not over-fit the data.

2.3 Framework

2.3.1 Notation

Before formally introducing our models, we first define the variables that describe the dataset. During the study period, a user makes N visits of variable duration to L distinct places, represented by the set $\mathcal{L} = \{1, \ldots, L\}$.

In Table 2.1, we list the variables corresponding to a user, as well as those relative to his nth visit. All time-relative variables are derived from the starting and ending times, which are given as absolute times. The binary variable $U(n)$ indicates whether there might be an unobserved place between $X(n)$ and $X(n+1)$. This situation typically arises when location data is partially missing between the two visits. In such case, we say

that the transition from $X(n)$ to $X(n+1)$ is not necessarily *direct*. The directness of a transition is given as a feature in the NMDC dataset.

To allow for various quantizations of the day, we introduce a *time resolution* parameter k. This enables us to consider a coarser segmentation of the day: instead of always splitting a day into 24 hours, we can choose to split it into k time periods. For instance, if $k = 2$, $H_s^k(n) \in \{1, 2\}$, with $H_s^k(n) = 1$ denoting the event that the nth visit starts between midnight and noon. Such a coarse segmentation can be helpful when training predictors for a user for which few data is available.

2.3.2 Home-Change Detection

To overcome one of the challenges caused by the non-stationary of the data (illustrated in Figure 2.1), we propose an algorithm for detecting changes in home location and adapt the learning process accordingly. At any moment t, we define *home* as the place where the user spends more than $T_{\text{threshold}}$ hours of his sleeping periods[1] during the interval of time $[t - T_{\text{history}}, t]$. The parameter T_{history} controls to which extent we keep in memory the past behavior of the user. At the end of the observation period corresponding to the training set, the user who changes his habits will have at least two places flagged as home. We declare the last place that was flagged as his *final home*. More importantly, the history of visits is modified as if the user's home has always been his final home.

Such a modification enables us to capture the user's habits while avoiding the lengthy process of adapting to a home change. The procedure for detecting home changes is summarized in Algorithm 1[2]. Empirical results show that applying our home-change detection algorithm results in a significant improvement in the prediction accuracy for the users who change their habits during the observation period.

2.3.3 Learning Procedure

For each user, we separate the data into three parts, as illustrated in Figure 2.2: we define the first 80 % of the data as set $\mathcal{A}$, the following 10 % as set $\mathcal{B}$ and the last 10 % as set $\mathcal{C}$. Finally, we call set $\mathcal{D}$ the undisclosed part of the data, on which our predictors were evaluated during the final part of the NMDC.

We divide the dataset deterministically—instead of doing for example cross-validation—because of the non-stationarity of the users' behavior, which we described in Section 2.2.2. We expect set $\mathcal{D}$ to be much more similar to the end of the dataset than to its beginning. Indeed, even if a user's behavior is globally non-stationary, it usually shows regular

[1] We define sleeping periods as the moments during the night where a user is most likely to be sleeping.

[2] Based on empirical evidence, we choose $T_{\text{history}} = 14$ days, $T_{\text{threshold}} = 18$ hours and the sleeping period to be between 3 a.m. and 6 a.m.

Algorithm 1: Home-change detection

Input: User's visits, $T_{\text{threshold}}$, T_{history}, sleeping period
Output: User's visits with all detected homes changed to the most recent one

$\mathcal{H} = \{\}$
for *each visit* n **do**
  $I = [T_s(n) - T_{history}, T_s(n)]$
  $P(n)$ = place where the user spent most of his sleeping periods in I
  $T_{P(n)}$ = time spent at $P(n)$ during the sleeping periods in I
  if $T_{P(n)} \geq T_{threshold}$ **then**
    $\mathcal{H} = \mathcal{H} \cup \{P(n)\}$
P_{home} = last element of $\mathcal{H}$
for *each visit* n **do**
  if $X(n) \in \mathcal{H}$ **then**
    $X(n) = P_{\text{home}}$

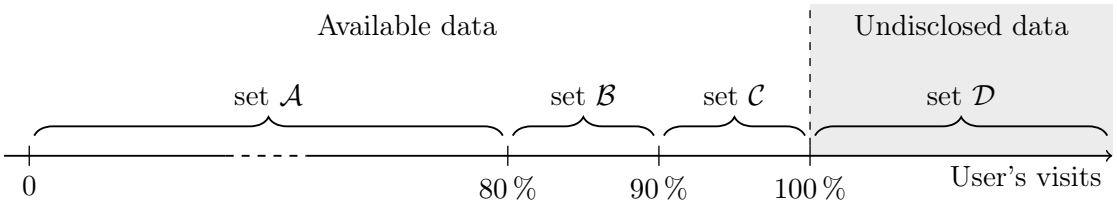


Figure 2.2 – Separation of a user's dataset. We define the first 80 % of the user's visits as set $\mathcal{A}$, the following 10 % as set $\mathcal{B}$, and the last 10 % as set $\mathcal{C}$. Finally, we call set $\mathcal{D}$ the undisclosed part of the dataset, on which submissions to the NMDC are evaluated.

patterns over short time intervals. Having set $\mathcal{C}$ as close as possible to set $\mathcal{D}$ maximizes the likelihood of their samples belonging to the same "stationary" period. Moreover, by training our predictors on "past" data and evaluating them on very recent data, we can test whether they are able to adapt to users' changes of habit.

The training is performed in three parts: First, we train each predictor on set $\mathcal{A}$, and evaluate its performance on set $\mathcal{B}$, to compare individual predictors. Then, we train each predictor on both sets $\mathcal{A}$ and $\mathcal{B}$, combine them (as explained in Section 2.5) using their performance computed before on set $\mathcal{B}$, and evaluate the prediction accuracy of the combination on set $\mathcal{C}$, . Finally, we train each predictor on sets $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ and combine them in order to predict for the samples in set $\mathcal{D}$.

2.3.4 Performance Measure

To evaluate the performance of a predictor on a set of visits, we consider its prediction accuracy, i.e., the proportion of samples for which it successfully predicts the next place.

First, consider a predictor $\boldsymbol{\phi}$: It takes as input $\boldsymbol{v}^{(n)}$, the data corresponding to the nth visit, and it outputs a probability distribution over the possible next places. More formally, consider $\Delta_L = \{\boldsymbol{x} \in [0,1]^L \colon \sum_{l=1}^{L} x_l = 1\}$, the $(L-1)$-dimensional simplex. Then, a predictor is a function

$$\boldsymbol{\phi} \colon \mathcal{V} \to \Delta_L,$$

where $\mathcal{V}$ is the space of features corresponding to a visit.

We could directly define the output of a predictor as the predicted next place. However, keeping a distribution over places as output enables us to combine predictors. Indeed, we can easily put several predictors together by computing a mixture of their output probability distributions over places. As explained in Section 2.5, there are different ways of choosing the weight of each predictor, each resulting in a unique global predictor.

The place $\hat{X}_n^{\boldsymbol{\phi}}$ predicted by $\boldsymbol{\phi}$ for the nth visit $\boldsymbol{v}^{(n)}$ is thus the most likely next place

$$\hat{X}_n^{\boldsymbol{\phi}} = \arg\max_{l \in \mathcal{L}} \phi_l(\boldsymbol{v}^{(n)}), \tag{2.1}$$

where $\phi_l(\boldsymbol{v}^{(n)})$ is the lth component of the vector output by $\boldsymbol{\phi}$ when given the data corresponding to the nth visit as input, i.e., the probability that the next visited place is l.

Finally, we define the prediction accuracy $A_{\mathcal{S}}(\boldsymbol{\phi})$ of the predictor $\boldsymbol{\phi}$ over the samples in set $\mathcal{S}$ as:

$$A_{\mathcal{S}}(\boldsymbol{\phi}) = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \mathbb{I}_{\left\{\hat{X}_i^{\boldsymbol{\phi}} = X(i+1)\right\}}, \tag{2.2}$$

where $X(i+1)$ is the true next place corresponding to the ith visit, and $\mathbb{I}_A$ is the indicator function, taking value 1 if the event A is true, and 0 otherwise.

2.4 Predicting Next Location

In this section, we present the different models we developed for predicting the next location of a user.

2.4.1 Artificial Neural Network

We consider next-place prediction as a multi-label classification task: Given the current place as input, and potentially some additional features, we want to predict the corre-

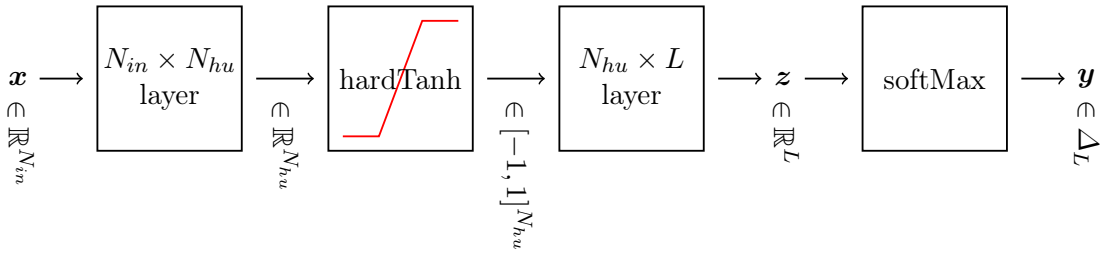


Figure 2.3 – Architecture of our 2-layer artificial neural network, with N_{in} inputs, N_{hu} hidden units and L outputs. A non-linear transfer function is applied between the first and second layer, and a softmax function is applied to the output, to obtain a probability distribution over places.

sponding class, i.e., the next visited location. With this approach, we train for each user a 2-layer artificial neural network (ANN) that has N_{in} inputs, N_{hu} hidden units and L outputs. The outputs are normalized to obtain a probability distribution over places. Such a network is illustrated in Figure 2.3.

Input Encoding

We encode places as categorical data: We represent each place l as a L-dimensional binary vector, where only the lth component is equal to 1, and all others to 0. Other visit attributes, such as the ending day $D_e(n)$ or the ending hour $H_e^k(n)$, can also be included as additional input features, and are encoded in a similar way if needed. For example, to use $(X(n), H_s^k(n), D_s(n))$ as inputs, we first encode them as binary vectors, as explained above, and then we simply concatenate them. The resulting input vector $\boldsymbol{x}$ is thus of size $N_{in} = L + k + 7$.

Training

To obtain a probability distribution $\{\boldsymbol{y} \in [0,1]^L : \sum_{l=1}^{L} y_l = 1\}$ from the output $\boldsymbol{z} \in \mathbb{R}^L$ of the second layer, we use a soft-max transfer function:

$$y_i = \text{softMax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{L} \exp(z_j)}, i \in \{1, \ldots, L\}.$$

A natural loss-function to train such a network is the *negative log-likelihood*. For the output $\boldsymbol{y} \in [0,1]^L$, corresponding to some input $\boldsymbol{x} \in \mathbb{R}^{N_{in}}$ and the ground truth $\boldsymbol{t} \in \{0,1\}^L$ (where $t_l = 1$ if l is the true next place, and 0 otherwise), we define the loss as

$$L(\boldsymbol{y}, \boldsymbol{t}) = -\sum_{l=1}^{L} t_l \log(y_l).$$

To find the optimal parameters of the ANN, we minimize the above loss function over the training set.

Features

For each user, we consider as input for the ANN different subsets of the following features, described in Table 2.1:

- $X(n)$
- $D_s(n), D_e(n)$
- $H_s^k(n), H_e^k(n)$ for $k \in \{2, 4, 6, 8, 12, 24\}$
- $W_s(n), W_e(n)$
- $C(n)$

By combining the above features in an exhaustive way[3], we obtain more than 200 ANNs for each user. We also tested more features, but chose not to use them in the end, as they did not improve the overall prediction performance.

Implementation

We implement our ANNs by using *Torch 5* [36], a machine-learning framework written in Lua. We use a stochastic gradient descent [78] to train each ANN, and we use early stopping as a regularization technique [83]. For all users, we empirically found that $N_{hu} = 50$ hidden units were sufficient.

To speed up the training, we use *hardTanh* as the non-linear transfer function between the two layers:

$$\text{hardTanh}(x) = \begin{cases} -1 & \text{if } x < -1, \\ x & \text{if } x \in [-1, 1], \\ 1 & \text{if } x > 1. \end{cases}$$

It is an approximation of the hyperbolic tangent, that is much faster to evaluate [35].

[3]The current place $X(n)$ is always used. We then include either $D_s(n)$ and $H_s^k(n)$, or $D_e(n)$ and $H_e^k(n)$, or both, for varying values of k. The other features $W_s(n), W_e(n)$ and $C(n)$ are used only when both starting and ending day/hour are included. We made this choice to reduce the number of combinations and thus the running time of our experiments.

2.4.2 Other Models

Gradient-Boosted Decision Trees

The second type of model we consider to predict the next location is a gradient-boosted decision tree (GBDT) [49, 50] . Similar to neural networks, GBDTs model the Next-Place Prediction Task as a multi-label classification problem, where the set of classes is $\mathcal{L}$, the places visited by the user. As input, we use the current place $X(n)$ and directness of the transition $U(n)$, as well as various subsets of the visit features described in Table 2.1: starting day and hour, ending day and hour, or both.

Two parameters control the structure of the GBDTs: the number of trees N_{tree} and the minimum number of observations N_{obs} required to create a terminal node in the trees. We implement our GBDTs in Python, and train several variants, with the subsets of features described above, $N_{\text{tree}} \in \{2, 5, 100\}$, and $N_{\text{obs}} \in \{2, 50, 100, 200, 500\}$.

Dynamic Bayesian Network

The last model we consider is a dynamic Bayesian network (DBN). In contrast to the first two families of models, which can be seen as *black-box* models, DBNs are tailored for the task at hand.

The rationale behind our DBN is as follows: The next place a user will visit depends on his current place and on the time at which he leaves it. The dependence between the current and next place is strong when the difference between the ending time of the current visit and the starting time of the next one is small (typically the case for direct transitions). However, as this time difference gets larger, the influence of the present place on the next one fades out, while the starting time of the next visit bears increasing importance. As we do not know the starting time of the next visit, the main challenge is to model its randomness, given carefully chosen information about the current visit.

As shown in Figure 2.4, the DBN captures these intuitions. The conditional distribution of the next place

$$\mathbb{P}(X(n+1) \mid X(n), H_e^k(n), W_e(n), U(n))$$

is a linear combination of place- and time-dependent distributions

$$\pi \underbrace{\mathbb{P}(X(n+1) \mid X(n))}_{\text{place-dependent}} + (1-\pi) \underbrace{\mathbb{P}\left(X(n+1) \mid H_e^k(n), W_e(n), U(n)\right)}_{\text{time-dependent}},$$

where $0 \leq \pi \leq 1$ is the parameter that governs the contribution of each distribution.

To predict future visits, we first learn the model parameters by using our training data. We introduce a latent variable z that dictates which mixture component is used for each

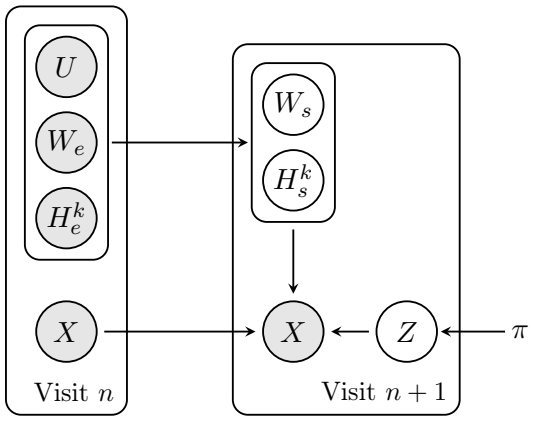


Figure 2.4 – Diagram of our dynamic Bayesian network. It explicitly models the next location $X(n+1)$ as depending both on the current location $X(n)$ and the starting hour $H_s^k(n+1)$ and starting day $W_s(n+1)$ of the next visit. The importance of these two components is governed by a mixture coefficient π. As the starting hour and day of the next visit are unknown when predicting, they are treated as latent variables, depending on the ending hour and day of the current visit, as well as the directness of the transition.

visit, and learn both distributions and the mixture coefficient π using an *expectation-maximization* algorithm [29, chap. 9] to maximize the likelihood of the data in set $\mathcal{A}$.

2.4.3 Results

As explained above, for each user we have several variants of each family of models, with different set of input features and/or parameters. We select, for each user and each family, the predictor that has the best performance on set $\mathcal{B}$, then train it again on both sets $\mathcal{A}$ and $\mathcal{B}$, to evaluate its performance over set $\mathcal{C}$. We show in Figure 2.5 the prediction accuracy for each of the three families of predictors presented above, averaged over all users. For comparison, we also include two baseline predictors: the first always predicts the most visited place, whereas the second uses a first order Markov chain.

Despite the fundamental differences between the three families of predictors, they exhibit very similar average prediction accuracies and outperform significantly the baseline predictors. However, when looking at users individually, the performance of each family varies greatly, as illustrated in Figure 2.6 for three selected users. We explain these variations by the diversity of types of user behaviors that are captured, with different levels of faithfulness, by each family of predictors.

Moreover, as shown in Figure 2.7, we observe a high variance of the predictability of users: We reach a prediction accuracy of 100 % for the most predictable user, and we predict correctly 0 % of the time for the least predictable one. Besides the intrinsic

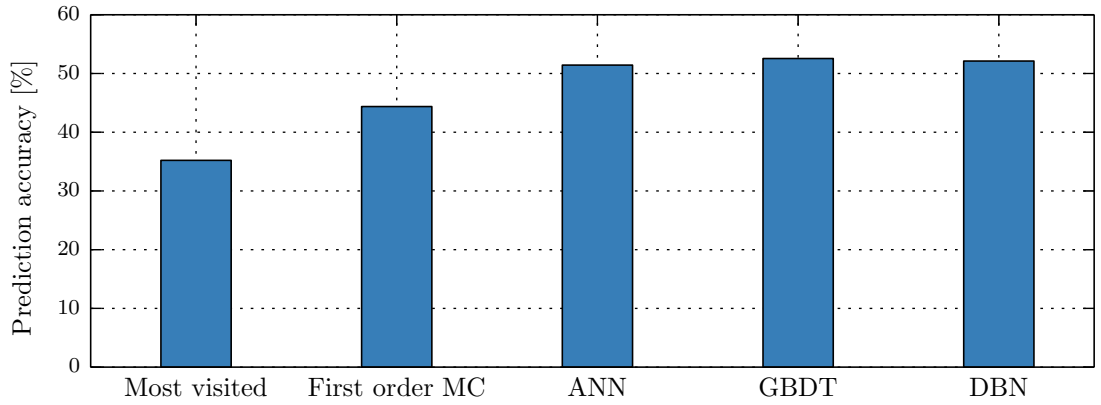


Figure 2.5 – Prediction accuracy on set $\mathcal{C}$ of the different families of models, trained on sets $\mathcal{A}$ and $\mathcal{B}$, averaged over all users. For each user and each family, we choose the set of features and parameters that yields the best prediction accuracy on set $\mathcal{B}$ when trained on set $\mathcal{A}$ only. As baselines, we also include one predictor that always outputs the most visited place, and one that uses a first order Markov chain.

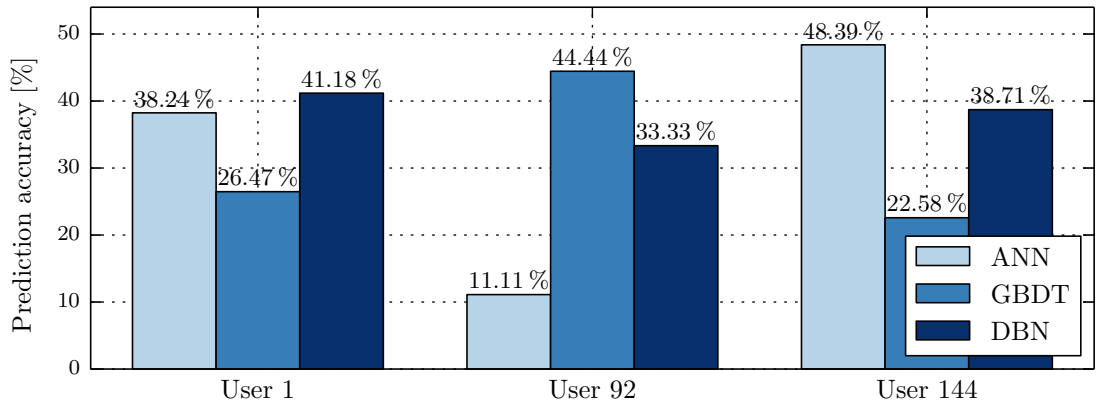


Figure 2.6 – Prediction accuracy of each family on set $\mathcal{C}$, for three selected users. For each family, we choose the best predictor on set $\mathcal{B}$. Even though Figure 2.5 shows that the three families of predictors have similar average performances, this figure clearly illustrates that, across users, their performances vary greatly.

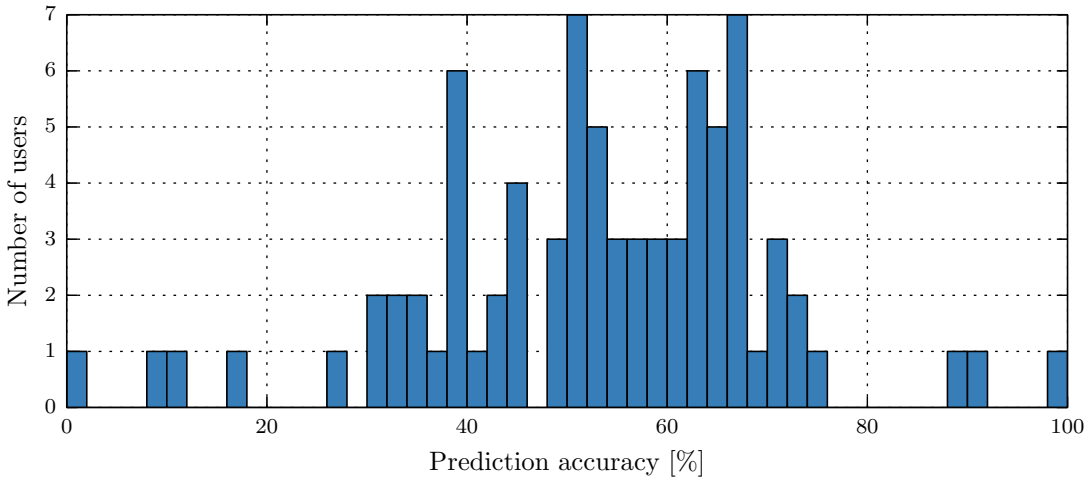


Figure 2.7 – Histogram of the prediction accuracy on set $\mathcal{C}$ per user. For each user, we chose the best predictor on set $\mathcal{B}$. There is a high variance in the predictability of users.

unpredictability of people, the major factor causing such a poor prediction-performance is the lack of data: the training data of the least predictable user only spans 12 days.

We obtained these results by using only basic features of the visits. In an attempt to improve the accuracy of our predictors, we included additional contextual information, such as distance between places, GSM cell towers, WLANs or accelerometer data, but we observed no improvement. We also implemented various preprocessing techniques, such as clustering and feature embedding, with no improvement either.

2.5 Combining Models

As expected, the accuracies of the models presented in Section 2.4 are not equal, but more importantly, each predictor makes different errors: certain samples for which a predictor fails might be those on which another excels, as illustrated below in Section 2.5.1. This idea prompts us to use blending, that is, to combine several predictors to take advantage of their diversity.

Notation

Before describing each blending strategy, we first define Φ, the set of all predictors trained on the data. This set can be split into three subsets of predictors $\Phi = \Phi_{\text{ANN}} \cup \Phi_{\text{GBDT}} \cup \Phi_{\text{DBN}}$, where each subset corresponds to all predictors of one same family. For instance, Φ_{GBDT} is the set of all predictors from the GBDT family.

A predictor $\boldsymbol{\phi}$ is defined by its family, some internal parameters, and the data it was trained on. Thus, we can refer to the predictor $\boldsymbol{\phi}$ in general, or to a specific predictor $\boldsymbol{\phi}(u)$, that was trained using the data of user u. We do not mention the dependence on u when it is obvious from the context.

2.5.1 Diversity of Predictors

We evaluate the diversity between two predictors by the proportion of samples, averaged over all users, for which *only* one of them predicts the correct next place. Therefore, the diversity between two predictors $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$ is defined as

$$\text{diversity}(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2) = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{N_u} \sum_{n=1}^{N_u} \left| \mathbb{I}_{\left\{\hat{X}_n^{\boldsymbol{\phi}_1}(u) = X_{n+1}(u)\right\}} - \mathbb{I}_{\left\{\hat{X}_n^{\boldsymbol{\phi}_2}(u) = X_{n+1}(u)\right\}} \right|,$$

where $\mathcal{U}$ is the set of all users, N_u is the number of samples for user u, $\hat{X}_n^{\boldsymbol{\phi}}(u)$ is the prediction made by the predictor $\boldsymbol{\phi}$ for the nth sample of user u, as defined in Equation 2.1, and $X_{n+1}(u)$ is the true next place.

To measure the diversity of a set of predictors $\boldsymbol{\Phi}$, we compute the average diversity between all possible pairs:

$$\text{diversity}(\boldsymbol{\Phi}) = \frac{1}{|\boldsymbol{\Phi}|(|\boldsymbol{\Phi}| - 1)} \sum_{\boldsymbol{\phi}_1 \in \boldsymbol{\Phi}} \sum_{\boldsymbol{\phi}_2 \in \boldsymbol{\Phi} \setminus \boldsymbol{\phi}_1} \text{diversity}(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2). \tag{2.3}$$

Similarly, we measure the diversity between two sets of predictors $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$ by computing the average diversity between all possible pairs:

$$\text{diversity}(\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2) = \frac{1}{|\boldsymbol{\Phi}_1||\boldsymbol{\Phi}_2|} \sum_{\boldsymbol{\phi}_1 \in \boldsymbol{\Phi}_1} \sum_{\boldsymbol{\phi}_2 \in \boldsymbol{\Phi}_2} \text{diversity}(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2). \tag{2.4}$$

Figure 2.8 shows the diversity of our predictors, both intra- and inter-family. To avoid taking into account predictors that have a poor prediction-performance, which would bias the diversity measure, we only keep the 10 best predictors of each family, according to their performance on set $\mathcal{B}$, and we compare their predictions over set $\mathcal{C}$.

With less than 6 % of samples for which, when taking a random pair of predictors, only one of them is correct, the DBN family is the most consistent of the three families. However, we see that DBN and ANN have more than 14 % of such samples, even though they have similar performances overall. This suggests that blending these families together will result in increased performance, which is confirmed in the results presented in Section 2.5.3.

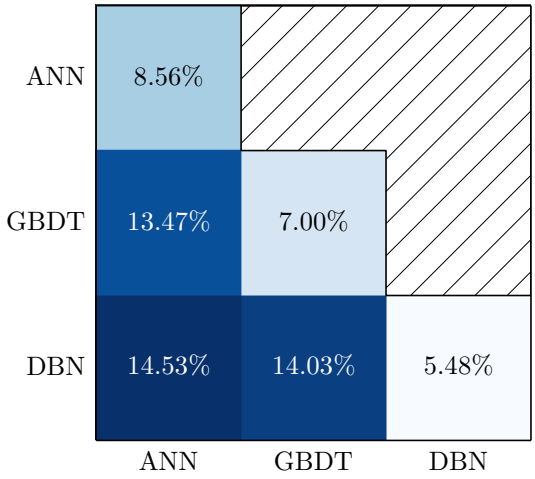


Figure 2.8 – Diversity between the best 10 predictors of each family, measured as the average proportion of samples for which, given a random pair of predictors, only one of them predicts the correct next place. The diagonal shows the diversity between predictors of the same family, as defined in Equation 2.3, whereas the other values show the diversity between two families of predictors, as defined in Equation 2.4. Predictors of different families clearly show a higher diversity than predictors of the same family, which suggests that blending all three families would result in a better accuracy than only using one individual family.

2.5.2 Blending Strategies

Below, we briefly explain the five blending strategies we used to generate our submissions to the NMDC. We designed these strategies to take advantage of (a) the diversity of our three families of predictors, and (b) the evaluation of the performance of each predictor. We compute the accuracy $A_{\mathcal{S}}(\phi)$ of each predictor ϕ on a given set $\mathcal{S}$ using Equation 2.2.

Strategy 1. For each user, we choose the predictor that has the best accuracy on set $\mathcal{B}$:

$$\phi_1 = \arg\max_{\phi \in \Phi} \{A_{\mathcal{B}}(\phi)\}.$$

Strategy 2. For each user, the predictor is a weighted mixture of all predictors, where the weight of each predictor is proportional to its average performance on set $\mathcal{B}$:

$$\phi_2 = \frac{1}{\sum_{\phi \in \Phi} A_{\mathcal{B}}(\phi)} \sum_{\phi \in \Phi} A_{\mathcal{B}}(\phi) \cdot \phi.$$

Strategy 3. For each user, we first select the best predictor of each family on set $\mathcal{B}$:

$$\begin{aligned}\boldsymbol{\phi}_{\text{ANN}} &= \arg\max_{\boldsymbol{\phi}\in\Phi_{\text{ANN}}} \{A_{\mathcal{B}}(\boldsymbol{\phi})\}, \\ \boldsymbol{\phi}_{\text{GBDT}} &= \arg\max_{\boldsymbol{\phi}\in\Phi_{\text{GBDT}}} \{A_{\mathcal{B}}(\boldsymbol{\phi})\}, \\ \boldsymbol{\phi}_{\text{DBN}} &= \arg\max_{\boldsymbol{\phi}\in\Phi_{\text{DBN}}} \{A_{\mathcal{B}}(\boldsymbol{\phi})\}.\end{aligned}$$

Then, we simply combine these three predictors uniformly:

$$\boldsymbol{\phi}_3 = \frac{1}{3}\boldsymbol{\phi}_{\text{ANN}} + \frac{1}{3}\boldsymbol{\phi}_{\text{GBDT}} + \frac{1}{3}\boldsymbol{\phi}_{\text{DBN}}.$$

Strategy 4. As we have many predictors (3 families with a large space of parameters), a majority of them has an average performance. Thus, when we blend them together, the few good predictions made by the best models are averaged out by all the other predictions. This is particularly true for Strategy 2. To prevent this and still guarantee some diversity, we first select the top 10 predictors[4] of each family (evaluated on set $\mathcal{B}$):

$$\Phi_{\text{top}} = \{\boldsymbol{\phi} : \boldsymbol{\phi} \text{ is one of the best 10 predictors of its family}\}.$$

The final predictor is a mixture of this subset of predictors, weighted by their performance on set $\mathcal{B}$:

$$\boldsymbol{\phi}_4 = \frac{1}{\sum_{\boldsymbol{\phi}\in\Phi_{\text{top}}} A_{\mathcal{B}}(\boldsymbol{\phi})} \sum_{\boldsymbol{\phi}\in\Phi_{\text{top}}} A_{\mathcal{B}}(\boldsymbol{\phi}) \cdot \boldsymbol{\phi}.$$

Strategy 5. We choose the predictor that has the best average accuracy over all users:

$$\boldsymbol{\phi}_5 = \arg\max_{\boldsymbol{\phi}\in\Phi} \left\{ \sum_{u\in\mathcal{U}} A_{\mathcal{B}}(\boldsymbol{\phi}(u)) \right\},$$

where $\mathcal{U}$ is the set of all users, and $\boldsymbol{\phi}(u)$ corresponds to the predictor $\boldsymbol{\phi}$ trained using the data of user u. Contrarily to the others, this strategy chooses the same predictor for all users.

We could also combine predictors using non-linear models such as neural networks, where we learn the optimal combination of the individual predictors. We could even go further and use sample-based blending techniques, where we adapt the combination of the blenders

[4] We tried several numbers of top predictors and empirically found that 10 were sufficient to ensure enough diversity while not "diluting" the good predictions.

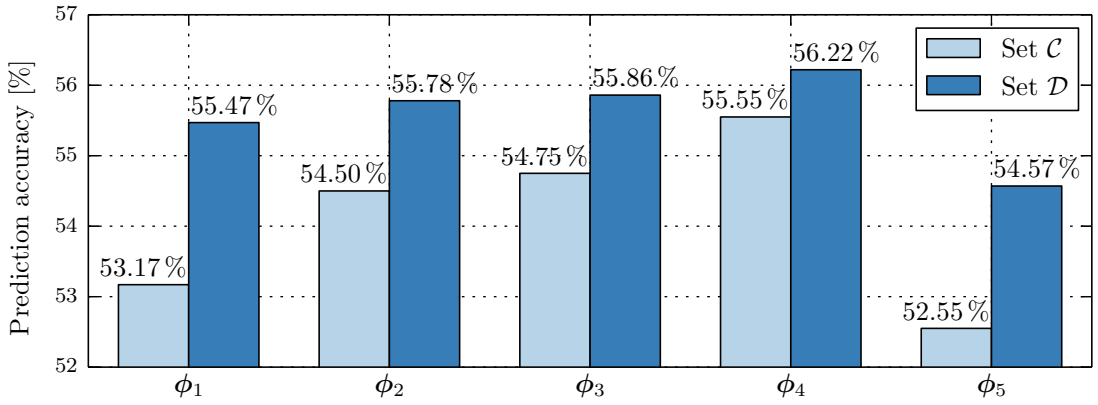


Figure 2.9 – Prediction accuracy on sets $\mathcal{C}$ and $\mathcal{D}$ of the five different blending strategies, described in Section 2.5.2.

to the features of each sample. However, due to the limited amount of data available in the NMDC, we limit ourselves to linear blenders. Indeed, more sophisticated blending techniques would require larger validation sets.

2.5.3 Results

Figure 2.9 shows the resulting prediction accuracies on set $\mathcal{C}$ of the blending strategies described above. The results confirm that making use of the diversity of the predictors, while taking into account their individual performances, increases significantly the prediction accuracy. For example, the most successful blending strategy (Strategy 4) is a mixture of the 10 best predictors of each family where the contribution of each predictor is proportional to its accuracy. This strategy outperforms Strategy 1 where we take simply the most accurate predictor for each user (55.55 % vs. 53.17 %, i.e., a relative improvement of 4 %).

These observations are corroborated by the prediction accuracies on the undisclosed set $\mathcal{D}$ (also shown in Figure 2.9), which were revealed by the organizers at the end of the NMDC. They confirm the accuracies measured on set $\mathcal{C}$: the ranking of the strategies relative to their accuracy is respected, and Strategy 4 is still the best with a prediction accuracy of 56.22 %.

2.6 Related Work

With the increasing availability of human-mobility datasets comes a growing scientific interest in studying human mobility and in understanding the mechanisms that govern it. The literature is composed of both descriptive and predictive approaches: the descriptive approach [70, 59, 108, 107, 99] is based on modeling both individual and group mobility. The main goal of the descriptive approach is to capture the statistical properties of human

mobility and to ensure, for the sake of realism, that the trajectories generated by mobility simulators exhibit these same properties. The predictive approach [25, 109, 33], however, focuses on the implementation of methods that accurately predict the locations users will visit in the near future. Naturally, the approach we take is predictive, as the main goal of our present work is to predict as accurately as possible the next place a user will visit. In this section, we present a selection of articles that we believe are representative of the rich literature about human-mobility prediction.

Song et al. [108] study the predictability of human mobility using a dataset of 45 000 mobile phone users. They try to answer the fundamental question, "To which extent is human mobility predictable?" They represent the mobility of each user as the sequence of detected cell towers. They quantify the users' mobility predictability by approximating the entropy rate of their mobility process (process generating a sequence of cell towers IDs). They find out that, on average, 1 bit of information is needed to describe the next cell tower a user will visit. They claim that humans are 93 % predictable, and that the predictability varies slightly across the whole population, which suggests that we are all equal in predictability. However, the authors do not implement a mobility predictor to verify empirically the claims presented.

Song et al. [109] evaluate the performance of several location predictors using a two-year trace of the mobility of over 6000 users. They represent the mobility of a user as the sequence of Wi-Fi access points detected. The authors claim that the major challenges faced when it comes to mobility prediction are the unseen contexts and the sudden change of users' habits. To overcome these drawbacks, the authors enhance their predictors with fallback and aging mechanisms, resulting in an enhanced prediction accuracy. Their best predictor is a second-order Markov chain with a fallback mechanism and has a median prediction accuracy of about 72 %.

Similarly, Scellato et al. [102] use a non-linear method for predicting the time and duration of a user's next visit to one of his significant places. Their method identifies patterns in a user's mobility history that are similar to his recent movements in order to predict his behavior. As stated in Section 2.2.1, we could not use high-order methods for the Next-Place Prediction Task because we have access to information about the present visit only.

In order to enhance the classic approach to mobility prediction, Cho et al. [33] study the influence of the social dimension on mobility: They claim that human mobility is a combination of periodic movements and seemingly random jumps that are correlated with the social network of the user. They develop a mobility model based on these observations and evaluate its performance on a mobility dataset composed of GPS points and cell tower IDs. The results do not show a systematic improvement of prediction accuracy when the social dimension is taken into account. However, the authors expect that, with denser datasets, the social dimension will bring significant improvement to mobility prediction.

Gao et al. [51] won third place in the NMDC. They present a probabilistic framework with fallback mechanisms, which relies on spatial and temporal features, similar to our tailored model presented in Section 2.4.2. Wang and Prabhala [120] ranked second in the NMDC. They propose a periodicity-based probabilistic model, similar to our way of combining the day of the week and hour of the day as input features, along with models that use a support vector machine. Both teams obtained an accuracy of 52 % on set $\mathcal{D}$, whereas our best strategy reached an accuracy of 56 %. Although they use features and models that are similar to those we proposed in this chapter, they do not address the problems caused by non-stationarities, such as home changes; nor do they combine the different models they develop to take advantage of their variability. We believe that these two differences were key in our winning the NMDC.

The related work we have introduced and, more generally, the studies on human mobility, rely heavily on empirical evidence: The authors analyze a mobility dataset in order to find interesting patterns, capture statistical properties or test the methods they implemented. The temptation is to draw from this analysis a conclusion about human mobility and its fundamental properties (distribution of distance between consecutive locations visited, predictability, etc.) without taking into consideration the specific characteristics of the dataset studied and their effect on the results found. For example, quantifying the predictability of human mobility depends strongly on the resolution of location information available: Predicting the next cell tower a user will visit could be straightforward [108], but finding his exact location within this cell is much more challenging.

2.7 Summary

In this chapter, we present the mobility predictors we developed for the Nokia Mobility Data Challenge. We use a wide range of techniques, including a dynamic Bayesian network and artificial neural networks. Moreover, we adapt these techniques to the characteristics of the data, by implementing mechanisms that ensure the adaptability of the predictors to the sudden changes in users' behavior and the sparsity of the data. In particular, a home-change detection algorithm enables us to improve significantly the prediction accuracy for the users with an important change of habits.

The three families of predictors we introduce obtain similar average prediction accuracies. However, their performance varies significantly across users, with some predictors failing when other succeed. In order to take advantage of the diversity of these predictors, we introduce several blending strategies that combine them into a global and more accurate predictor. Despite the simplicity of these techniques, the predictors — obtained after blending — are able to bring a relative improvement of up to 4 % over individual predictors. Each of the five blending strategies we submitted to the NMDC outperformed all the other participants' submissions, enabling our team to win the Next-Place Prediction Task.

3 Will They Get There?

> It is well known that a vital ingredient of success is not knowing that what you are attempting cannot be done.
>
> *Terry Pratchett*

The launch of ArtistShare [1] in 2003 marked the beginning of a new way of funding creative projects: crowdfunding. Contrary to traditional funding strategies, where an entrepreneur would obtain the money required to launch his business from banks or a few individual private investors, crowdfunding allows businessmen to cut intermediaries and obtain the money they need directly from their potential clients. In exchange for their money, investors usually receive either a reward, or equity in the funded company. Relatively little known before 2010, crowdfunding has gained traction recently, with more than $5.1 B raised worldwide in 2013 [30].

Kickstarter [7] is one of the largest crowdfunding websites; as of 2015, more than $1.5 B have been received in pledges from 7.8 million backers to fund more than 200 000 projects [69]. It is a reward-based crowdfunding platform: people pledging money towards a project receive rewards ranging from the acknowledgement of their participation to deep involvement in the project's development.

An important characteristic of Kickstarter is that its fundraising model is *all or nothing*. When launching a campaign, the creator sets a funding goal and a deadline. Once its deadline is reached, a campaign is considered successful *if and only if* it has reached its goal. In this case, backers actually pay the money they pledged and the project idea is realized. In the case where the goal is not reached, the campaign is considered as failed and no money is exchanged.

3.1 Problem: Crowdfunding Success Prediction

As only 38 % of campaigns reach their goal[1], for creators it is of high interest to know early on about the probability of success of their campaign to be able to react accordingly. Users whose campaigns are failing to take off might want to increase their visibility and start a social media campaign, whereas those whose campaigns are highly likely to succeed could already start working on their projects to get a head start, or look into possible extensions of their goal.

Similarly, backers could also benefit from a prediction. If the predicted probability of success of a campaign is low shortly after its launch, they could engage their friends and social network in backing it. When the predicted success probability is high, backers could also adjust their pledge, maybe reducing it a little in order to support another campaign, while being confident that this campaign will still succeed.

In this chapter, we present our work on predicting the success of Kickstarter campaigns. We first describe the data we collected for this task in Section 3.2. We then describe two classes of models that we developed: models that use the amount of pledged money to predict the campaign's success, in Section 3.3.1, and models that use other social data, in Section 3.3.2. Finally, we show in Section 3.4 that *combining* both classes of models significantly improves the prediction accuracy over individual models.

3.2 Dataset

Our dataset[2] consists of data collected on Twitter and on Kickstarter's website between September 2012 and May 2013.

3.2.1 Collecting the Data

We learn about new campaigns on the *Recently Launched* page of Kickstarter [9]. Once a new campaign is detected, its main characteristics, such as its category, funding goal and deadline, are collected and stored in a database. Then, we regularly, until the campaign reaches its end, crawl the page of each campaign to record the current amount of pledged money and number of backers. On average, we record the status of a campaign every 15 minutes, from its beginning to its end.

In parallel, we monitor[3] Twitter for any public tweet containing the keyword *kickstarter*. For each tweet matching our search, we record all its data in our database. To determine if

[1] As of July 2015. Globally, the success rate of Kickstarter campaigns has been slowly decreasing, as illustrated in Figure 3.1.

[2] The data presented in this chapter is available online [12].

[3] We use the Twitter Streaming API [117] to search for the keyword *kickstarter*. Because few tweets match this search query compared to the global number of published tweets, we know that we get a large uniform fraction of the relevant tweets (usually close to 100 %) [116].

the tweet is related to a particular campaign, we search its text for a link to a Kickstarter page. If any Kickstarter URL is found, the tweet is identified in the database as a reference to the corresponding campaign. We thus have, for each campaign, all public tweets related to it.

In addition to using Twitter, Kickstarter integrates Facebook into its website, as another way of spreading the word about campaigns. However, contrary to Twitter, most Facebook posts are not public, being usually restricted to the user's friends. As a result, a search similar to the one described above performed on Facebook usually yields very few results. For this reason, we only use Twitter in our dataset.

Finally, we regularly crawl the *Backers* page of each campaign to get the list of users who pledged money and to store them in our database. As this last step is time-consuming to perform, it is done only every couple of days, resulting in only a few snapshots of the list of the backers of a campaign. This means that we only have a coarse resolution of the time at which each backer joined a campaign, rendering difficult to extract meaningful information from these times. For this reason, we do not use them in our models.

3.2.2 Overview

Table 3.1 describes the global statistics of our dataset, separately for successful and failed campaigns, as well as the combined total. Table 3.2 shows the average statistics of individual campaigns. As one could expect, failed campaigns have on average a much higher goal than those that succeed (close to four times higher), but it is interesting to note that they also have a longer duration[4]. Moreover, we have a nearly even split between successful and failed campaigns, with more than 48 % of campaigns that reach their funding goal. The reported global success rate of Kickstarter is lower, with 38 % of successful campaigns overall [69]. We explain this difference by the fact that our dataset was collected during an intermediary period of Kickstarter, during which it was popular enough to attract backers, but new enough to not get many low-quality submissions.

Interestingly, the global success rate of Kickstarter campaigns has slowly decreased over the last few years. To illustrate this trend, we briefly consider an extended version[5] of our dataset that contains the campaigns collected by our crawler between October 2012 and July 2015. We show in Figure 3.1 the monthly success rate of these campaigns, i.e., the proportion of campaigns that reach their funding goal. We see that it varies slightly from month to month[6] and that the global trend is downwards, as highlighted by the linear least-square fit shown as a dashed line.

[4]Project creators can choose the duration of their campaign. The default value is 30 days, with a maximum of 60 days.

[5]As said above, the data we study in this chapter was collected between September 2012 and May 2013.

[6]The unusual peak in September 2013 comes from a downtime of our crawler, during which we mostly collected campaigns through Twitter, thus biasing our data towards successful campaigns.

	Campaigns			Pledges			
	Count	Ratio	Users	Count	Amount	Average	Tweets
Successful	7739	48.24 %	1 207 777	2 030 032	141 942 075	69.92	564 329
Failed	8303	51.76 %	171 450	212 195	16 084 581	75.80	173 069
All	16042	100.0 %	1 309 295	2 242 227	158 026 656	70.48	737 398

Table 3.1 – Global statistics of our dataset of Kickstarter campaigns. Users are unique people who have backed at least one campaign. We show the total number of pledges, the corresponding amount in $, and the average pledge for these users.

	Goal ($)	Duration (days)	Backers	Final amount	Tweets
Successful	9595	30.89	262	216.60 %	73
Failed	34 693	33.50	25	11.40 %	20
All	22 585	32.24	139	110.39 %	46

Table 3.2 – Average campaign statistics of our Kickstarter dataset. The final amount is the total amount of pledged money reached at the end of the campaign, relative to its goal.

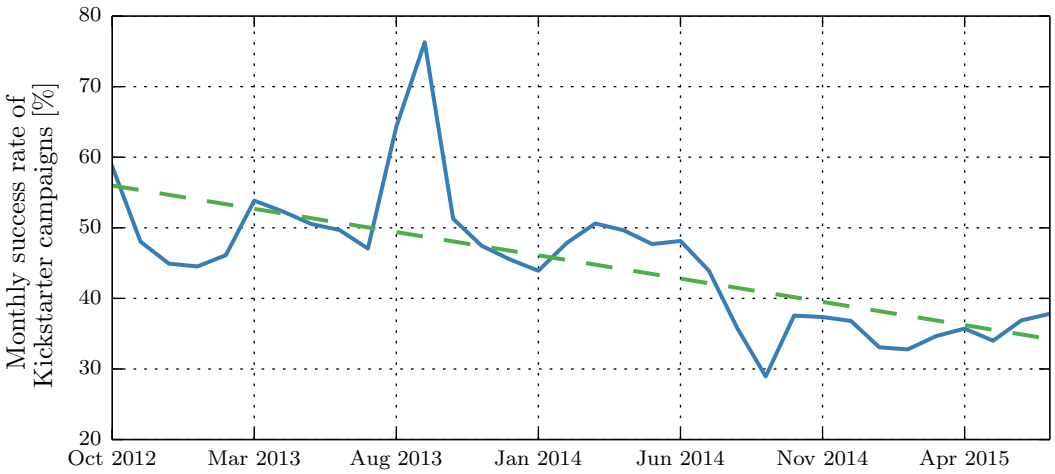


Figure 3.1 – Monthly success rate of Kickstarter campaigns. The solid blue line shows for each month the proportion of campaigns that successfully reached their funding goal. The dashed green line shows the linear least-square fit, highlighting the downward trend of the success rate. The peak in September 2013 comes from a downtime of our crawler, where we mostly gathered campaigns through Twitter, thus biasing our collected data towards successful campaigns.

3.2.3 Preprocessing

As explained in Section 3.2.1, our crawler regularly visits the page of each campaign to get its current *status*, i.e., the amount of pledged money and the number of backers. On average, a campaign's status is sampled every 15 minutes, resulting in thousands of samples at irregular time intervals.

To be able to compare campaigns with each other, we resample each campaign's sequence of statuses to obtain a fixed number of $N_s = 1000$ statuses. We normalize the time at which each status was captured with respect to the campaign's launch date and duration. To obtain a normalized amount, we divide the current amount of money pledged of each status by the goal amount of the campaign.

A campaign c is thus characterized by its funding goal $g(c)$, launch date $l(c)$, duration $d(c)$, final state $f(c)$ (equal to 1 if the campaign succeeded, 0 otherwise) and a sequence of status samples $\{\boldsymbol{s}_i(c)\}_{1 \leq i \leq N_s}$. Each status $\boldsymbol{s}_i(c) = (m_i(c)\ b_i(c))$ is itself composed of the normalized amount of money pledged $m_i(c)$ (i.e., we divide the amount by $g(c)$) and the number of backers $b_i(c)$.

Because each campaign is resampled to have N_s evenly-spaced statuses, the time $t_i(c)$ of the ith status $\boldsymbol{s}_i(c)$ is simply defined as

$$t_i(c) = l(c) + \frac{i-1}{N_s - 1} d(c),\ 1 \leq i \leq N_s.$$

Table 3.3 summarizes the variables describing a campaign c. Figure 3.2 illustrates the evolution of $m_i(c)$ over time, for successful and failed campaigns separately. Although there are many campaigns that clearly succeed (reaching their goal after less than 20 % of their duration), or clearly fail (never passing 10 % of their goal), we see that there are also many intermediary campaigns. These campaigns follow the global trend, with a sharp jump at the beginning, a steady increase in the middle, and usually a surge in pledges at the end [73]. However, a significant portion of these campaigns never reach their goal, as shown in Figure 3.2(b).

3.3 Models for Success Prediction

An important characteristic of our prediction problem is that we want to be able to predict the success of a campaign given its status at various stages: just after its beginning, two days into the campaign, etc. This means that the longer a campaign has been running, the more information a model could take into account to make its prediction. There are thus two approaches we could take: (a) use models that are able to handle variable amounts of input data, or (b) create a distinct model for each state of progress in a

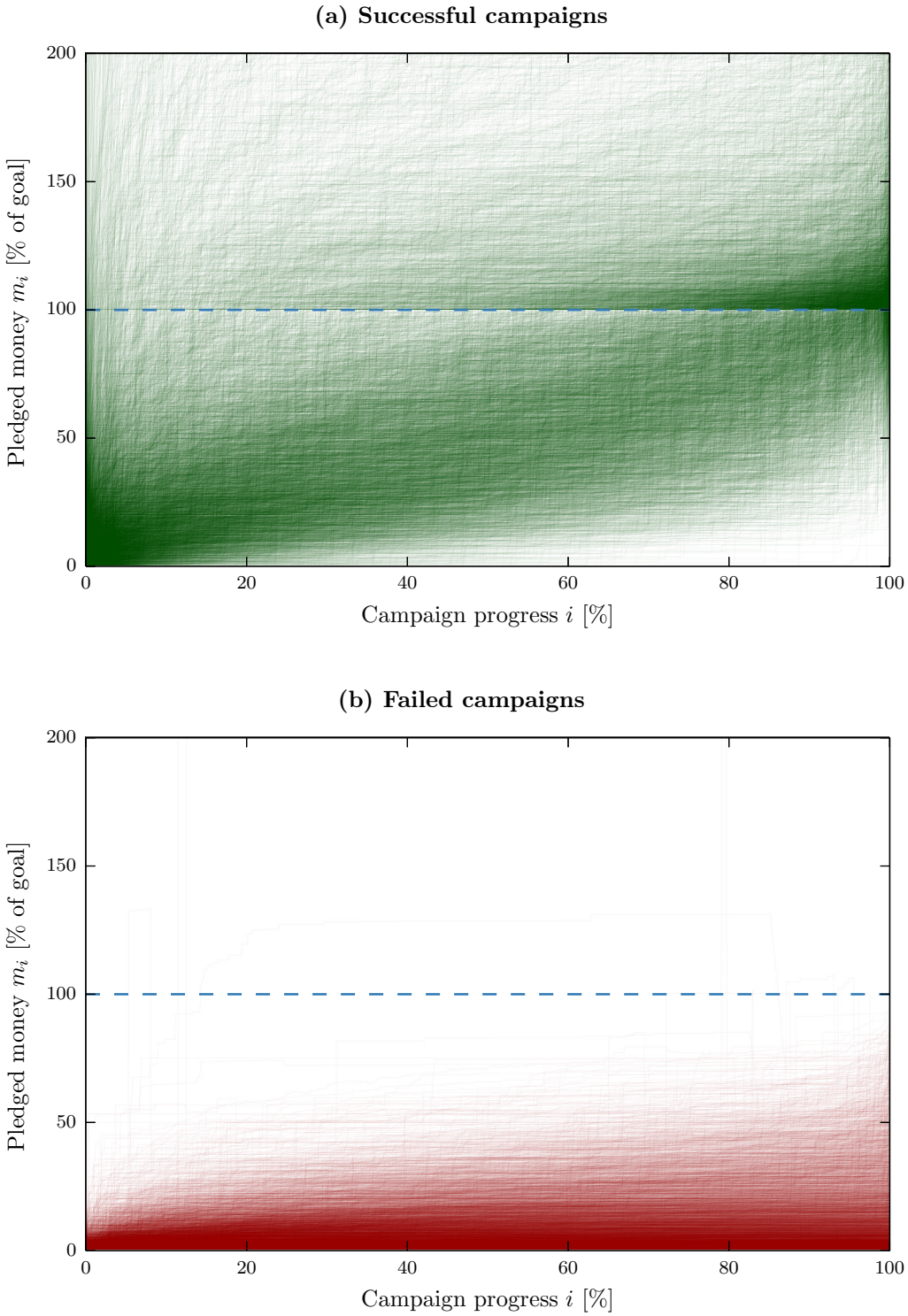


Figure 3.2 – Visualization of the amount of money pledged towards campaigns, from their beginning (0 %) to their end (100 %). The amounts are normalized with respect to the goals of the campaigns. We show (a) successful campaigns (i.e., campaigns that have raised at least their goal by the end) and (b) failed campaigns. The dashed horizontal line corresponds to the funding goal.

Variable	Description
$g(c)$	Funding goal
$l(c)$	Launch date
$d(c)$	Duration
$f(c)$	Final state (1 if successful, 0 otherwise)
$\{\boldsymbol{s}_i(c)\}_{1\leq i\leq N_s}$	Sequence of resampled statuses
$t_i(c)$	Sample time of the ith status
$m_i(c)$	Normalized amount of money pledged at time t_i
$b_i(c)$	Number of backers at time t_i

Table 3.3 – List of the variables describing a campaign c. The campaign statuses (time of capture, current pledged amount, and number of backers) captured by our crawler are resampled to obtain $N_s = 1000$ statuses $\{\boldsymbol{s}_i(c)\}_{1\leq i\leq N_s}$ at regular time intervals, as explained in Section 3.2.3.

campaign. As models with variable input sizes are intrinsically more complicated to handle, we choose to learn separate models for each state of progress.

Below, we introduce the models we use for predicting the success of a campaign. We investigate two families of models: in Section 3.3.1, we consider models that only rely on the amounts of pledged money to make their predictions. In contrast, the models presented in Section 3.3.2 do *not* take money into account, but instead rely solely on social features.

As explained above, all these models use only partial information: to predict the success of a campaign c, they only consider a prefix $\{\boldsymbol{s}_i(c)\}_{i\in\mathcal{I}}$ of its sequence of statuses, where $\mathcal{I} = \{1, \ldots, S\}$ and $1 \leq S < N_s$. When presenting results below, we show the performance of models for $S = 5, 10, 15, \ldots, 995$, i.e., every half a percent of the duration of a campaign.

3.3.1 Money-Based Models

The first family of models that we define only uses the sequence of amounts of money pledged $\{m_i(c)\}_{i\in\mathcal{I}}$, which we call the *trajectory*, to predict the outcome of a campaign c. When considering the trajectory of a campaign, two approaches can be used to predict its outcome: (a) assume that the whole trajectory is important (i.e., that it matters if the campaign just got to its current amount or has not gotten any pledges for a while), or (b) simply consider the current amount of pledged money and discard the rest of the trajectory. We investigate both approaches: the first model takes the whole trajectory into account, whereas the second just uses the current amount.

k-Nearest Neighbors

Our first model is a k-Nearest Neighbors (k-NN) classifier [38]. Given a new campaign c, its partial trajectory $\{m_i(c)\}_{i\in\mathcal{I}}$ and a list of campaigns for which the ending state is known, k-NN first computes the distance between c and each known campaign c':

$$d_{\mathcal{I}}(c,c') = \sqrt{\sum_{i\in\mathcal{I}} (m_i(c) - m_i(c'))^2}.$$

Then, it selects $\text{top}_{k,\mathcal{I}}(c)$, the k known campaigns that are the closest to c with respect to the distance defined above. Finally, it computes the probability of success $\phi_{k\text{-NN}}(c,\mathcal{I})$ of c as the average final state of these k nearest neighbors:

$$\phi_{k\text{-NN}}(c,\mathcal{I}) = \frac{1}{k} \sum_{c'\in\text{top}_{k,\mathcal{I}}(c)} f(c').$$

Markov Chain

Our second model uses the campaign trajectories to build a time-inhomogeneous Markov Chain that characterizes their evolution over time, and then it simply uses the current amount a new campaign has reached to compute its probability of success.

To define the Markov Chain, we first need to discretize the (time, money) space into a $T \times M$ grid. This means that we resample time, to have T samples of each campaign's trajectory, and we map the amounts of pledged money to a set $\mathcal{M}$ of M equally-spaced values, ranging from 0 to 1[7]. For example, if $M = 3$, $\mathcal{M} = \{0, 0.5, 1\}$.

We thus obtain for each campaign c a sequence of discretized amounts of money pledged $\{m'_j(c)\}_{1\leq j\leq T}$. Let M'_j be the random variable associated with the discretized amount m'_j. The Markov model defines, for each time j, the transition probability

$$P_{m,m'}(j) = \mathbb{P}\left(M'_{j+1} = m' \mid M'_j = m\right).$$

Putting these probabilities together, we define the $M \times M$ transition matrix $\boldsymbol{P}(j) = (P_{m,m'}(j))_{m,m'\in\mathcal{M}}$, for each $1 \leq j < T$. These transition matrices are not specific to a campaign but learned globally over all campaigns.

Success Prediction with the Markov Model By using the transition probabilities described above, predicting the success of a new campaign c at time i is straightforward, given its current amount of pledged money $m_i(c)$. First, we compute the corresponding resampled time j and discretized amount $m'_j(c)$. Then, the probability of c being successful

[7] All values larger than 1 are mapped to 1.

is simply the probability of its final discretized amount $M'_{N_s}(c)$ to be equal to 1 (100 % of the goal), given that is current discretized amount is $m'_j(c)$.

We compute this success probability $\phi_{\text{Markov}}(c, j)$ given its current discretized amount of pledged money m'_j as

$$\begin{aligned}\phi_{\text{Markov}}(c,j) &= \mathbb{P}\left(M'_{N_s}(c) = 1 \mid M'_j(c) = m'_j(c)\right) \\ &= \sum_{m' \in \mathcal{M}} \mathbb{P}\left(M'_{N_s}(c) = 1 \mid M'_{j+1}(c) = m'\right) \mathbb{P}\left(M'_{j+1}(c) = m' \mid M'_j(c) = m'_j(c)\right) \\ &= \left[\prod_{j'=j}^{T-1} \boldsymbol{P}(j')\right]_{m'_j(c),1},\end{aligned}$$

where the last step is obtained by repeatedly applying the law of total probability.

Example Figure 3.3 gives an example of such a Markov model, for $T = 50$ time steps j, and $M = 30$ possible values for the discretized amounts m'_j. Figure 3.3(a) represents the transition probabilities $P_{m,m'}(j)$ as edges between each pair (m_j, m'_{j+1}), for all times $1 \leq j < T$. The thickness and color of these edges represent their likelihood, with thin orange edges being unlikely, and thick purple edges being likely. We observe that, under this model, campaigns that have trouble getting pledges early on are likely to never take off (with very thick purple edges linking the low amounts of pledged money). We also remark that there exist steep transitions from low to high amounts, both in the very early stages and close to the end of a campaign. These correspond to the general trends that we observe in the shape of trajectories in Figure 3.2.

Figure 3.3(b) shows the probability of success associated with each state m'_j, computed as explained above. We clearly see a transition, with a thin yellow band representing states with a success probability close to 50 %. States shown in green, in the upper-left part above the line, have a high probability of success, while those in red, in the lower-right part below the line, are more likely to fail.

3.3.2 Social Models

We now introduce the second family of models. Contrary to those presented above, which use the amount of money pledged to predict the success of campaigns, the social models use only side information obtained from Twitter and Kickstarter's social graph. To build these models, we simply extract social features, as described below, add some features about the campaign such as its goal and duration, and then use a support-vector machine (SVM) [37] with a Gaussian Radial-Basis Function (RBF) kernel to predict whether a

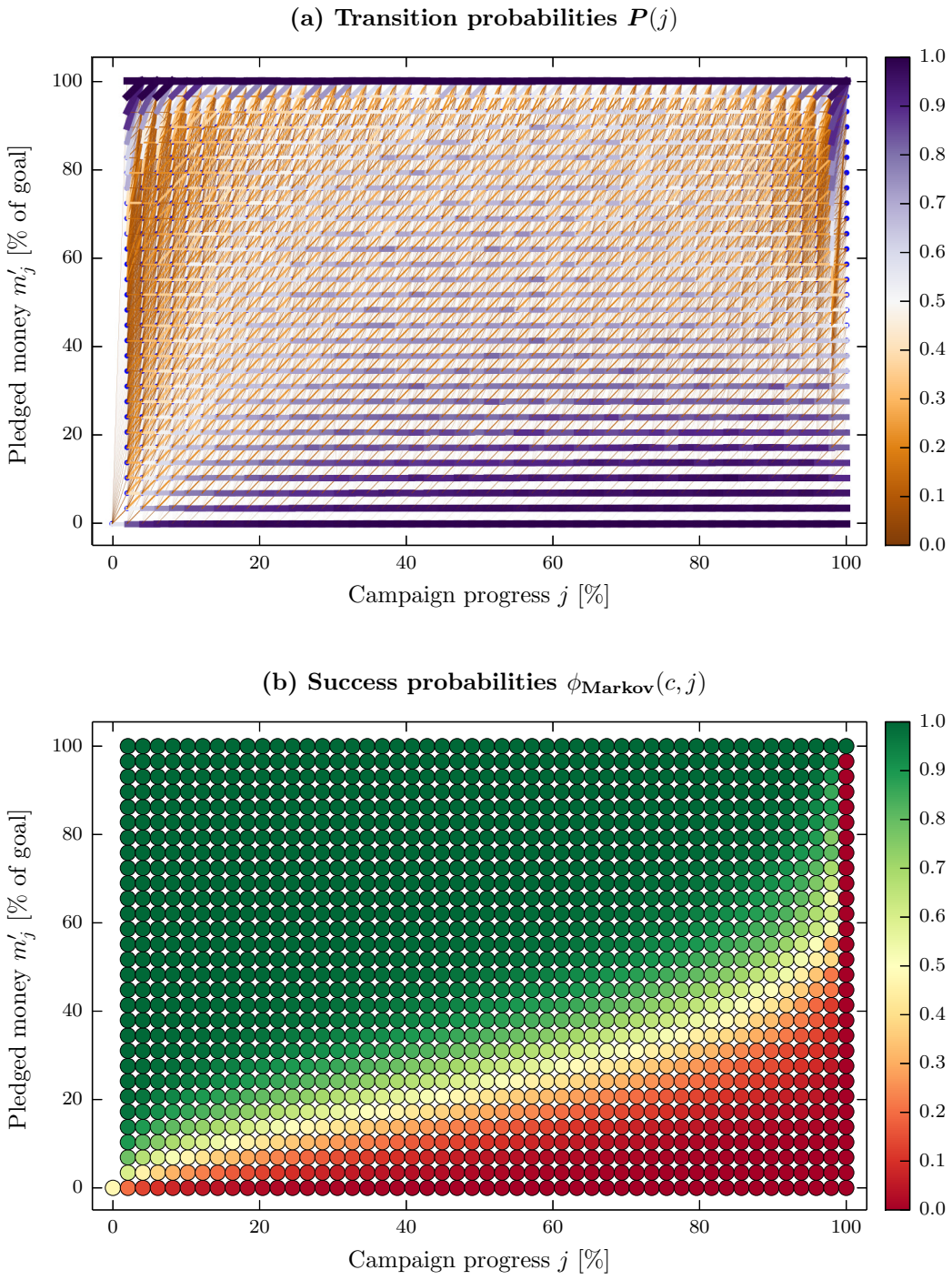


Figure 3.3 – Illustration of the Markov model, with $T = 50$ and $M = 30$. We show (a) the transition probabilities between the money states of each time step and those of the next. The width and color of the edges reflect their likelihood: thin orange edges are unlikely, thick purple eges are likely. We also show (b) the success probability of each state, i.e., the probability that a campaign reaches its goal if it is currently at that state. We see a clear transition, between states where campaigns have a high probability of success (upper-left part), and those where failure is more likely (lower-right part).

campaign will succeed or not, based on these features. As we only have a few features for each sample and our dataset is of reasonable size, we chose to use a SVM to be guaranteed to have the optimal classifier for each set of hyperparameters.

Tweets

We extract the first set of social features from Twitter: Indeed, we expect that the chatter about a campaign should be a good indicator of its popularity. As mentioned in Section 3.2, we have, for each campaign c, the list of all public tweets that contain a URL pointing to its page. As each tweet $\text{tweet}_n(c)$ has a timestamp, we can select the subset of tweets $\mathcal{T}_t(c) = \{\text{tweet}_n(c) \mid \text{timestamp}(\text{tweet}_n(c)) < t\}$ that were published before a time t. Using $\mathcal{T}_t(c)$, we extract the following features about each campaign c at time t:

- number of tweets, replies and retweets,
- number of users who tweeted,
- estimated number of backers[8].

We then add the campaign's goal $g(c)$ and duration $d(c)$ to these features and feed them to a SVM, resulting in a model $\phi_{\text{tweets}}(c, t)$.

Campaigns/Backers Graph

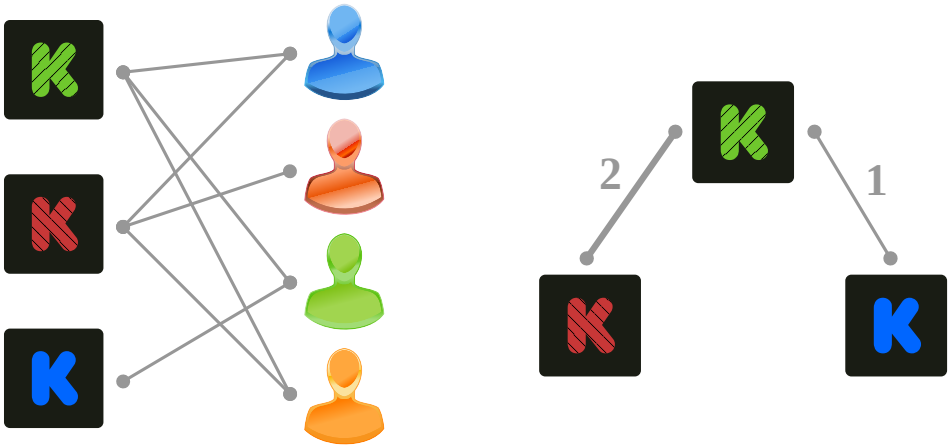
Although the features extracted from Twitter contain information about a campaign's popularity, they tell us nothing about the identity of the backers or about their behavior on Kickstarter. To get this information, we investigate the social graph of Kickstarter. This graph G_1 contains all campaigns and backers as vertices, and it has an edge between a campaign c and a backer b, if and only if b backed c. G_1 is thus an undirected and unweighted[9] bipartite graph.

From G_1, we can extract the co-backers graph G_2: it is the projection of G_1 onto the campaigns vertices. G_2 is an undirected weighted graph, where vertices are campaigns and the weight of an edge between two campaigns c_1 and c_2 is the number of backers who have pledged money to both c_1 and c_2. Figure 3.4 shows an example of (a) a campaigns/backers graph G_1 and (b) the corresponding co-backers graph G_2.

Using G_1 and G_2, we can now extract a second set of social features. We consider a new campaign c whose probability of success we want to estimate at some time t. We simply

[8] We estimate the number of backers by counting the number of tweets that contain texts such as "I just backed project X", which is the default message proposed by Kickstarter.

[9] It would be interesting to consider a weighted version of this graph, where the weight of each edge corresponds to the amount of money pledged. Unfortunately, we do not have access to this information, and thus can only consider the unweighted version.



(a) Campaigns/backers graph G_1 (b) Co-backers graph G_2

Figure 3.4 – Example of (a) a campaigns/backers graph G_1 and (b) the corresponding co-backers graph G_2. G_1 contains both campaigns (left) and backers (right) as vertices, and has an edge between a campaign c and a backer b if and only if b pledged money to c. G_2 is the projection of G_1 onto the campaigns vertices, where the weight of an edge between two campaigns represents their number of common backers.

integrate c into G_1 and G_2, using its list of backers at time t to add the necessary edges in both graph.

Then, we extract the following features of c:

- number of campaigns with co-backers (i.e., the degree of c in G_2),
- number and proportion of these campaigns that are successful,
- number of backers,
- number and proportion of first-time backers[10].

As with tweets, we then add the campaign's goal $g(c)$ and duration $d(c)$ to these features and feed them to an SVM, resulting in a model $\phi_{\text{graph}}(c, t)$.

3.3.3 Training and Performance Evaluation

In order to train our models, select their hyperparameters and evaluate their performance, we separate the dataset into 3 parts: 70 % of the campaigns are selected as the *training set*,

[10]First-time backers are users that pledged money only to the campaign c. Intuitively, a campaign that is able to get new people to sign up on Kickstarter might be more likely to succeed.

Model	Selected hyperparameters
k-NN	$k = 25$
Markov	$M = 30$
Tweets	$C = 1000, \gamma = 0.1$
Campaigns/backers graph	$C = 1000, \gamma = 0.001$
Combined	$C = 100, \gamma = 0.1$

Table 3.4 – Hyperparameter values selected for the different models. We use 70 % of the data as training data and 20 % as validation data, and select the best hyperparameters with the best average performance on 10 random assignments.

20 % as the *validation set* and the remaining 10 % as the *test set*. These sets are randomly chosen and all results presented below are averaged over 10 different assignments.

Each of the models presented above has some hyperparameters to be tuned:

- for k-NN, the number of neighbors k,
- for Markov, the number of money values M,
- for the two social models, the SVM parameters (soft margin penalty C and RBF kernel coefficient γ).

We select the best hyperparameters for each model by doing an exhaustive search on a wide range of values and by evaluating the corresponding average prediction accuracy on the validation set, for 10 random assignments. All models are implemented in Python, and we use the SVM implementation of scikit-learn [89]. We report the selected hyperparameters in Table 3.4.

3.3.4 Results

Figure 3.5 shows the prediction accuracy of the models presented above. Every 0.5 % of the campaign duration, we train a new model on 90 % of the campaigns and predict the success of the remaining 10 % of campaigns, based on their status at that time. We report the median prediction accuracy of each model over 10 random assignments of train/test data, plus/minus one standard deviation. In addition to the model we introduced, we show as a baseline the model developed by Greenberg et al. [60]. This baseline uses static campaign attributes, such as category, goal, and whether they have a video description or not, to predict the success of campaigns before their launch. This approach obtains a prediction accuracy of 68 %. This accuracy does not change with the progression of the campaign, as the baseline only uses static attributes.

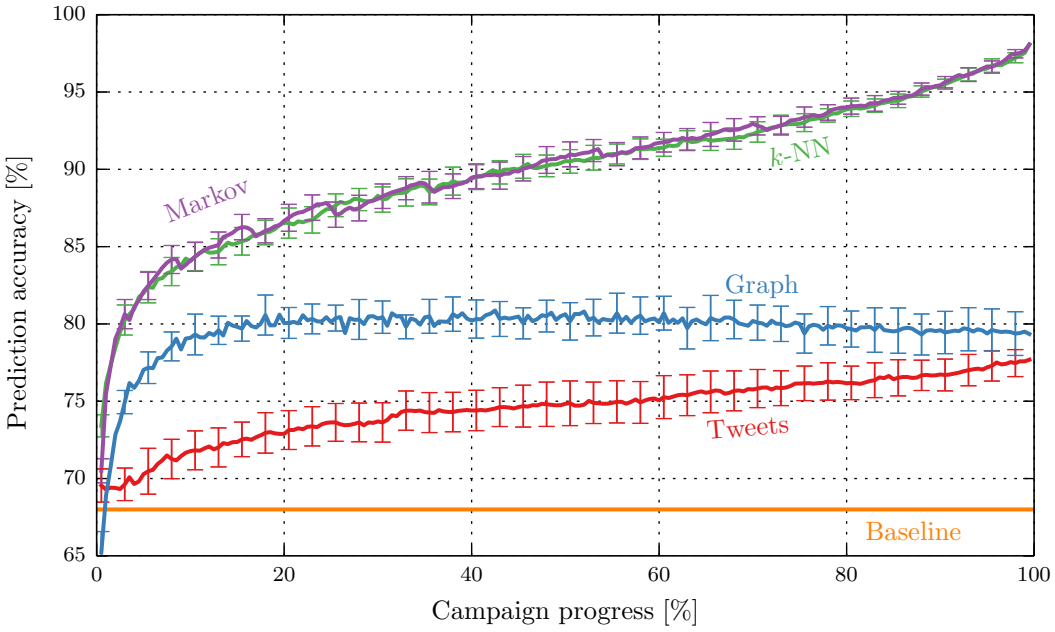


Figure 3.5 – Prediction accuracy of the models described in Section 3.3, along with the static baseline of Greenberg et al. [60]. For each state of progress of a campaign, we train a new model using 90 % of the campaigns and evaluate its performance on the remaining 10 %. We show the median accuracy over 10 random assignment, plus/minus one standard deviation.

The two money-based models perform similarly, and very well: after 15 % of the duration of a campaign, its current amount of money pledged enables us to predict its success with an accuracy higher than 85 %. As time goes by, this accuracy steadily increases, to reach more than 97 % in the very last moments. However, the k-NN model is very costly compared to the Markov model: it requires that all training samples be kept in memory and that the distance to each of them be computed when we want to classify a new sample. In contrast, the Markov model is compact, requiring only the matrices $\boldsymbol{P}(j)$ to be stored, and it computes the success probability of new samples very efficiently, because this requires only matrix multiplications. It is thus noticeable that such a lightweight and elegant model performs as well as a heavyweight method.

The two social models have different results. Although their performances are clearly inferior to those of the models that use the sequence of pledges, both social models quickly outperform the baseline performance of 68 % obtained by Greenberg et al. [60]. It is interesting to note that although the Graph model has a fast increase in accuracy after a few time steps, it quickly stabilizes to about 80 %, up to the end. The Tweets model, however, constantly improves its accuracy.

3.4 Combining Models

The models that use the sequence of pledges show a good prediction accuracy, especially towards the end of a campaign. At the beginning, however, the accuracy could still be improved. Such an improvement would be very useful to creators and backers, enabling them to react accordingly to correct the course of a campaign. A higher accuracy at later stages, however, would not be of high interest.

To improve the accuracy of the models presented in Section 3.3, we propose to combine the money-based models with those using social features. Indeed, the early results of money-based models seem to indicate that money alone is not sufficient to distinguish successful campaigns from those that will fail. Adding social features such as popularity on Twitter and links with other successful campaigns could help further refine the predictions.

3.4.1 Combined Model

We build a combined model by taking the predictions of all individual models and by training an SVM to combine them into a final prediction. The features used by this combined model are the campaign goal $g(c)$, its duration $d(c)$, along with the probabilities of success obtained by using each of the four individual models described in Section 3.3.

3.4.2 Hyperparameter Selection

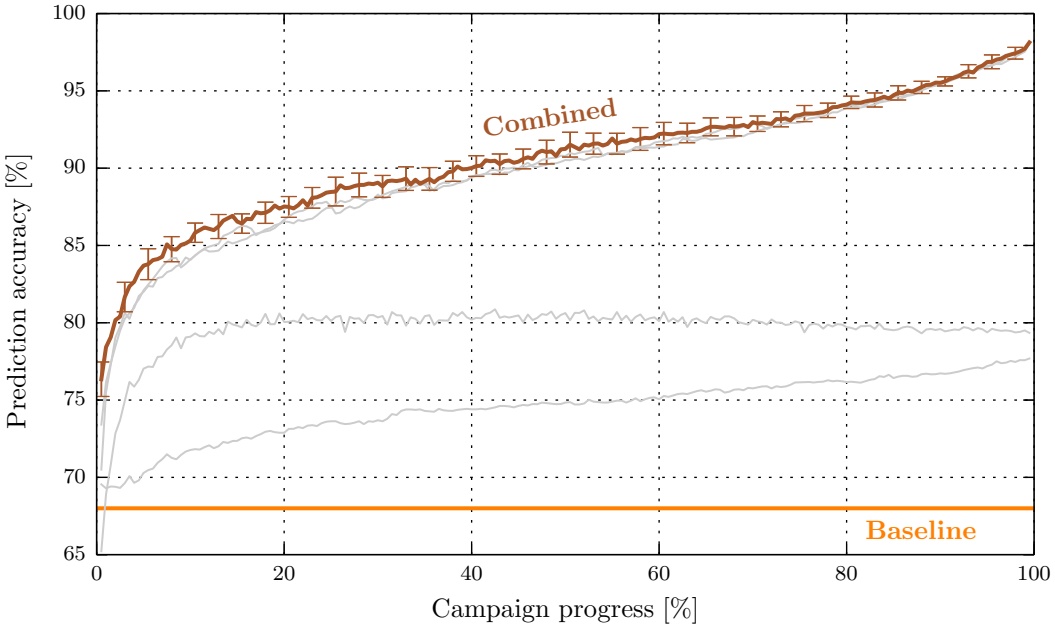
As with the social models described in Section 3.3, we use a RBF kernel for the SVM, thus having two hyperparameters C and γ to tune. We do an exhaustive search on a logarithmic scale for both hyperparameters and choose those that maximize the average prediction accuracy on the validation sets, for 10 random assignments. The hyperparameter values we select are $C = 100$ and $\gamma = 0.1$.

3.4.3 Results

Figure 3.6(a) shows the prediction accuracy of the combined model, along with the static baseline presented in Section 3.3.4. We highlight in Figure 3.6(b) the relative improvement of the combined model with respect to the best individual model, at each time step.

Overall, the improvement of the combined model is the strongest at the beginning of the campaign, increasing significantly the accuracy: the first combined prediction is 4 % more accurate than any individual model. In other words, on average 4 hours after the launch of a campaign, the combined model can assess the campaign's success with an accuracy higher than 76 %.

(a) Prediction accuracy of the combined model



(b) Relative improvement over the best individual model

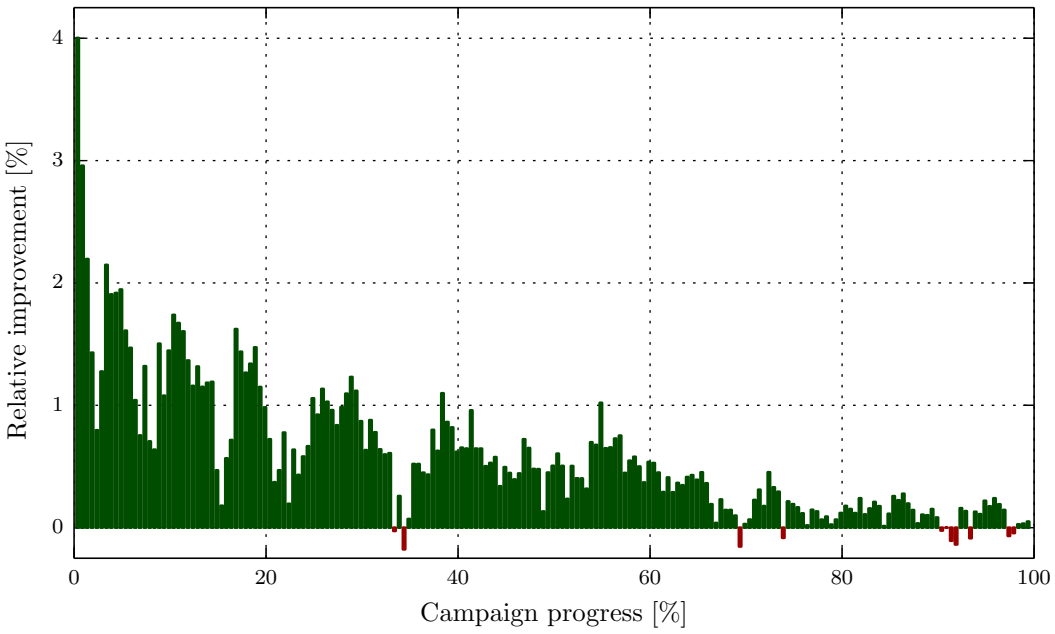


Figure 3.6 – Results of the combined model. For each state of progress of a campaign, we train a new model using 90 % of the campaigns and evaluate its performance on the remaining 10 %. We show (a) the median accuracy over 10 random splits (plus/minus one standard deviation) and (b) the improvement of the combined model relative to the best individual model. Combining social data with the amounts of pledged money is helpful for early predictions, with a relative improvement of 4 % for the first prediction.

3.5 Related Work

With the increasing popularity of crowdfunding platforms, people have are interested in studying the dynamics of crowdfunding campaigns and in understanding what drives their success or failure. Some online tools, such as *Kicktraq* [8], provide tracking tools for crowdfunding campaigns and basic trend estimators, but none has yet implemented proper success prediction.

In the scientific community, several studies have been published on crowdfunding platforms: Kuppuswamy and Bayus [73] study the dynamics of backers and the effect of responsibility among them. They highlight the effect of past support on future backers, as well as the the "last rush" effect that we observed, where some campaigns have a sudden increase in pledges close to their deadline. Similarly, Mollick [82] describes the dynamics of the success and failure of Kickstarter campaigns. He presents various statistics about the determinant features for success and analyzes the correlation of many campaign characteristics with its outcome.

Crosetto and Regner [39] take a similar approach, with data extracted from Startnext, the biggest crowdfunding platform in Germany. Wash [121] focuses on a different platform, called Donors Choose, where people can donate money to buy supplies for school projects. He describes how backers tend to give larger donations when the donation would enable a campaign to reach its goal, and he also studies the predictability of campaigns over time. Agrawal et al. [23] investigates the importance of geographical location and social links on the timing of pledges.

Greenberg et al. [60] propose a success model for Kickstarter campaigns, based solely on their *static* attributes, i.e., attributes available at the launch of a campaign. They obtain a prediction accuracy of 68 %, which we use as a baseline when presenting the results in Section 3.3. More recently, Rao et al. [95] study the time-series of money pledges on Kickstarter to investigate the extent to which the inflows and their derivatives are indicators of the success of a campaign. They reach accuracies that are similar to ours.

3.6 Summary

In this chapter, we introduce the problem of predicting the success of a crowdfunding campaign. We highlight the importance of combining information about both the evolution of a campaign (its sequence of pledges over time) and its social components (popularity on Twitter and identify of its backers). We show that even though models that rely solely on money give good predictions, taking social features into account improves predictions made in the early stages of a campaign, where they are the most helpful. The data presented in this chapter is available online [12] and the models are integrated into an online tool that we describe briefly in Appendix A.

4 Should They Be There?

> "Listen, Peaches, trickery is what humans are all about," said the voice of Maurice. "They are so keen on tricking one another all the time that they elect governments to do it for them."
>
> *Terry Pratchett*

Since the early 2000s, several governments, organizations, and academic groups set up *voting advice applications* (VAAs). VAAs are usually implemented as websites that enable politicians and interested citizens to express their preferences on political issues, by answering a series of pre-determined questions spanning a variety of topics. The candidates have public profiles containing their responses (as well as various other information, such as their birthdate, interests, or Facebook profile), and the voters are matched with candidates based on their own responses. This enables citizens to obtain for example voting recommendations for future elections or simply to investigate the difference of opinion between politicians and themselves.

At first limited to European countries, VAAs can now be found worldwide, some spanning multiple countries or even continents: *Vote Compass* [16] in Canada, the USA and Australia, *Vote Match* [17] in Europe, and *Preference Matcher* [11] in Europe and South America. Some countries also have their own local VAA, such as *StemWijzer* [15] in the Netherlands, *Wahl-O-Mat* [18] in Germany, and *Smartvote* [14] in Switzerland. These platforms are well received and used by a significant portion of the public: StemWijzer was used by 40 % of the Dutch electorate in 2006, and Wahl-O-Mat by 12 % of the German electorate in 2009 [52].

The initial reason for the existence of these VAAs was to increase political transparency and citizen participation. Yet, as a byproduct, they also provided researchers with new ways of mining and (re-)discovering patterns that are peculiar to political life, but that

usually require tedious manual analysis and knowledge of the domain. Although the nature of this data is not new, its scale is unprecedented.

4.1 Problem: Identifying (Changes in) Ideologies

In this chapter, we study the political opinions of politicians and citizens. We exploit the scale of the data extracted from a VAA to answer several fundamental questions from a statistical perspective. We first begin by describing the data in Section 4.2. Then, in Section 4.3, we identify the main themes that separate politicians the most, to understand if the traditional left/right and liberal/conservative axes are "efficient" ways of dividing the ideological space. To raise awareness about the potential misuse of VAAs, we show in Section 4.4 that it is possible to craft on a VAA a new profile that gets twice as many recommendations as the best real profile. Finally, we propose in Section 4.5 a way of detecting such misuse, by combining the votes cast by elected politicians with their VAA responses to detect significant changes in opinion between their pre-electoral profile and their voting behavior once elected.

4.2 Dataset

In this chapter, we focus on Switzerland, as it has a very rich political landscape (seven major political parties compete for seats in the parliament), easily available data (all votes cast in the parliament are recorded and downloadable for free), and an established voting advice application (Smartvote has been active since 2003).

4.2.1 Background: Politics of Switzerland

Before introducing the two datasets we use in this chapter, we give a brief overview of the Swiss political system. It consists of a *Federal Council* (7 seats) and a bicameral parliament, which is composed of the *Council of States* (46 seats) and the *National Council* (200 seats). The Federal Council serves as head of state and executive power, and the parliament possesses the legislative power (together with citizens, as per the constitutional right for citizens to launch initiatives[1]). The Council of States represents the *cantons* (the states of the federal state), and each canton is attributed two seats (except six "half" cantons that have only one seat). The National Council represents the people, and each canton is attributed a number of seats proportional to its population.

The National Council and the Council of States are elected at the same time every four years. Several political parties are represented in the parliament. In this chapter, we focus on the seven largest parties (in terms of votes obtained during the National Council elections in 2011) shown in Table 4.1.

[1]Initiatives, similar to propositions in California, allow any citizen or organization to gather a predetermined number of signatures to propose a new piece of legislation [125].

	Full name	Ideology	Votes
SVP	Swiss People's Party	National conservatism	26.6 %
SP	Social Democratic Party	Social democracy	18.7 %
FDP	Free Democratic Party	Classical liberalism	15.1 %
CVP	Christian Democratic People's Party	Christian democracy	12.3 %
Greens	Green Party	Green politics	8.4 %
BDP	Conservative Democratic Party	Conservatism, economic liberalism	5.4 %
GL	Green Liberal Party	Green liberalism	5.4 %

Table 4.1 – The seven major political parties in Switzerland after the National Council elections of 2011. The last column lists the percentage of votes each party obtained during these elections.

4.2.2 Opinions Expressed on the Smartvote VAA

Our first dataset consists of the responses given on the Smartvote VAA [14] by the citizens and candidates during the Swiss parliamentary elections of 2011[2]. Smartvote proposes a long and a short survey. The short survey is composed of 32 questions and the long survey is composed of 75 questions, which include the 32 questions from the short survey. The *voters* (meaning here the visitors of the website) have the freedom to choose which survey to answer, but the candidates have to answer all the questions of the long survey. The questions address various topics ranging from society to economy and finance, and they were carefully selected to cover topics as representative as possible of current political issues. Answering consists in selecting one of the following options: *strongly agree - agree - disagree - strongly disagree*. An additional set of "budget questions" requires selecting one of the options: *less - no change - more*. Finally, the voters can also select "no answer" (an option not available to the candidates). Each possible answer is mapped internally by Smartvote to a number in the set $\{0, 0.25, 0.75, 1\}$ for regular questions, and in the set $\{0, 0.5, 1\}$ for budget questions. The final recommendation given to each voter is a list of candidates, in decreasing order of distance (using the l_2-norm) to this voter [113].

2990 candidates filled out the survey, which represents about 82 % of all the candidates. Unless otherwise specified, we consider the responses given by voters who participated in the short survey (which was the most popular survey). This amounts to about 229 000 voters[3], which corresponds to 9.3 % of the total voter turnout of 2011. Detailed statistics about this dataset are summarized in Table 4.2.

[2]The Smartvote dataset can be obtained on demand for research purposes, by sending a request to `contact@smartvote.ch`.

[3]Obtaining a precise figure for the number of unique voters is difficult, as one voter can ask for several recommendations on the website. This number is an estimate, obtained by Smartvote after filtering out identical web sessions.

Number of questions in the short survey	32
Number of questions in the long survey	75
Number of candidates who answered the long survey	2 990
Approximate number of unique voters that requested recommendations	436 726
Number of voters who completed all questions of the short survey	229 133
Number of voters who completed all questions of the long survey	80 067

Table 4.2 – Statistics about the political opinions dataset that contains the responses given on the Smartvote VAA by the citizens and candidates during the Swiss parliamentary elections of 2011.

4.2.3 Votes in the Parliament

Our second dataset consists of all the votes of the members of the National Council cast during the first half of the 49$^{\text{th}}$ legislature, between December 2011 and December 2013. There were 2,494 votes by the 200 national councilors during this period[4]. To compare the opinions given on Smartvote with the votes in the National Council, we discard the votes of the councilors that did not answer the Smartvote survey, hence our final dataset contains the votes (or abstentions) of 181 national councilors.

4.3 Ideological Space

In this section, we provide an analysis of the political landscape of Switzerland. We observe that simple dimensionality-reduction techniques can produce useful visual representations of political positions. We then analyze the difference of distribution and polarization between voters and candidates (before and after the elections) in such political spaces. Finally, we compute pairwise similarities between political parties, as measured by the opinions expressed by their members.

4.3.1 Dimensionality Reduction

To be able to compare candidates with voters, we consider only responses to the questions of the short survey. Each candidate can thus be represented as a point in a space of 32 dimensions. Because it is likely that some politicians tend to think similarly on several questions, we can expect that some of these dimensions are strongly correlated. For instance, it could be the case that two persons who answer similarly to the question "Should access to naturalization be made more difficult?" also answer similarly to the question "Are you in favor of legalizing the status of illegal immigrants?". Therefore, one of the first questions that we could ask concerns the intrinsic dimensionality of this dataset. In the following, we use a singular value decomposition (SVD) [58, 119] in order to compute the sets of questions that best capture these correlations.

[4]The data is publicly available via a dedicated web-service: `http://ws.parlament.ch/votes`.

We merge budget and regular questions and denote by $\mathcal{A} = \{0, 0.25, 0.5, 0.75, 1\}$ the set of possible responses to any question on Smartvote. Let Q be the number of questions and C the number of candidates. Using this notation, we define $\boldsymbol{C}$ as the $C \times Q$ matrix of candidates' responses, whose (i, j)th entry $c_{i,j} \in \mathcal{A}$ is the response of the ith candidate to the jth question. We start by centering $\boldsymbol{C}$ so that it has zero mean. We then compute the SVD of $\boldsymbol{C}$ as

$$\boldsymbol{C} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^T,$$

where $\boldsymbol{U}$ is the $C \times C$ matrix whose columns are the left-singular vectors of $\boldsymbol{C}$, $\boldsymbol{\Sigma}$ is a $C \times Q$ diagonal matrix, whose Q non-zero entries are given by the singular values of $\boldsymbol{C}$, and $\boldsymbol{W}$ is the $Q \times Q$ matrix whose columns are the right-singular vectors of $\boldsymbol{C}$. We adopt the usual convention, according to which the columns of $\boldsymbol{U}$ and $\boldsymbol{W}$, and the diagonal elements of $\boldsymbol{\Sigma}$ are ordered by decreasing amplitude of the corresponding singular values.

The projection of $\boldsymbol{C}$ onto the basis constituted by its singular vectors is given by $\boldsymbol{C'} = \boldsymbol{C}\boldsymbol{W}$. The matrix $\boldsymbol{C'}$ has a diagonal covariance matrix, i.e., all its dimensions are uncorrelated. Furthermore, if we denote by s_i the singular value associated with the ith singular vector, the variance of the data along the ith dimension of $\boldsymbol{C'}$ is proportional to s_i^2. It follows that, for any $k \leq Q$, the first k dimensions of $\boldsymbol{C'}$ are the k dimensions that capture most of the variance of the data.

We use this property in Figure 4.1 (top) to obtain a visual representation of the candidates on the plane, by showing the first two columns of $\boldsymbol{C'}$, i.e., the projection of $\boldsymbol{C}$ onto its first two singular vectors. In Figure 4.1 (bottom), we also show the representation of the same candidates using the *Smartmap* provided by Smartvote [90]. The Smartmap employs a similar dimensionality-reduction technique based on correspondence analysis [63], and it has been manually validated in order to obtain the correspondence with traditional left/right and liberal/conservative directions [105].

The relative positions of candidates and political parties are qualitatively similar in both cases, which confirms that our dimensionality-reduction approach is consistent with traditional ideological representations.

Interestingly, the singular value decomposition easily recovers the usual left/right and liberal/conservative divisions, by looking *only* at the responses (and not at the questions themselves). In Table 4.3, we show the two most important questions corresponding to each of the first four singular vectors of $\boldsymbol{C}$ (i.e., the two questions with the largest absolute weights for each of the first four columns of $\boldsymbol{W}$), along with their weight relative to the most important question of each axis. It very clearly appears that the first two axes refer, broadly speaking, to openness and integration of foreigners, and to economic liberalism. The third axis tends to be dominated by "societal" issues, such as drug consumption and adoption by same-sex couples, and the fourth axis is hard to interpret.

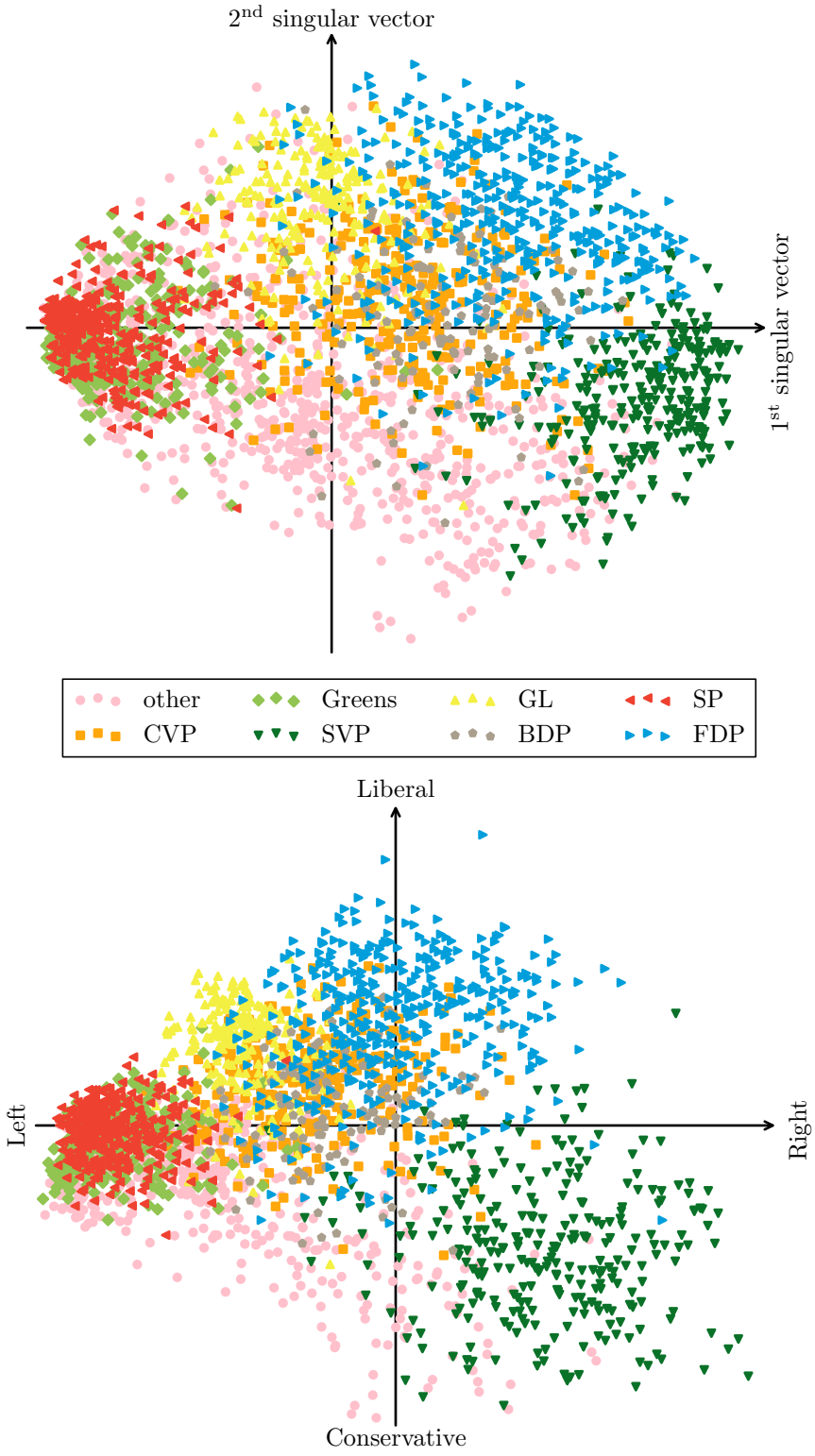


Figure 4.1 – Top: projection of candidates onto the first two singular vectors of the matrix of their Smartvote responses. Bottom: projection obtained by the Smartvote *Smartmap*, with qualitative axes referring to traditional ideological separations. The two projections are qualitatively very similar.

Singular vector	Weight	Most important questions dividing candidates
First	100 %	Would you support foreigners who have lived for at least ten years in Switzerland being given voting and electoral rights at municipal level throughout Switzerland?
	96 %	Are you in favour of legalizing the status of sans papiers immigrants (i.e., immigrants who have no official paperwork) through a one-off, collective granting of residency permits?
Second	−100 %	Are you in favour of the complete liberalization of shop opening times (i.e., shops would be able to choose their own opening times)?
	−93 %	Should Switzerland conclude an agricultural free trade agreement with the EU?
Third	−100 %	Should Switzerland legalize the consumption of hard and soft drugs as well as the possession of such drugs for personal consumption?
	−97 %	Should same-sex couples who have registered their partnership be able to adopt children?
Fourth	100 %	Should the acquisition of owner-occupied residential property be promoted through tax allowances for construction savings?
	93 %	Should the powers of the security services be increased to include "preventative" supervision of communication by post, e-mail and telephone?

Table 4.3 – Two most important questions of the first four singular vectors of the candidates' answers matrix $\boldsymbol{A}$. These questions are those that contribute the most, in absolute value, to each of the singular vectors. Their weight is shown relative to the most important question of each axis. They can be used to interpret the different themes on which the candidates tend to disagree the most.

We apply the same dimensionality reduction technique on the voters' responses matrix $\boldsymbol{V}$. Table 4.4 shows the two most important questions corresponding to each of the first four singular vectors of the voters' response matrix. The first two singular vectors capture themes that are similar to those of the candidates, whereas the fourth singular vector corresponds to the candidates' third. Interestingly, the question of liberalizing shop opening times is the most important for both the second and the third singular vectors of the voters' response matrix, suggesting that this issue divides many voters.

4.3.2 Candidates, Voters, and Polarization

Using the dimensionality-reduction approach presented above, we compare the distribution of candidates with that of voters in the ideological space. To this end, we divide the two-dimensional region of Figure 4.1 (top) into a 30×30 grid and compute the candidate density as the number of candidates in each cell. We follow the same procedure for voters and show both densities in Figure 4.2. Perhaps the most striking feature of this figure is

Singular vector	Weight	Most important questions dividing voters
First	100 %	Would you support foreigners who have lived for at least ten years in Switzerland being given voting and electoral rights at municipal level throughout Switzerland?
	94 %	Should popular initiatives be declared invalid if their entry into force would lead to infringement of the European Convention on Human Rights (ECHR)?
Second	100 %	Are you in favour of the complete liberalization of shop opening times (i.e., shops would be able to choose their own opening times)?
	−96 %	A recently launched popular initiative is proposing to limit the maximum salary that can be earned in a company to 12 times the lowest salary (the "1:12" initiative). Do you support this idea?
Third	100 %	Are you in favour of the complete liberalization of shop opening times (i.e., shops would be able to choose their own opening times)?
	98 %	Are cuts in federal taxes over the next four years something you believe should be a priority?
Fourth	100 %	Should Switzerland legalize the consumption of hard and soft drugs as well as the possession of such drugs for personal consumption?
	83 %	Should same-sex couples who have registered their partnership be able to adopt children?

Table 4.4 – Two most important questions of the first four singular vectors of the voters' answers matrix V. These questions are those that contribute the most, in absolute value, to each of the singular vectors. Their weight is shown relative to the most important question of each axis. They can be used to interpret the different themes on which the voters tend to disagree the most. The first two singular vectors are similar to those of the candidates.

the comparatively large density of candidates residing on the "left" of the political space. As it has already been observed [56], left-wing candidates appear to be very consistent in their responses and exhibit little variance. It seems to be that these candidates, more than the others, tend to strongly agree on the issues raised in the first two singular vectors. It is also possible that this is partly an artifact due to the (publicly admitted [19]) existence of "guidelines" provided by some parties and used by their candidates to answer Smartvote questions.

The difference between the two densities of Figure 4.2 also suggests that politicians are somewhat more polarized than citizens. This fact has often been observed by political scientists, in particular in Switzerland [79]. It is confirmed by the first two plots in Figure 4.3 that show the proportion of total variance that is captured by each of the first three singular vectors (as well as the remaining variance, captured by the remaining singular vectors). We see that the first three singular vectors capture 36 % of the variance in the voters responses, whereas candidates have 58 % of their variance captured by these first three dimensions.

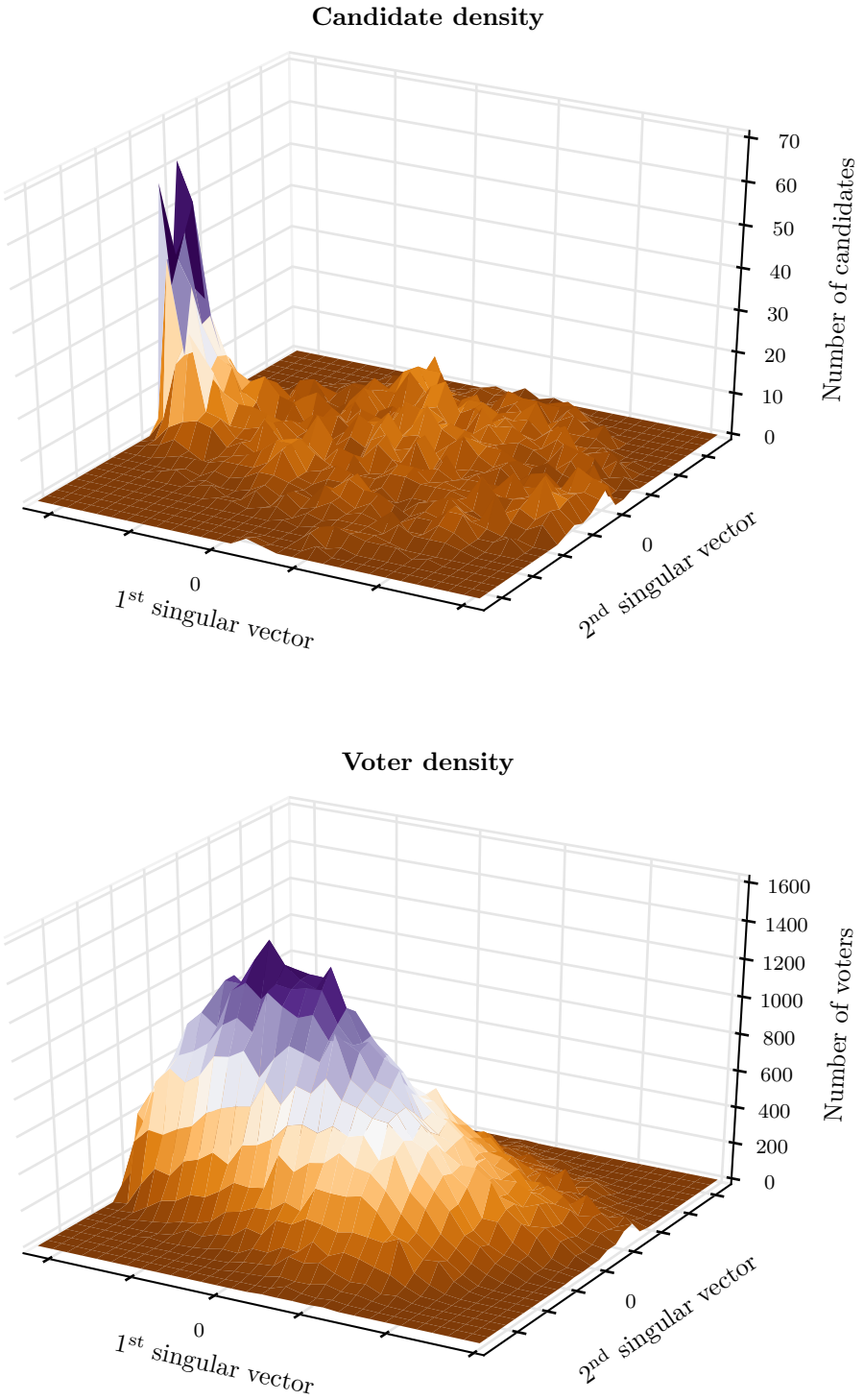


Figure 4.2 – Density of candidates (top) and voters (bottom) in their ideological spaces, computed from their Smartvote responses. The distributions are very different: voters are well spread, whereas candidates show a density peak on the "left" side of the space, where they have very low variance in their Smartvote responses.

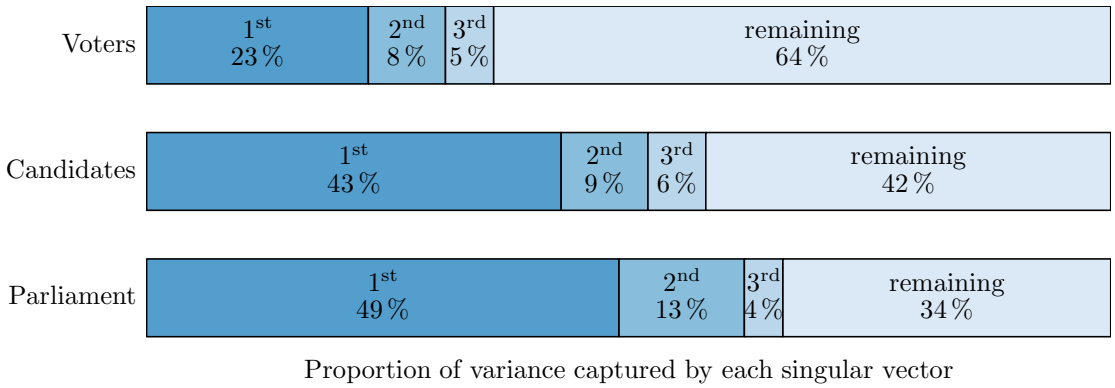


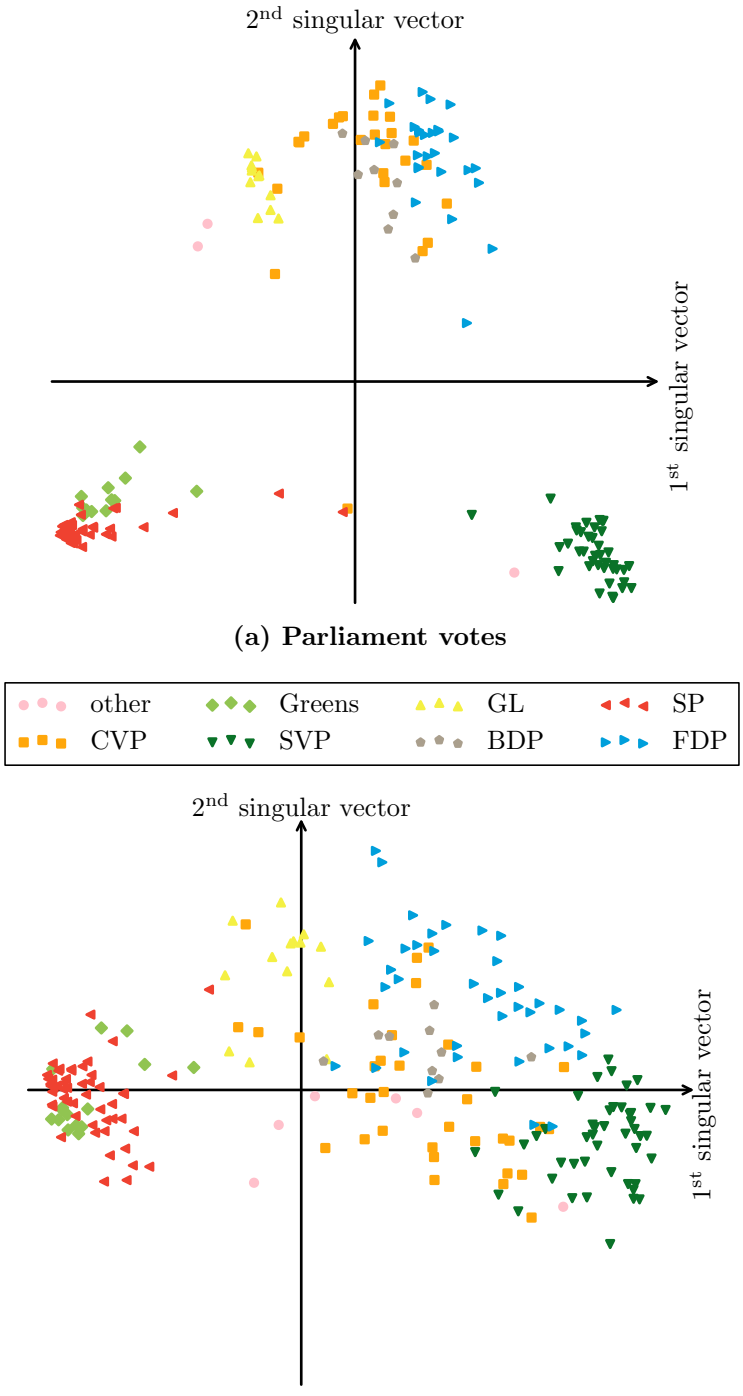
Figure 4.3 – Proportion of the variance captured by the singular vectors of the matrices corresponding to voters, candidates, and parliament members. The opinions given by the candidates are more polarized than the opinions given by the voters. Votes at the parliament are more polarized than the opinions given on Smartvote.

To further investigate the polarization of politicians, we apply the same dimensionality-reduction approach to their votes, once elected at the parliament. We consider the votes of the member of the Swiss parliament during the 49$^{\text{th}}$ legislature, described in Section 4.2.3, and we gather them in a matrix of parliament votes $\boldsymbol{P}$. We then compute its SVD and project $\boldsymbol{P}$ onto its first two singular vectors. The resulting two-dimensional representation of the members of the parliament is shown in Figure 4.4 (top). Unfortunately, the singular vectors of the matrix $\boldsymbol{P}$ are not easily interpretable, as they are composed of votes at the parliament that are often technical and very specific to the issue at hand.

Once elected, politicians are much more clustered compared to their pre-electoral opinion expressed on Smartvote (shown at the bottom[5] of Figure 4.4). This polarization of elected candidates can be explained by the existence of coalitions in the parliament. To highlight this effect, we show in Figure 4.5 a mapping between the two representations of elected candidates. We show their two-dimensional representation computed from their Smartvote responses on the left and from their votes in the parliament on the right. We see that a few candidates with similar opinions expressed on Smartvote vote quite differently once in the parliament.

We also show in the last plot of Figure 4.3 the variance captured by each singular vector of the matrix of parliament votes $\boldsymbol{P}$. It confirms that votes in the parliament are strongly polarized, with 66 % of the total variance explained by only the first three axes. The candidates, in contrast, are somewhat less polarized during the pre-electoral campaign, but still significantly more than the voters.

[5]The representation show at the bottom of Figure 4.4 is the same as the one shown at the top of Figure 4.1, but only showing elected candidates instead of all of them.



(a) Parliament votes

(b) Smartvote responses

Figure 4.4 – Two-dimensional representations of elected candidates, obtained (a) from the dataset of their votes in the parliament and (b) from their Smartvote responses. The votes in the parliament are more clustered than the pre-electoral opinions given by candidates. The representation obtained from Smartvote responses (b) is the same as the one shown in Figure 4.1 (top), but showing only elected candidates instead of all of them.

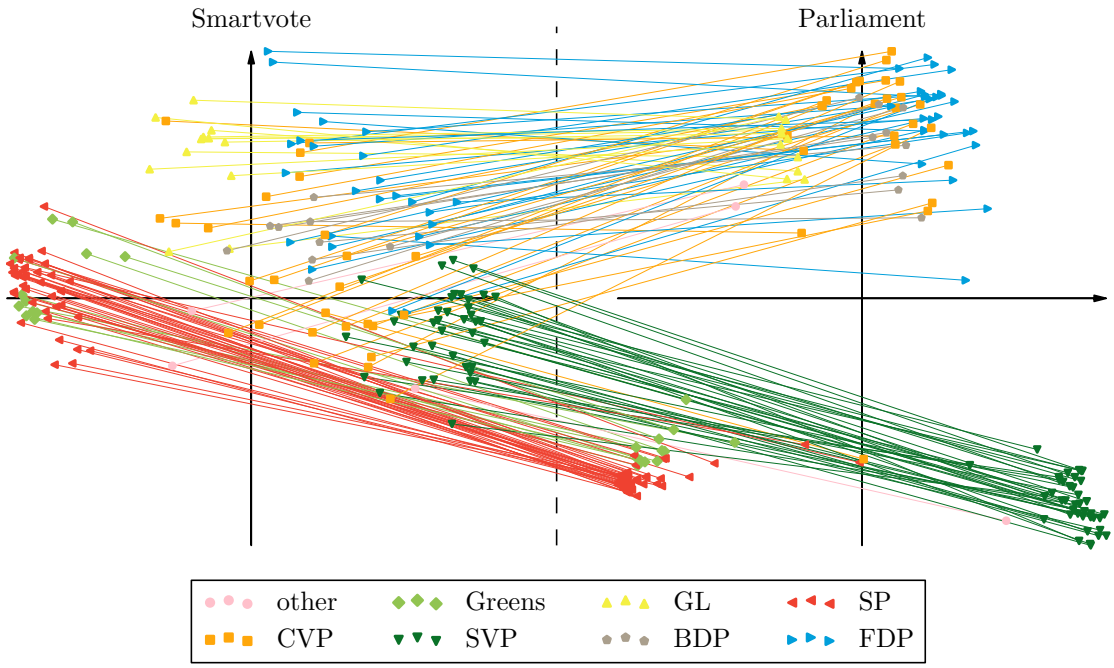


Figure 4.5 – Mapping between the two-dimensional representations of elected candidates computed from their Smartvote responses (left) and their votes in the parliament (right). A few candidates with similar opinions expressed on Smartvote vote quite differently once in the parliament.

4.3.3 Parties Overlap

Figure 4.1 shows that some subsets of the political parties significantly overlap with each other. To verify whether such overlaps still exist in the original 32-dimensional space, we compute, for each party, the proportion of candidates of this party closer to the median answer of the candidates of *at least* one other party than to the median of their own party. These proportions are shown in Figure 4.6. It appears that several of the main parties have a large proportion of their candidates closer to at least one other party. This concerns more than 20 % of the candidates of four of the seven parties. The FDP, CVP and BDP show exceptionally large figures; more than 35 % of FDP, 45 % of CVP and 50 % of BDP candidates are closer to the median answer of at least one other party. These parties do not belong to political extremities, rather they share a region near the center of the political space, which partly explains why they largely overlap. In practice, this means that using Smartvote questions, it is hard to determine which party best suits a person with centrist opinions.

In order to gain more insight into which parties are actually close to each other, we look at detailed pairwise overlaps. Specifically, for each pair of parties (i, j), we show in Figure 4.7 the proportion of candidates of party i closer to the median opinion of party j than to the median of their own party i. Again, these proportions are computed in the

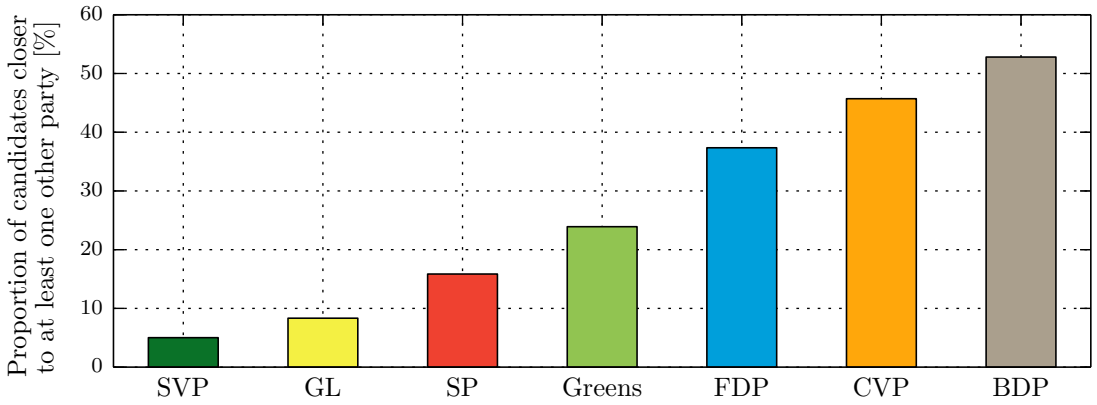


Figure 4.6 – Proportions of candidates of each party that are closer to the median of *at least* one other party than to the median of their own party.

	SVP	GL	SP	Greens	FDP	CVP	BDP
SVP	–	0.9 %	0.0 %	0.6 %	8.0 %	4.3 %	4.8 %
GL	1.3 %	–	0.7 %	3.7 %	10.2 %	14.1 %	7.2 %
SP	0.3 %	1.3 %	–	25.8 %	1.4 %	4.9 %	1.6 %
Greens	0.3 %	2.2 %	18.8 %	–	0.9 %	6.7 %	2.4 %
FDP	4.0 %	1.8 %	0.2 %	0.6 %	–	21.5 %	17.6 %
CVP	2.0 %	5.7 %	0.2 %	1.6 %	10.9 %	–	16.8 %
BDP	6.0 %	6.1 %	0.2 %	1.9 %	18.4 %	33.1 %	–

Figure 4.7 – Inter-party overlaps. The number in row i and column j indicates the percentage of candidates of party i that are closer to the median position of party j than to the median of their own party i.

original 32-dimensional space, and are thus not subject to distortion due to dimensionality reduction. It is surprising that even opposite parties (such as SVP and SP, or SVP and Greens) have a few overlapping candidates.

As an example of the interpretation of these overlaps, consider two parties: BDP and SVP. BDP was created in 2009, when SVP excluded one of its members who was part of the Federal Council. As a response, the cantonal branch of SVP to which this member belonged split from the national party and renamed itself to BDP. Over the following months, members of SVP from other cantons followed, moving from SVP to BDP. When looking at their overlap in Figure 4.7, we see that only 6 % of the members of BDP are closer to the median position of SVP than to that of BDP. Conversely, only 4.8 % of the members of SVP are closer to the median position of BDP than to that of SVP. These low overlaps thus suggest that the members of BDP indeed have diverging opinions from those of the members of SVP, which supports the reasons behind the division of the two parties.

4.4 Crafting a New Political Profile

As explained in Section 4.2.2, the Smartvote VAA gives voting recommendations to visitors by first computing the l_2-distance between their responses and those of each candidate, and then by recommending the candidates who gave responses closest to those of the visitors. This means that the responses candidates give to each question in the Smartvote survey influence directly the number of voting recommendations they get. Hence, it is interesting to see if it is possible to create an "optimized" profile, in order to obtain as many recommendations as possible.

Computing the optimal set of answers that maximize the likelihood of a candidate appearing at the top of recommendations would require knowing the answers of all the candidates and the voters. However, at the time of completing the survey, candidates can only access the answers given by their fellow candidates (which are publicly listed on the website), but not those of the voters. Furthermore, even if the set of answers given by voters were known in advance, the computation of an optimal profile is of combinatorial complexity; if there are n questions with k possible choices, an exhaustive search requires $O(k^n)$ computations. More efficient techniques (e.g., based on geometric approximation [62]) could be used to solve this problem. We leave a more formal study of this optimization problem for future work.

Instead, we propose a simple but efficient heuristic to craft a new candidate profile, by looking only at the answers of the other candidates. Our method consists in inspecting the distribution of candidates in the two-dimensional ideological space depicted in Figure 4.1. We see that there are several spots where the density of candidates is quite low. However, from Figure 4.2, we know that voters tend to have a more uniform distribution, thus

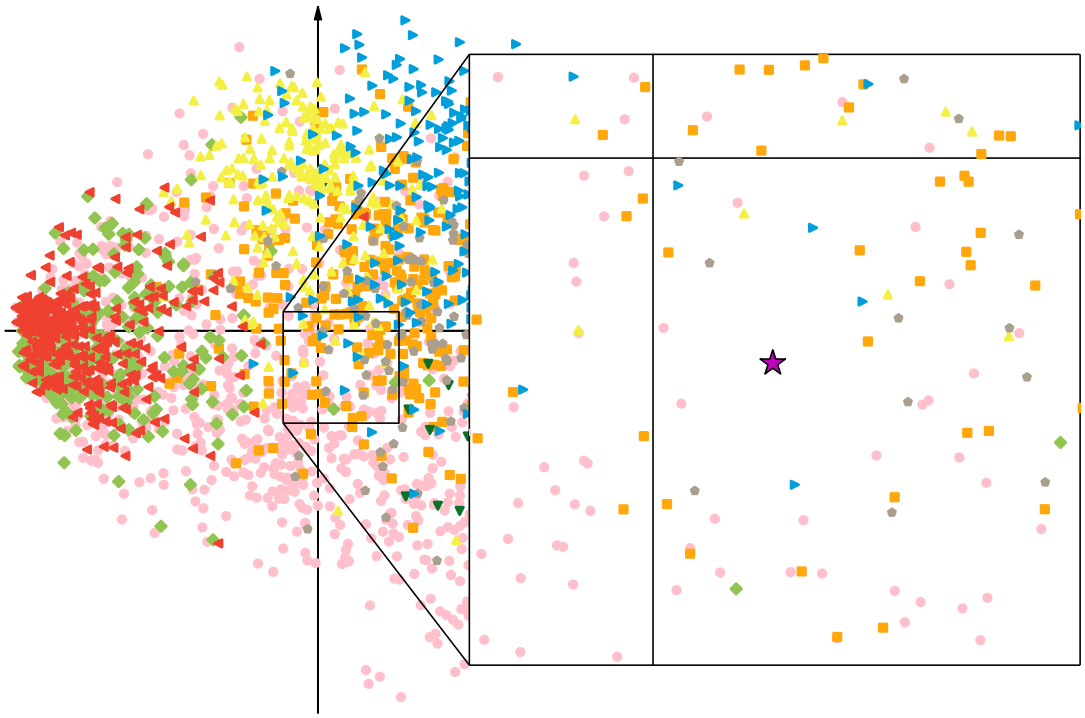


Figure 4.8 – Zoom into the two-dimensional representation of candidates shown in Figure 4.1. We clearly see an area with few candidates, which we choose for the location of our crafted opinion, represented as a star.

suggesting that these spots might correspond to "under-represented" citizens. Thus, we choose to place our crafted candidate in one of those spots, filling a gap in the ideological space but staying far from the extremes.

Such an "optimal" positioning problem has been studied from a game-theoretical point of view in simpler settings [40, 85] and researchers have shown that choosing the median position leads to obtaining the most voting recommendations. However, selecting the median answer to each question as our crafted profile does not give satisfactory results in our setting, as shown in Figure 4.9.

To compute the actual responses this crafted candidate should give to the Smartvote survey, we proceed as follows: First, we find the coordinates of an empty spot in the ideological space, represented in Figure 4.1, is still close to the center of the space. The intuition behind this choice is that we want to be as far away as possible from any other candidate, and still be close to the majority of voters. Such a location is illustrated in Figure 4.8. Then, we perform the inverse operation of the projection explained in Section 4.3.1, to project a two-dimensional point back onto the 32-dimensional space of Smartvote responses. Because the responses can only take values in $\mathcal{A}$, we round each component of the resulting projected answer to the closest value.

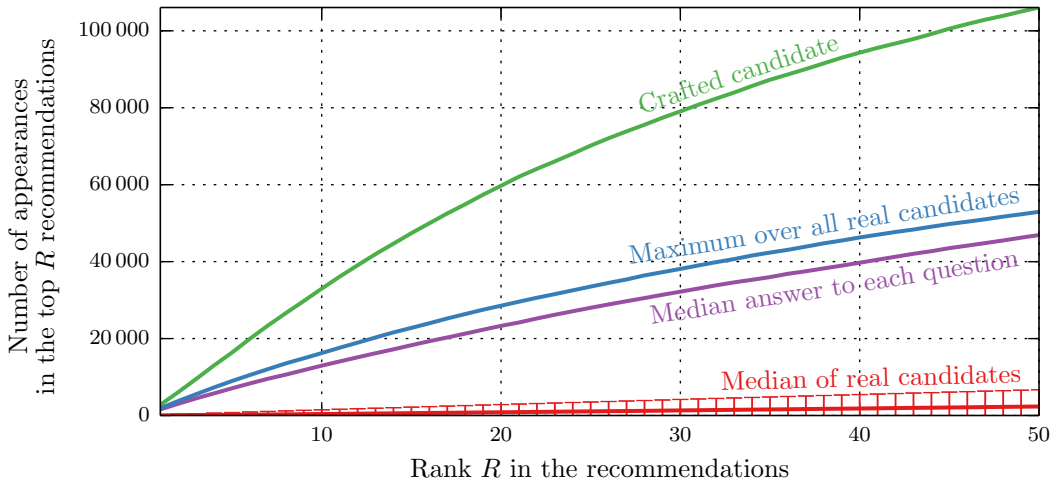


Figure 4.9 – Number of appearances of candidates in the top recommendations. The curves show how many times the median, best, and crafted candidates appear in the top R recommendations for voters. We also show that simply giving the median answer to each question does not beat the best real candidate. The crafted candidate uses responses corresponding to the star shown in Figure 4.8. It gets more recommendations than any other candidates, appearing in the top 50 recommendations for close to half of the 230 000 voters.

Finally, we add this crafted set of responses (obtained from the point shown in Figure 4.8) to the list of candidates and compute recommendations for each voter. We count, for all candidates, the number of times they appear in the top R recommendations of a voter, for $R \in \{1, \ldots, 50\}$ and show the results in Figure 4.9. The lower curve shows how many times the median candidate appears in the top R recommendations, and the error bars indicate the standard deviation. The second curve shows how many times a crafted profile with simply the median answer to each question appears in the top R recommendations. The third curve from the bottom shows the maximum number of times a real candidate appears in the top R recommendations. The upper curve shows how many times our crafted profile appears in the top R recommendations.

We see that our crafted profile appears significantly more often in the top recommendations than any other candidate. For example, it appears more than 100 000 times in the top 50 recommendations, about twice as much as the best real candidate. As our dataset consists of around 230 000 voters, this means that our crafted profile is recommended to almost half of the voters. Although the effect of these recommendations on direct votes has not been clearly determined [118], Ladner et al. [74] indicate that 67 % of Smartvote users state that Smartvote had an influence on their choice of party. This influence is even more significant for swing voters [75] and for younger or less informed voters [55]. Thus, both parties and individual candidates would benefit from an increased number of recommendations.

Instead of trying to optimize the profile of a single candidate, a party could be interested in jointly optimizing the answers of all of its members. This would allow the party to make sure that it covers as well as possible the region of the ideological space that corresponds to its political agenda. Indeed, if two of its candidates have similar profiles, they will compete with each other when the voting recommendations are emitted. Instead, they could try to distance themselves from each other and compete with representatives of other parties. We leave for future work the study of this more general global positioning problem.

4.5 Detecting Ideological Changes

We showed above how an unscrupulous candidate could create a profile that would gather more recommendations than any other. This could result in the election of this candidate, who would then have to vote daily in the parliament. However, in this case, the votes this candidate would cast in the parliament might not be in accordance with the opinion expressed by his crafted Smartvote responses. As all votes of the members of the parliament are publicly disclosed, a concerned citizen could monitor legislators in order to detect *flip-floppers*, i.e., candidates changing their opinion after they are elected. We propose here a method for measuring the shift in opinion of candidates, between the profile they advertised on Smartvote (or any other VAA) during an electoral campaign, and their voting patterns in the parliament once they are elected. Note that our method only quantifies opinion shifts. Of course, there are many contexts where politicians can be reasonably expected to change opinions with time, and moderate opinion shifts need not always to be interpreted as bad signals.

4.5.1 Predicting Parliament Votes from Smartvote Profiles

The first step towards detecting changes of opinion is to map a set of Smartvote responses to votes in the parliament. To do so, we identify parliament votes that can be predicted by Smartvote responses. Indeed, our intuition is that, as Smartvote responses are a good indicator of a candidate's political opinion, some votes can be accurately predicted from a set of Smartvote responses. Therefore, we define the following learning problem: Given the Smartvote profiles of all elected candidates, and their votes in the parliament on a given issue, learn a model that predicts the vote $v_c \in \{\text{yes}, \text{no}\}$ of a new candidate c on this issue from his Smartvote profile $\boldsymbol{c}_c$.

We train a linear classifier[6] for each of the 2494 votes in the parliament dataset. For each vote, we filter candidates to keep those that actually voted (some are sometimes

[6]We use logistic regression [43], implemented in Python with scikit-learn [89]. We also tried non-linear methods such as gradient-boosted decision trees and found no improvement. We thus use a linear model for simplicity and efficiency.

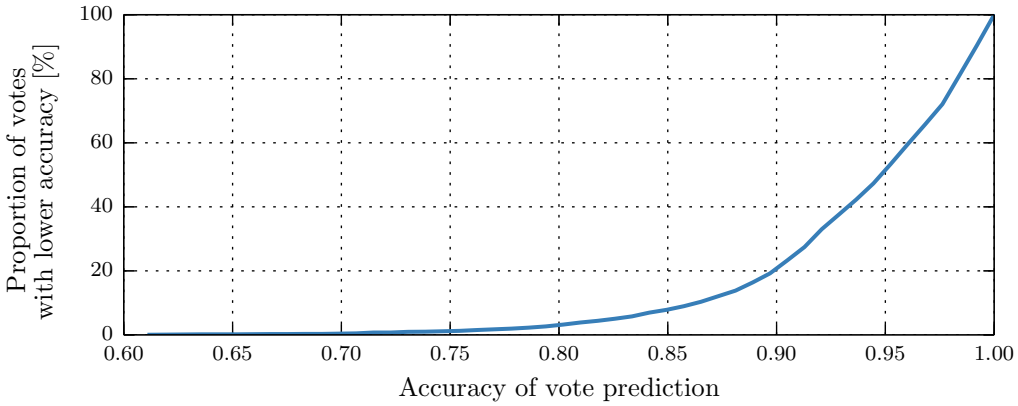


Figure 4.10 – Cumulative distribution function of the predictability of parliament votes from Smartvote responses. For each issue in the parliament dataset, we use the Smartvote profile of elected candidates to predict their votes, and report the average accuracy over 10 folds, where 90 % of the candidates are used for training and 10 % for evaluation. We see that close to 50 % of votes can be predicted with an accuracy higher than 95 %, using only the Smartvote profiles of legislators.

absent, or abstain) as learning samples. We evaluate the predictability of each vote by computing the prediction accuracy of our linear classifier on 10 folds, where, for each fold, the classifier is trained on 90 % of the candidates and evaluated on the remaining 10 %. We then compute the average accuracy on these 10 folds, and report the results in Figure 4.10.

Figure 4.10 shows the cumulative distribution function of the prediction accuracy for each vote, averaged over the 10 folds. We observe that the vast majority of votes in the parliament can be predicted with a high accuracy from Smartvote profiles; more than 90 % of votes can be predicted with an accuracy higher than 85 %, and close to 50 % of the votes can be predicted with an accuracy higher than 95 %.

4.5.2 Comparing Expected and Actual Votes

Now that we have a way to map Smartvote opinions to parliament votes, we can compute the expected votes of legislators, based on their Smartvote profile, and compare them with their actual votes. To do so, we first choose the 1000 most predictable votes, in order to maximize the confidence in our predicted votes. This corresponds to the top 40 % of votes, meaning that each of them can be predicted with an accuracy higher than 96 % (see Figure 4.10).

We then use a procedure, similar to the one described in Section 4.5.1, for predicting the expected votes of each candidate on these 1000 issues. For each candidate, we first train one linear classifier for each issue, using the Smartvote profile and vote on this

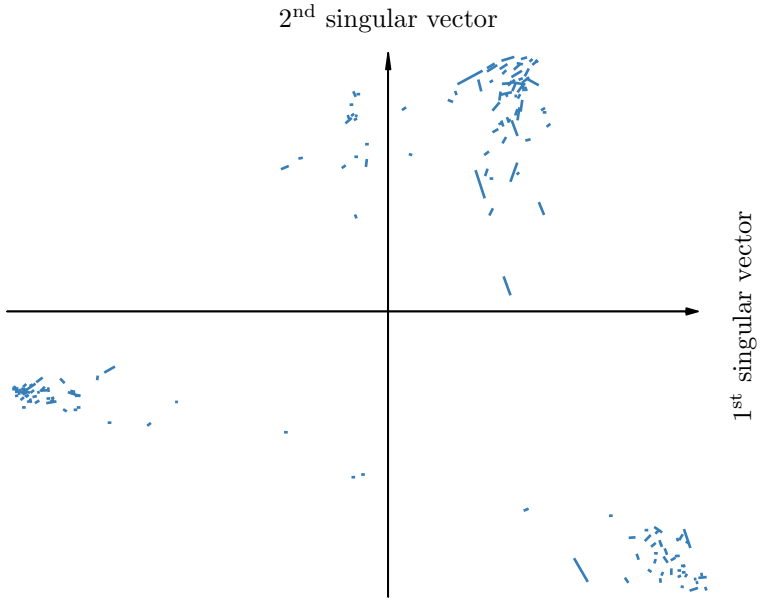


Figure 4.11 – Comparison of expected and actual votes of parliament members. Each segment represents a legislator, and goes from his expected votes (according to his Smartvote profile) to his actual votes. The median legislator has only 0.3 % of votes that are different than what can be predicted from his advertised opinion. The largest difference is 3.75 %. Interestingly, the magnitudes of the shifts seem to be different for the three coalitions.

issue of all other elected candidates. Then, we use each classifier and the candidate's Smartvote profile to compute his expected votes on these issues, and we compare them with his actual votes. To summarize the results, we compute the proportion of actual votes that differ from the expected votes. This proportion corresponds to the shift in opinion of the candidate, between his Smartvote profile and his actual voting behavior in the parliament.

The 181 legislators voted on a median number of 906 issues. The median discrepancy between the votes predicted from Smartvote profiles and the actual votes is only 0.3 %. This means that the median candidate votes coherently with his advertised Smartvote opinion 99.7 % of the time. The candidate with the largest discrepancy has 3.75 % of his votes in opposition to his advertised opinion; although this distance is an order of magnitude larger than the median distance, it still means that 96 votes out of 100 are coherent with what he advertised, which is a somewhat reassuring observation. A larger distance could mean that he falsely advertised his opinion on Smartvote, or that he "flip-flopped", i.e., he changed his opinion significantly after being elected. However, it can also be expected that legislators sometimes divert from their advertised positions, for example to follow their party on a specific issue. Thus, interpreting such differences between expected and actual votes should be done carefully.

To visualize these opinion shifts, we show in Figure 4.11 the two-dimensional representation[7] of the expected and actual votes of each councilor, computed as explained in Section 4.3.1. Each candidate is represented as a segment, with one end corresponding to his expected votes, and the other to his actual votes. The longer a segment, the more significant the shift in opinion between his Smartvote profile and his votes in the parliament. Interestingly, the magnitudes of the shifts seem to be different for the three coalitions.

4.6 Related Work

Spatial approaches are often used to represent politicians or parties, most often using one or two dimensions. Some studies use dimensionality-reduction techniques similar to ours [56, 112, 124, 123]. However, to the best of our knowledge, we are the first to apply it on datasets of this scale. Furthermore, we show how it can be used to design ideal VAA profiles, and put them in contrast with parliamentary votes.

Hansen et al. [61] explore the cohesiveness of political parties using VAA data, by measuring the agreement among party members. We propose a different approach that enables us to measure the overlap between each pair of parties.

Many researchers studied roll calls in the U.S. Congress [93, 92, 34]. For instance, Poole and Rosenthal [93] study voting patterns in Congress, and find that legislators can be described in a space of low dimensionality. Based on spatial voting theory, Enelow and Hinich [42] propose a method for predicting congressional votes. Their method relies only on past congressional votes to make predictions. Although we study the predictability of votes, we do not use our predictors to predict future votes. Instead, we propose a method that permits mapping one space (the opinions expressed on a VAA) onto another (the votes in the parliament), in order quantify opinion shifts.

Related to the votes prediction and the opinion shifts measurement that we propose in Section 4.5, Gerrish and Blei [57] study the prediction of lawmakers' position on a bill, using the text of the bill. The authors use the resulting model to explore how lawmakers deviate from their expected voting patterns. Finally, Poole [91] studies members of the U.S. Congress and finds that they "adopt a consistent ideological position and maintain it over time".

[7] We restrict the parliament dataset to the 1000 most predictable votes, instead of all 2494 votes, resulting in a projection slightly different than that shown in Figure 4.4.

4.7 Summary

In this chapter, we study the opinion of citizens and politicians on various societal topics, using Switzerland as our laboratory. We show that the data extracted from Smartvote, a Swiss VAA, corroborates the left/right and liberal/conservative simplifications of the ideological space that is often used: the sets of questions that correspond to these themes capture a large portion of the differences of opinion, both for citizens and politicians. We also notice that many political parties have a significant overlap with others, some having up to 40 % of their members closer to the median opinion of at least one other party.

To raise awareness about the potential misuse of VAAs, we describe how an unscrupulous candidate could create a synthetic VAA profile, in order to gather a very large number of voting recommendations. To hold a legislator accountable for his opinions expressed on a VAA, we propose to combine VAA responses with votes cast in the parliament in order to build a mapping between pre-electoral opinions and voting behaviors of elected candidates. By comparing expected votes with actual votes, our technique enables us to spot legislators that vote in contradiction to the opinions that they expressed on a VAA.

Overall, our work applies to any country where similar data is available, and it points to some avenues created by open government initiatives that enable new data-mining approaches to political and social science.

5 How Will They Vote?

> Ankh-Morpork had dallied with many forms of government and had ended up with that form of democracy known as One Man, One Vote. The Patrician was the Man; he had the Vote.
>
> *Terry Pratchett*

In order to promote transparency and accountability, as well as to stimulate citizen awareness, an increasing number of governments across the globe are adopting *open government* directives [76]. These result in the release of massive amounts of structured data about multiple aspects of state affairs, politics, and governmental agencies in various countries. As of 2015, the website Data Portals [4] references more than 420 such local, regional and national datasets.

Among these datasets, the detailed outcomes of issue votes are published in many countries. Such votes are direct expressions of the opinion of the people, on various issues such as education, economy, and even ethics. In some cases, the detailed results are released at a fine geographical level, along with the national outcome of these votes. This newly available data gives an unprecedented view into the political landscape of a country. It enables us to gain a deeper understanding of the different voting behaviors across regions, and to investigate what makes regions similar, or dissimilar.

Of course, political parties are very interested in this information. Being able to identify patterns in vote results, based on characteristics of the vote or the regions, would enable them to better focus their campaigning efforts. The media also spend much of their resources trying to predict the outcome of votes, both before and during the day of the vote. Knowing whether a vote will be a narrow or clear win, and being able to identify regions that are crucial in determining the national outcome, would enable media and polling agencies to better focus their attention on.

5.1 Problem: Vote Results Prediction

In this chapter, we study the voting behavior of small administrative regions in a country, and the relationship between the local and the national results of votes. Taking Switzerland as an example, we investigate, in Section 5.3, the voting patterns across municipalities and their evolution over time. Then, we address the problem of predicting the outcome of votes. We first show, in Section 5.4, that it is possible to identify individual regions that have a high predictive power of the binary national outcome of a vote. However, such a binary prediction is of limited use, especially because it does not tell how certain the result will be and it requires obtaining the result of a *specific* region.

To overcome this requirement, we propose, in Section 5.5, to take a collaborative-filtering approach and to model jointly the outcome of a vote across all regions, given side information about the vote and the regions. Such a model enables us to predict the outcome of a vote in all regions, based on a few observed results, and then to refine the predictions as more results are made available throughout the day. Moreover, by taking into account characteristics about the vote and the regions, the model is able to make accurate predictions with only a few observed results. These regional predictions can then be combined to predict the national outcome with high precision. We also show, in Section 5.5.7, that the models are easily interpretable, enabling us to investigate which regions are likely to vote the same and to identify the relationship between the characteristics of regions and the correlation between their results.

5.2 Dataset

As mentioned above, we consider the case of Switzerland, as it has a very active political system with easily available data. Swiss citizens vote on average eight times per year, on various issues regarding military, finances, transportation, culture, integration of foreigners, public health, etc. The results (i.e., the proportions of "yes") are publicly available [47] for each Swiss municipality[1]. In December 2014, there were 2352 municipalities in Switzerland. Our dataset consists of the outcomes of the federal (i.e., nationwide) issue votes in each municipality between January 1981 and December 2014. There were 281 such votes.

In addition to the vote results, we gather side information about both votes and municipalities. For each vote, political parties publish voting recommendations, such as "in favor", "against", or "no recommendation". We obtain, for each vote, the voting recommendation from the 13 main political parties in Switzerland[2]. For each municipality, we gather 25 features about its location, population, and electoral profile[3]. Table 5.1 lists the municipality features and their main statistics.

[1] Municipalities are the smallest administrative regions in Switzerland.
[2] The voting recommendation are available on request at the Swiss Federal Statistics Office [5].
[3] All the features describing Swiss municipalities are available online [46].

Feature	Unit	Min	Max	Mean	Median
x	meters	487 212.59	826 224.39	633 278.12	627 998.66
y	meters	76 505.50	294 280.03	200 725.71	206 342.33
Elevation	meters	196.00	1 960.00	607.49	521.00
Population	count	12.00	380 777.00	3 419.26	1 333.00
Population density	inhabitants/km^2	0.80	11 866.48	389.49	157.83
Age 0-19	%	0.00	38.24	21.63	21.60
Age 20-64	%	33.33	80.00	61.09	61.17
Age 65+	%	4.76	66.67	17.28	16.81
Social aid	%	0.00	11.45	1.71	1.27
Foreigners	%	0.00	60.76	14.84	12.55
Jobs	count	4.00	444 198.92	2 064.51	453.91
Election BDP	%	0.00	82.15	7.35	4.84
Election CVP	%	0.00	87.20	14.20	8.42
Election PEV	%	0.00	24.13	2.56	1.89
Election FDP	%	0.00	92.11	14.42	12.13
Election SP	%	0.00	55.33	16.46	16.11
Election PST	%	0.00	28.50	1.58	0.52
Election GL	%	0.00	18.13	5.35	4.81
Election SVP	%	0.00	100.00	30.45	30.08
Election Greens	%	0.00	32.10	7.10	6.27
Election other right	%	0.00	60.73	3.26	1.60
Speaks German	0 = No, 1 = Yes	0	1	0.64	1
Speaks French	0 = No, 1 = Yes	0	1	0.30	0
Speaks Italian	0 = No, 1 = Yes	0	1	0.06	0
Speaks Romansh	0 = No, 1 = Yes	0	1	0.03	0

Table 5.1 – Summary of the 25 features describing each Swiss municipality. The x and y coordinates are defined in the Swiss coordinates system [44]. Election features show the proportion of votes for each party during the national elections of 2011 (see Table 4.1 for more information about the main Swiss political parties). The language features show which languages are spoken in each region (some regions are multilingual).

5.2.1 Preprocessing

Administrative regions change over time. It is common to have fusions and divisions of municipalities in Switzerland, and the total number of municipalities has been reduced by more than 10 % since 2000 [45]. This means that some of the current municipalities did not exist at some point in the past, and thus have no explicit results for some past votes. To make sure that all regions have a result for all votes, we could simply discard all municipalities that did not exist at some point in time during the whole 34 years that our dataset spans. However, this would result in about 7 % of discarded regions. To have as much data as possible, and to be able to make predictions about all regions that exist today, we chose to interpolate missing results instead.

To do so, we begin by constructing the history of each region. This history contains, for each region d, the set of regions d' that are related to it, either by fusion (d' merged with other regions and became d) or division (d' was divided into smaller regions, one of which being d). We use the list of all fusions and divisions, available from the Swiss federal statistical office as a list of pairs ($d_{\text{old}} \Rightarrow d_{\text{new}}$). The procedure for constructing the regions' history is described in Algorithm 2.

Algorithm 2: Construction of the history of all regions

Input: List of fusions and divisions of regions, as pairs ($d_{\text{old}} \Rightarrow d_{\text{new}}$)
Output: History of each region, as a mapping $d_{\text{new}} \Rightarrow \left\{d_{\text{old}}^{(1)}, d_{\text{old}}^{(2)}, \dots\right\}$

for *each region* d **do**
  history(d) = {}
for *each change* ($d_{old} \Rightarrow d_{new}$) *in chronological order* **do**
  history(d_{new}) = history(d_{new}) $\cup$ $\{d_{\text{old}}\}$ $\cup$ history(d_{old})

Using the history of all regions, we can now interpolate the missing results. We define regions existing today as the set of current regions $\mathcal{R}$. We want to have, for each region $d \in \mathcal{R}$, a result for each vote in our dataset, interpolated if needed.

For each vote n and region d, we use the true result of d if d existed at the time of n. If not, we interpolate its result as follows: Using the history of d, we select the set of regions $\mathcal{R}_n(d) = \{d' \in \text{history}(d) : d' \text{ existed at the time of } n\}$ that are related to d and have a result for n. We define the interpolated result of d as the average of the results of regions in $\mathcal{R}_n(d)$, weighted by the number of ballot papers that each region collected. This procedure is summarized in Algorithm 3.

Algorithm 3: Interpolation of the missing results of a vote

Input: Current regions $\mathcal{R}$, history of all regions, outcome and ballot of vote n for regions existing at the time of n
Output: Interpolated result of vote n in current regions that did not exist at the time of n

for *each current region* $d \in \mathcal{R}$ **do**
  if d *did not exist at the time of* n **then**
    $\text{result}_n(d) = 0$
    sum = 0
    for *each region* $d' \in$ *history*(d) **do**
      if d' *existed at the time of* n **then**
        $\text{result}_n(d) = \text{result}_n(d) + \text{ballot}_n(d') \cdot \text{result}_n(d')$
        sum = sum + $\text{ballot}_n(d')$
    $\text{result}_n(d) = \text{result}_n(d)$ / sum

We also standardize the features of votes and regions, such that they have zero mean and unit variance.

5.3 Voting Patterns

We begin by taking a closer look at the voting patterns across the regions, and their evolution over time. We consider the $D \times N$ matrix $\boldsymbol{Y} = (y_{dn})_{1 \leq d \leq D, 1 \leq n \leq N}$, where $D = 2352$, $N = 281$ and $y_{dn} \in [0, 1]$ is the outcome (proportion of "yes") of vote n in region d.

5.3.1 The Infamous "Röstigraben"

To identify the main voting patterns across regions, we use principal components analysis [103, 106], similarly to our approach in Section 4.3. We center $\boldsymbol{Y}$, compute its singular value decomposition, and project $\boldsymbol{Y}$ onto its first two singular vectors to obtain $\boldsymbol{Y'}$, a two-dimensional representation of the voting behavior of each region. We show the resulting projection in Figure 5.1(a). In this figure, each municipality is represented by a point whose shape and color indicates the language spoken by the majority.

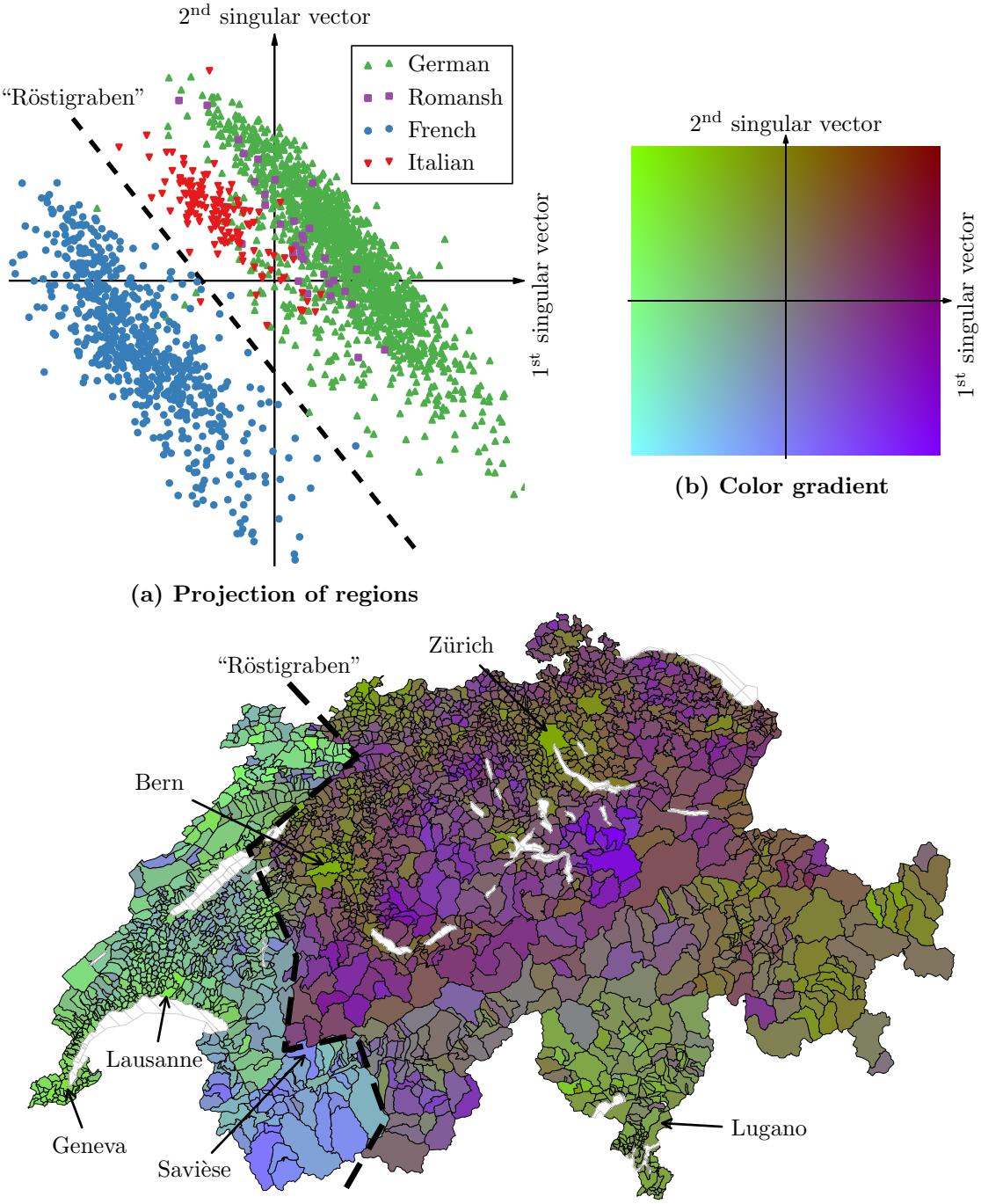
Figure 5.1(a) shows two clear clusters, corresponding to the French-speaking regions on one side, and the remaining regions on the other, separated by what Swiss people humorously call the *Röstigraben*[4]. The gap between the two clusters reflects the difference in opinion that often arises during federal votes in Switzerland, where the results of French-speaking regions are significantly different than those of German-speaking regions. Although the Italian-speaking regions are culturally closer to the French-speaking ones (and are usually placed on the same side of the Röstigraben), in this projection their voting patterns seem to be globally closer to those of German-speaking regions.

5.3.2 Geographical Patterns

To investigate the relationship between the geographical location of a municipality and its voting behavior, we map each point of the two-dimensional space represented in Figure 5.1(a) to a color, illustrated by the gradient in Figure 5.1(b). We then draw the map of Switzerland in Figure 5.1(c), where each region is shown with the color corresponding to its location in Figure 5.1(a). Thus, two regions with similar voting behaviors have a similar color on the map[5]. Lakes are shown in dashed gray.

[4] Literally *Rösti ditch*, also called the *hashbrown curtain*. This term describes the cultural difference between the German-speaking Switzerland, on one side, and the French-speaking part (sometimes together with the Italian-speaking part) on the other.

[5] An interactive version of this map is available online, on the platform described in Appendix B.



(a) Projection of regions

(b) Color gradient

(c) Geographical voting patterns

Figure 5.1 – Voting patterns of Swiss municipalities. We show in (a) the projection of the results of each region onto the first two singular vectors of the vote results matrix Y. The shape and color of each point indicates the language spoken by the majority. To visualize the geographical distribution of voting behaviors in (c), we color each municipality based on its location in (a), using the color gradient shown in (b). The *Röstigraben*, corresponding to the cultural difference between French-speaking and German-speaking municipalities, is clearly visible from the difference in voting patterns.

Again, the separation between the French and German-speaking parts is clearly visible. Moreover, it is possible to identify different types of regions: Urban centers, such as the greater areas of Geneva, Lausanne, Bern, and Zürich have relatively similar tints of green, indicating that they share similar voting patterns, whereas rural areas in the German-speaking part share a deep purple color. The French-speaking part of the mountainous canton of *Valais*, located in the southwestern part of Switzerland, has its own unique voting pattern, shown in light blue.

5.3.3 Changes over Time

The voting patterns illustrated in Figure 5.1 are the summary of 34 years of votes. However, voting behaviors change over such long periods of time. The population of municipalities grows, people move, and the societal attitudes on various subjects evolve. To visualize the variations in the voting behavior of regions over time, we take subsets of votes, and perform the same dimensionality reduction as described above.

For each window of 75 contiguous votes, we project the result of each municipality onto the first two singular vectors of the corresponding subset of the columns of $\boldsymbol{Y}$. We thus obtain a two-dimensional representation of regions, for each window of 75 votes. As we are more interested in the relative position of regions in the projected space than their absolute location, we use the projection of the first 75 votes as a reference point, and then we take the symmetry and rotation of the subsequent representations that minimize the total displacement of regions.

We show in Figure 5.2 four snapshots of the voting behaviors of regions. Each snapshot corresponds to 75 contiguous votes, spanning between 8 and 11 years and covering the whole duration of the dataset. Although the Italian-speaking regions have voting patterns that are globally similar to those of the German-speaking ones, as seen in Figure 5.1(a), they change their behavior between snapshots (b) and (c): their more recent votes are more similar to those of the French-speaking regions than the German-speaking ones.

Figure 5.2 shows snapshots of the voting patterns, during different time intervals. Instead, we can look at the "trajectory" of specific municipalities over all snapshots, to visualize how their relative behaviors changed over time. Figure 5.3 illustrates the evolution of the voting patterns of four Swiss municipalities: Bern (the German-speaking capital), Geneva (the largest French-speaking city), Lugano (the largest Italian-speaking city), and Savièse (a rural French-speaking municipality from the canton of Valais). Globally, Bern and Geneva have relatively stable voting patterns over time, with both their relative and absolute positions changing little. Savièse has significantly evolved in its voting behavior: It starts with an "extreme" position, corresponding to the more rural regions with low population, and slowly moves to the center of the space, as its population grows over

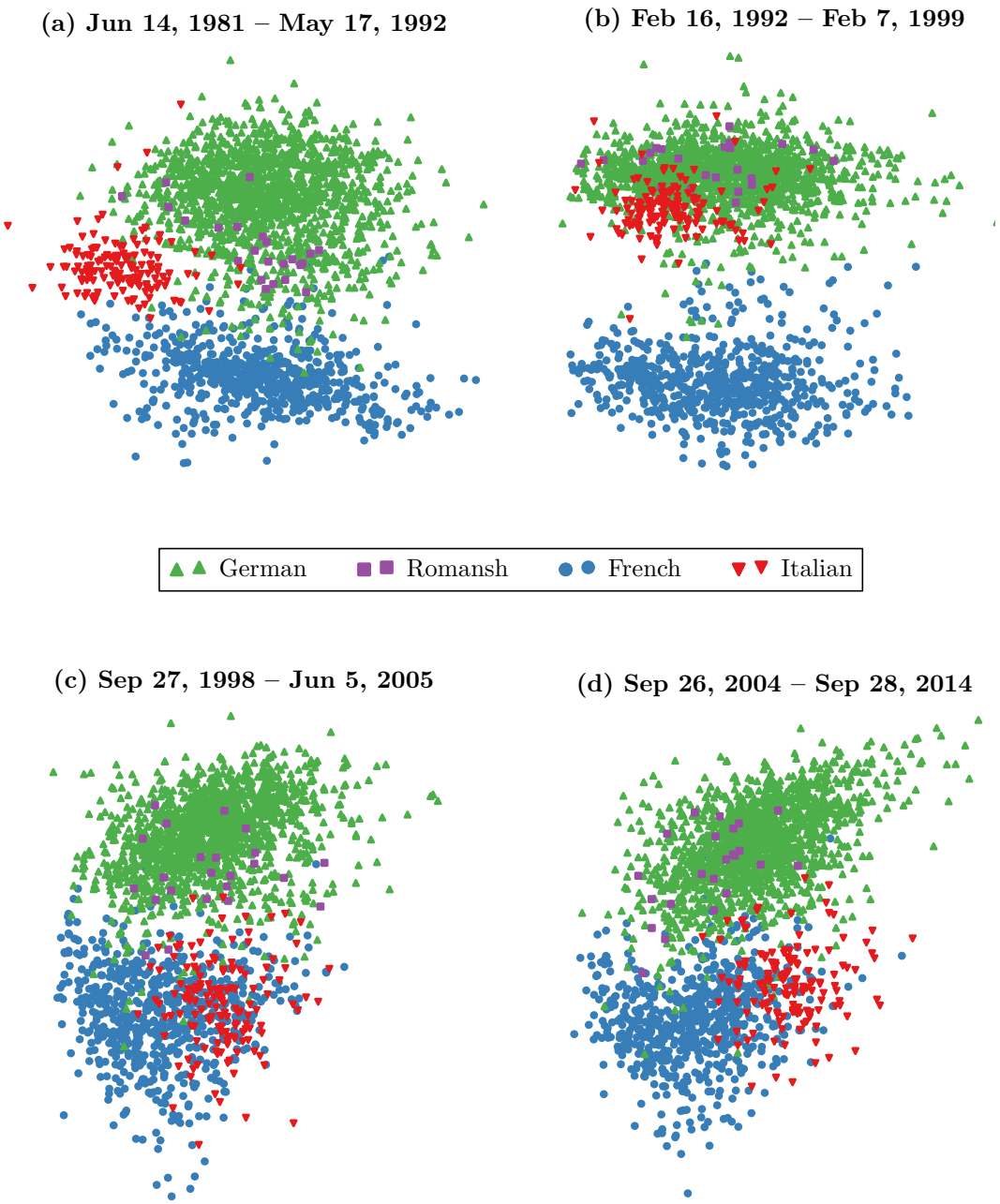


Figure 5.2 – Evolution of the voting behavior of regions over time. Each snapshot takes 75 votes into account and shows the projection of their results onto the first two singular vectors of the corresponding subset of the columns of Y. Between snapshots (b) and (c), the Italian-speaking regions change their voting behavior, from being closer to the German-speaking regions to being closer to the French-speaking ones.

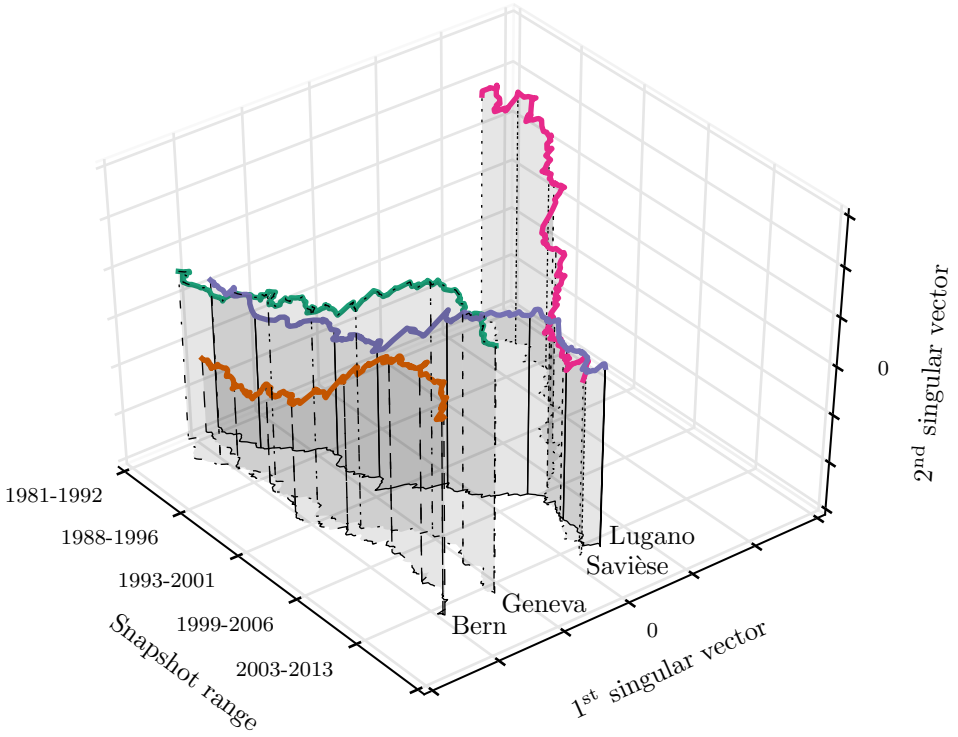


Figure 5.3 – Evolution of the voting behavior of four Swiss municipalities, over snapshots of 75 contiguous votes. Bern and Geneva, two of the largest cities, globally maintain their (relatively similar) voting behaviors. Lugano, one of the main Italian-speaking cities, distances itself over time from Geneva and increasingly ressembles Savièse, a rural French-speaking municipality from the canton of Valais.

time[6]. Its final voting behavior is similar to that of Lugano, which starts close to Geneva but slowly distances itself. Figure 5.3 is a simple example, but it highlights the powerful exploration and interpretation tool that such a visualization provides.

5.4 Predictions from a Single Municipality

We have seen in Section 5.3 that regions vary substantially in their voting patterns. One question that arises from this observation is whether it is possible to find one region whose voting behavior is representative of the global national outcomes. To answer this question, we study in this section the predictability of the binary outcome of votes at the federal level, using the outcome in a single municipality as unique feature.

We therefore define the following learning problem: Given the outcome $y_{dn} \in [0, 1]$ of the nth vote in the dth region, can we predict its outcome $o_n \in \{\text{yes}, \text{no}\}$ at the federal level?

[6]The population of Savièse nearly doubled between 1981 and 2014, with about 2700 registered voters in 1981 and 5000 in 2014.

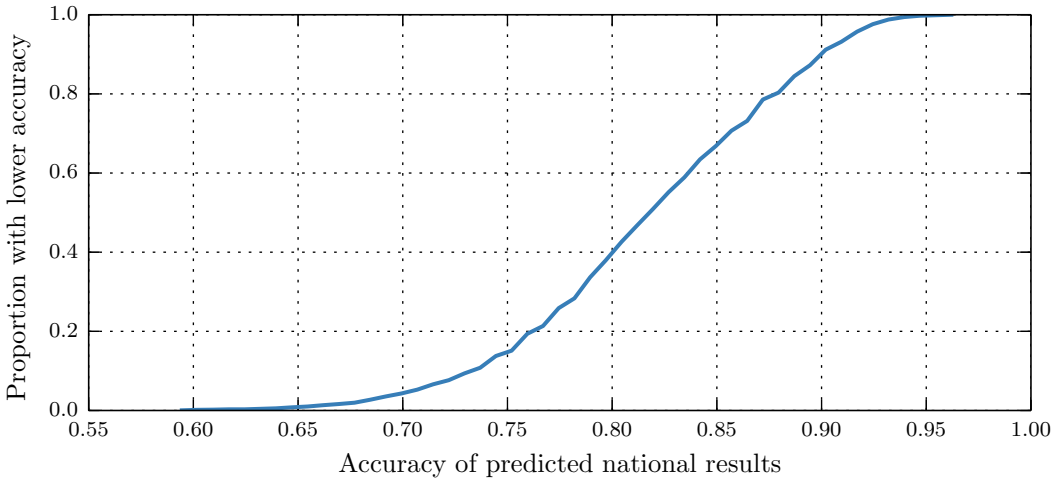


Figure 5.4 – Cumulative distribution function of the accuracy of the prediction of the outcome of votes at the federal level, given the outcome of a single municipality. The accuracies are averaged over 10 cross-validation folds. 10 % of municipalities allow to predict vote results at the national level with an accuracy higher than 90 %.

We split our dataset of 281 votes by taking the first 80 % (224 votes) as a training set, and the remaining 20 % (57 votes) as a test set. We train one binary classifier[7] for each municipality $d \in \{1, \ldots, D\}$. The parameters of the classifiers are selected using 10-fold cross-validation on the training set.

Figure 5.4 shows the cumulative distribution function of the accuracy of these D classifiers, averaged over the 10 validation sets (i.e., over the 10 cross-validation folds). About 10 % of regions correspond to an accuracy higher than 90 %, which means that knowing their result enables us to predict the binary outcome at the national level with less than 10 % of mistakes. Moreover, some municipalities reach accuracies of more than 96 % on the validation set. The municipality reaching the highest average prediction accuracy on the validation sets is *Rüegsau*, a village of 3000 inhabitants in the canton of Bern. The classifier which uses the vote outcome in Rüegsau as a feature to predict the national binary outcome obtains a prediction accuracy of 93 % on the test set. This means that out of the 57 votes of our test set, only 4 are incorrectly predicted by the classifier of Rüegsau.

Having such a representative sample could be extremely useful to many: Polling institutes, political parties and even news agencies would be able to target this municipality instead of sampling the population at random, thus maximizing the utility of their opinion surveys.

[7] We use a gradient-boosted decision tree [49, 50], implemented in Python using scikit-learn [89]. GBDTs are good candidates for this problem, as they naturally capture the thresholding that happens when a proportion is converted to a binary outcome. Moreover, they give better results than simpler methods such as logistic regression.

5.5 Collaborative Prediction of Vote Results

Instead of predicting the binary national outcome of a vote using the result of one *specific* municipality, a more useful model would enable us to jointly predict the outcome in all regions for a given vote, having observed the result of *any* subset of the regions. By using such a model, we would be able to predict the outcome of future votes in all regions at once, and to take into account local results as they are released on the day of the vote in order to refine these predictions. Moreover, the national outcome of a vote simply being the aggregation of local results, these predictions could also be used to predict the national result.

We observed, in Figure 5.1, that many municipalities have similar voting patterns. Some of these similarities can be partially explained by the characteristics of the municipalities, such as their spacial proximity, their demographic attributes, and their political orientations. In addition to similarities between regions, we also observe that some votes share similar patterns of regional results. We show in Figure 5.5 two examples of such groups of votes, in which (a) the French-speaking regions (and sometimes the Italian-speaking regions) vote in opposition to the rest of the country, and (b) densely-populated regions and rural regions vote differently. Other patterns, such as unanimous results, are also very common.

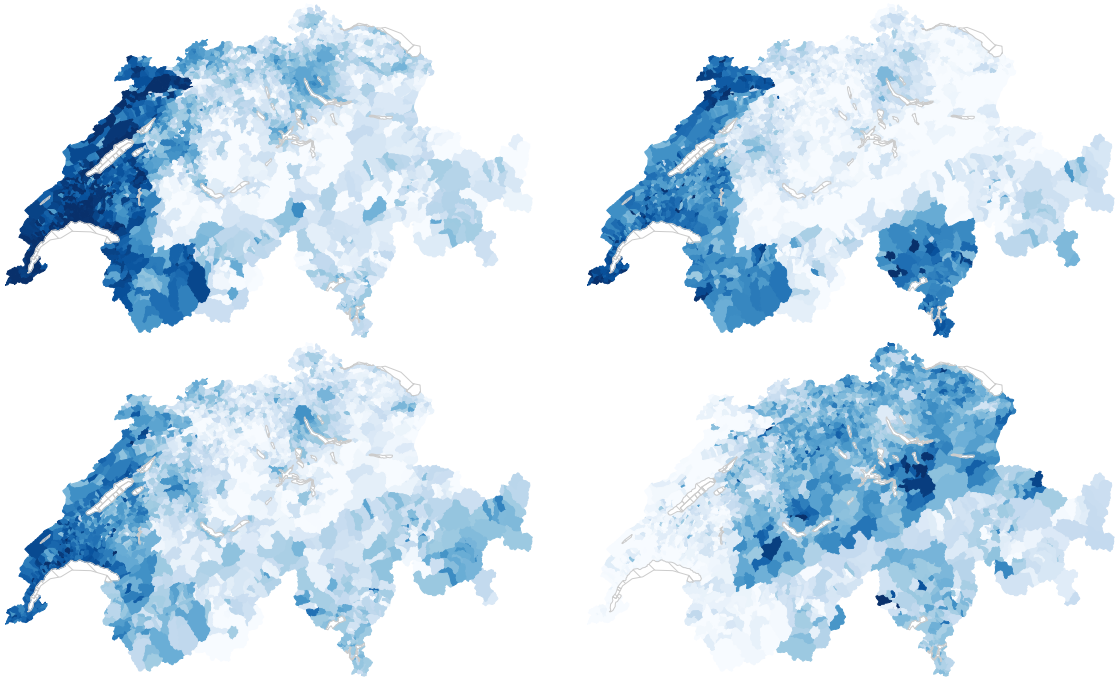
To take into account these similarities between votes and between municipalities, we view the problem as a *collaborative-filtering* problem [111] where each region expresses its opinion toward all votes. We use a latent-factor model to capture the *bi-clustering* of regions and votes [72, 100]. Moreover, to accurately predict when few regional results are available, we incorporate several features about the regions and votes, which addresses the *cold-start* problem [21]. These features not only improve predictions but also enable us to interpret and understand voting behaviors.

As we will show in Section 5.5.5, it is difficult to properly choose the hyperparameters of such a model. Taking a Bayesian approach enables us to properly set hyperparameters and make use of uncertainty to obtain stable predictions.

5.5.1 Notation

As before, we denote by y_{dn} the outcome of the nth vote in the dth region. We have $D = 2352$ regions and $N = 281$ votes. We gather the outcomes of the nth vote into in a D-dimensional vector $\boldsymbol{y}_n$ and all outcomes into the $D \times N$ matrix $\boldsymbol{Y}$. For each vote n, the national outcome $\bar{y}_n$ is the average of the regional results $\boldsymbol{y}_n$, weighted by the turnout in each region. We gather all national results into the N-dimensional vector $\bar{\boldsymbol{y}}$. We denote the 25 features of the dth region by $\boldsymbol{x}_d$ and the 13 features of the nth votes by $\boldsymbol{w}_n$ (both sets of features are described in Section 5.2). Finally, we gather the features $\boldsymbol{x}_d$

(a) "Röstigraben": French-speaking versus others



(b) City versus countryside

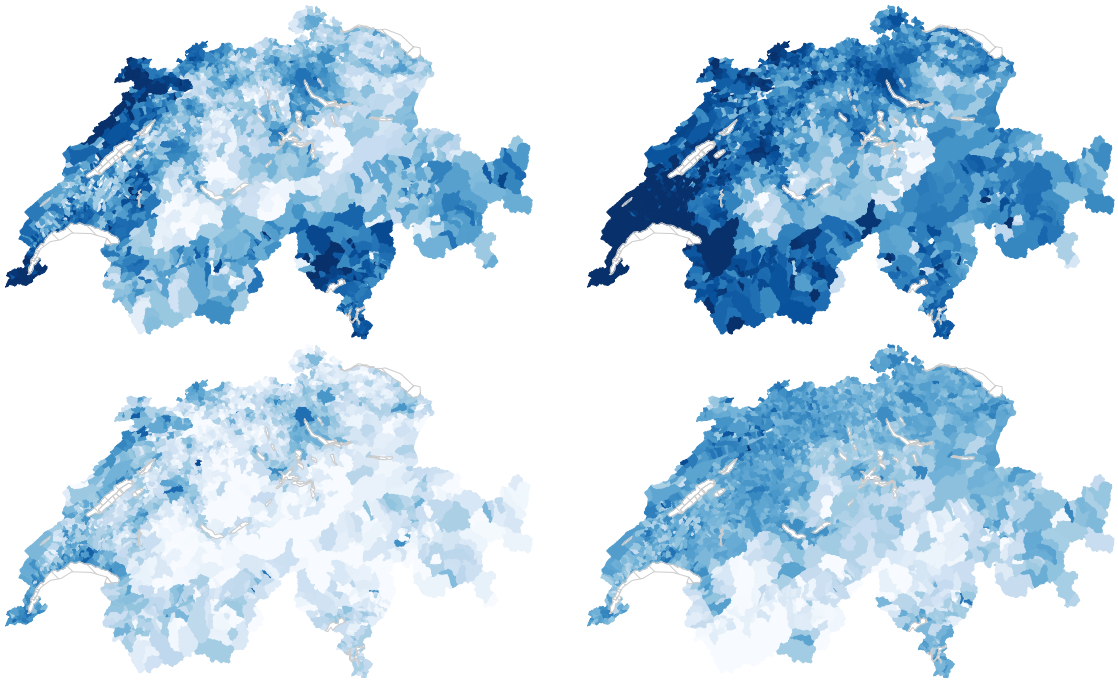


Figure 5.5 – Example of some patterns of results (from 0 % of "yes" in white to 100 % in dark blue). Many votes show similar result patterns. For example, we show votes where (a) the French-speaking regions (and sometimes the Italian-speaking regions) vote in opposition to the rest of the country, and (b) the densely-populated regions vote in opposition to the rural areas.

Variable	Dimension	Description
$D = 2352$	1	Number of regions, indexed by d
$N = 281$	1	Number of votes, indexed by n
y_{dn}	1	Outcome of the nth vote in the dth region
$\boldsymbol{y}_n$	D	Vector of the outcomes of the nth vote in all regions
$\boldsymbol{Y}$	$D \times N$	Matrix of the outcomes of all votes in all regions
$\bar{y}_n$	1	National outcome of the nth vote
$\bar{\boldsymbol{y}}$	N	Vector of the national outcomes of all votes
$\boldsymbol{x}_d$	25	Vector of the features of the dth region
$\boldsymbol{X}$	$25 \times D$	Matrix of the features of all regions
$\boldsymbol{w}_n$	13	Vector of the features of the nth vote
$\boldsymbol{W}$	$13 \times N$	Matrix of the features of all votes

Table 5.2 – Summary of the notation and the dataset sizes.

of all regions into the $25 \times D$ matrix $\boldsymbol{X}$ and the features $\boldsymbol{w}_n$ of all votes into the $13 \times N$ matrix $\boldsymbol{W}$.

Our dataset is thus $\mathcal{D} = \{\boldsymbol{Y}, \bar{\boldsymbol{y}}, \boldsymbol{X}, \boldsymbol{W}\}$. Table 5.2 summarizes the notation and the dataset sizes.

5.5.2 Goals

We are interested in predicting the outcome of a new vote with feature vector $\boldsymbol{w}_\star$ in all regions. Moreover, we would like to make these predictions in an online manner, i.e., to refine the predictions as more regional results are made available.

Suppose that, at a certain time t on the day of the vote, we have observed its outcome in $D_t < D$ regions. Denote the set of observed regions by $\mathcal{O}_t = \{d_1, d_2, \ldots, d_{D_t}\}$ (the ith observed outcome was that of region d_i). Denote the corresponding D_t-dimensional vector of outcomes by $\boldsymbol{y}_{\mathcal{O}_t,\star}$.

At all times t and given $\mathcal{D}$, $\boldsymbol{w}_\star$, and $\boldsymbol{y}_{\mathcal{O}_t,\star}$, our goal is thus to make the following two predictions:

1. predict the outcome $y_{d\star}$ in all regions $d \notin \mathcal{O}_t$,
2. predict the national outcome $\bar{y}_\star$.

In addition, we are interested in explaining and interpreting the reasons behind the predictions.

5.5.3 Model

A popular approach to model collaborative data such as ours is to use matrix factorization, where we assume that $\boldsymbol{Y}$ can be predicted using a low-rank matrix. As mentioned above, this low-rank model can be combined with a regression model to address the cold-start problem. We take a similar approach. We model the "preferences" z_{dn} of the dth region for the nth vote using an additive model with four components (we describe each component in detail below):

$$z_{dn} = \underbrace{\mu_n}_{\text{bias}} + \underbrace{f_n(\boldsymbol{x}_d)}_{\substack{\text{regression using} \\ \text{region features}}} + \underbrace{f_d(\boldsymbol{w}_n)}_{\substack{\text{regression using} \\ \text{vote features}}} + \underbrace{\boldsymbol{v}_d^T \boldsymbol{u}_n}_{\substack{\text{matrix} \\ \text{factorization}}}, \tag{5.1}$$

where $\mu_n \in$ is a bias, $f_n : \mathbb{R}^{25} \to \mathbb{R}$ and $f_d : \mathbb{R}^{13} \to \mathbb{R}$ are regression functions, and $\boldsymbol{v}_d \in \mathbb{R}^L$ and $\boldsymbol{u}_n \in \mathbb{R}^L$ are latent factors. A benefit of such a model is that we can obtain many specialized models by adding/removing components. For our analysis, this proves to be useful as it enables us to establish the significance of individual components. Below, we first describe each component in details and then list the combinations we use in our experiments.

Bias

The first component μ_n is a bias term for each vote n. We could also add a bias term μ_d for each region d and a global bias μ, although in our experiments these do not make any difference.

Regression using Region Features

The second component $f_n(\boldsymbol{x}_d)$ is a regression term that uses region features. We use two types of regression models. The first type is a linear regression model:

$$f_n(\boldsymbol{x}_d) = \boldsymbol{\beta}_n^T \boldsymbol{x}_d, \tag{5.2}$$

where $\boldsymbol{\beta}_n$ is a 25-dimensional weight vector associated to each vote n. We gather all weight vectors into the $25 \times N$ vote weights matrix $\boldsymbol{B}$. We assume that each $\boldsymbol{\beta}_n$ is sampled independently from a normal distribution with precision parameter $\lambda_\beta > 0$:

$$\boldsymbol{\beta}_n \sim \mathcal{N}(0, \lambda_\beta^{-1} \boldsymbol{I}),$$

where $\boldsymbol{I}$ is the identity matrix of the required dimension.

The second type of model is based on a Gaussian Process (GP) [97]. Here, each f_n is sampled independently from a GP with mean function $m(\cdot)$ and covariance function $k(\cdot, \cdot)$:

$$f_n(\boldsymbol{x}_d) \sim \mathrm{GP}(\, m(\boldsymbol{x}_d),\, k(\boldsymbol{x}_d, \boldsymbol{x}_{d'})\,).$$

We use a zero mean function and a squared-exponential (SE) covariance function with two hyperparameters $\sigma_s \in \mathbb{R}$ and $\boldsymbol{l} \in \mathbb{R}^{25}$:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_s^2 \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}')^T \mathrm{diag}(\boldsymbol{l}^2)^{-1}(\boldsymbol{x} - \boldsymbol{x}')\right),$$

where $\mathrm{diag}(\boldsymbol{l})$ is the diagonal matrix constructed from the vector $\boldsymbol{l}$.

Regression using Vote Features

The third component $f_d(\boldsymbol{w}_n)$ of Equation 5.1 is a regression term that uses vote features and is defined similarly to the model described in Equation 5.2:

$$f_d(\boldsymbol{w}_n) = \boldsymbol{\gamma}_d^T \boldsymbol{w}_n,$$

where $\boldsymbol{\gamma}_d$ is a 13-dimensional weight vector associated to each region d. Again, we gather all weight vectors into the $13 \times D$ region weights matrix $\boldsymbol{\Gamma}$. These weights also have a Gaussian prior, with associated precision parameter $\lambda_\gamma > 0$:

$$\boldsymbol{\gamma}_d \sim \mathcal{N}(0, \lambda_\gamma^{-1} \boldsymbol{I}).$$

We do not include a GP version of this component, because it gives similar results in our experiments but renders computations in the combined model significantly more complicated.

Matrix Factorization

The fourth and last component of Equation 5.1 is a matrix-factorization model. The latent features $\boldsymbol{v}_d$ are associated with the dth region and $\boldsymbol{u}_n$ are those associated with the nth vote. These latent features are vectors of length L to which we associate Gaussian priors, with precision parameter $\lambda_v > 0$ for $\boldsymbol{v}_d$ and $\lambda_u > 0$ for $\boldsymbol{u}_n$:

$$\boldsymbol{v}_d \sim \mathcal{N}(0, \lambda_v^{-1} \boldsymbol{I}),$$
$$\boldsymbol{u}_n \sim \mathcal{N}(0, \lambda_u^{-1} \boldsymbol{I}).$$

We gather the latent features of regions into the $L \times D$ matrix $\boldsymbol{V}$ and those of votes into the $L \times N$ matrix $\boldsymbol{U}$.

Observation Noise

Finally, given the preference z_{dn} of the dth region for the nth vote, we model the corresponding outcome as:

$$y_{dn} = z_{dn} + \epsilon_{dn},$$

where ϵ_{dn} is the observation noise.

For each vote and each region, the noise is drawn i.i.d. from a Gaussian prior:

$$\epsilon_{dn} \sim \mathcal{N}(0, \sigma_o^2),$$

where σ_o^2 is the noise variance.

This is not an ideal choice, as y_{dn} lies in the interval $[0, 1]$, however this choice does lead to simple inference algorithms. It is straightforward to use a different likelihood function by using more sophisticated inference methods such expectation-propagation or sampling methods [97].

5.5.4 Models and Inference Methods

We now summarize the models that we use in the remainder of this section. The purpose of our comparison is to demonstrate that Bayesian methods works well in the context of vote results prediction for two reasons. First, during online predictions, a Bayesian method takes the uncertainty into account and combines the components of Equation 5.1 appropriately to obtain accurate predictions. Second, during offline learning, a Bayesian approach enables us to find good hyperparameter values using the automatic relevance determination (ARD) framework [97, Section 5.1]. This approach, as we will show, generalizes much better than non-Bayesian methods such as cross-validation (CV).

We compare a variety of models to show results in favor of the above arguments. The complete list of the models we use in our experiments is given in Table 5.3. We describe below each model in detail.

The first model BIAS is our baseline and consists only of the bias term. For the Gaussian likelihood, μ_n is simply the sample mean of the vector $\boldsymbol{y}_n$. All of our models include the bias term.

The next three models are linear regression models involving a combination of $f_n(\boldsymbol{x}_d)$ using region features and $f_d(\boldsymbol{w}_n)$ using vote features. We fit LIN(r) and LIN(v) using least-squares (LS).

The LIN(r) + LIN(v) model simply combines regression terms on vote and on region features:

$$z_{nd} = \mu_n + \boldsymbol{\beta}_n^T \boldsymbol{x}_d + \boldsymbol{\gamma}_d^T \boldsymbol{w}_n.$$

We find the regression weights $\boldsymbol{B}$ and $\boldsymbol{\Gamma}$ by minimizing the following objective function:

$$f(\boldsymbol{B}, \boldsymbol{\Gamma}) = \frac{1}{2} \sum_{d,n} (y_{dn} - \boldsymbol{\beta}_n^T \boldsymbol{x}_d - \boldsymbol{\gamma}_d^T \boldsymbol{w}_n)^2 + \frac{1}{2} \lambda_\beta \sum_n \boldsymbol{\beta}_n^T \boldsymbol{\beta}_n + \frac{1}{2} \lambda_\gamma \sum_d \boldsymbol{\gamma}_d^T \boldsymbol{\gamma}_d.$$

Name	Description	Inference	Hyperparam.	Learning
BIAS	Bias term only	LS	-	-
LIN(r)	Linear regression using region features	LS	λ_β	CV
LIN(v)	Linear regression using vote features	LS	λ_γ	CV
LIN(r) + LIN(v)	Linear regression using region and vote features	ALS	$\{\lambda_\beta, \lambda_\gamma\}$	CV
GP(r)	GP regression using region features	Bayes	$\{\sigma_o^2, \sigma_s^2, \boldsymbol{l}\}$	ARD
MF	Matrix factorization	ALS	$\{\lambda_u, \lambda_v\}$	CV
MF + LIN(r)	Matrix factorization with linear regression using region features	ALS	$\{\lambda_u, \lambda_v, \lambda_\beta\}$	CV
MF + GP(r)	Matrix factorization with GP regression using region features	Bayes	$\{\sigma_o^2, \sigma_s^2, \boldsymbol{l}\}$	ARD
MF + GP(r) + LIN(v)	Matrix factorization with GP regression using region features and linear regression using vote features	Bayes	$\{\sigma_o^2, \sigma_s^2, \boldsymbol{l}, \lambda_\gamma\}$	ARD

Table 5.3 – Summary of the models we compare and the learning and inference methods we use. We give a short description of all models and list their hyperparameters. LS stands for *least-squares*, ALS for *alternating least-squares*, CV for *cross-validation*, Bayes for *Bayesian inference*, and ARD for *automatic relevance determination*.

If we consider $\boldsymbol{\Gamma}$ as fixed, we can minimize the above objective function with respect to each $\boldsymbol{\beta}_n$ independently:

$$f(\boldsymbol{\beta}_n) = \frac{1}{2}(\boldsymbol{y}'_n - \boldsymbol{X}^T\boldsymbol{\beta}_n)^T(\boldsymbol{y}'_n - \boldsymbol{X}^T\boldsymbol{\beta}_n) + \frac{1}{2}\lambda_\beta\boldsymbol{\beta}_n^T\boldsymbol{\beta}_n, \tag{5.3}$$

where $\boldsymbol{y}'_n = \boldsymbol{y}_n - \boldsymbol{\Gamma}^T\boldsymbol{w}_n$. Equation 5.3 is the objective function of ridge regression, whose solution is

$$\boldsymbol{\beta}_n = (\boldsymbol{X}\boldsymbol{X}^T + \lambda_\beta\boldsymbol{I})^{-1}\boldsymbol{X}\boldsymbol{y}'_n.$$

With $\boldsymbol{\Gamma}$ fixed, we can thus find each $\boldsymbol{\beta}_n$ using the above equation. Then, we fix $\boldsymbol{B}$ and apply the same procedure to find each $\boldsymbol{\gamma}_d$, alternating the two until convergence. We summarize the resulting procedure in Algorithm 4.

Intuitively, we expect LIN(r)+LIN(v) to be better than either LIN(r) or LIN(v). However, we will show that, if we use cross-validation to set the hyperparameters, the combination does not perform better than individual models.

Algorithm 4: Alternating least-squares algorithm for the LIN(r) + LIN(v) model

Input: Vote outcomes $\boldsymbol{Y}$, region features $\boldsymbol{X}$, vote features $\boldsymbol{W}$, weight precision parameters λ_β and λ_γ
Output: Weight matrices $\boldsymbol{B}$ and $\boldsymbol{\Gamma}$

Initialize $\boldsymbol{\Gamma}$
while *convergence criterion is not met* **do**
  $\boldsymbol{B} = \left(\boldsymbol{X}\boldsymbol{X}^T + \lambda_\beta \boldsymbol{I}\right)^{-1}\left(\boldsymbol{X}\left(\boldsymbol{Y} - \boldsymbol{\Gamma}^T\boldsymbol{W}\right)\right)$
  $\boldsymbol{\Gamma} = \left(\boldsymbol{W}\boldsymbol{W}^T + \lambda_\gamma \boldsymbol{I}\right)^{-1}\left(\boldsymbol{W}\left(\boldsymbol{Y}^T - \boldsymbol{B}^T\boldsymbol{X}\right)\right)$

The next model GP(r) is a GP regression model that uses region features. We use the standard Bayesian method for inference with GPs. We learn the hyperparameters by maximizing the marginal likelihood [97, Section 5.4].

The next four models combine regression models with the matrix-factorization model. The first model MF does not have any regression terms hence is expected to gives worse predictions for new votes. It expresses the outcome of a vote as the product of two latent factors:

$$z_{nd} = \mu_n + \boldsymbol{v}_d^T \boldsymbol{u}_n.$$

Similarly to LIN(r) + LIN(v), we find the parameters of MF by minimizing the following objective function [126]:

$$f(\boldsymbol{U}, \boldsymbol{V}) = \frac{1}{2}\sum_{d,n}(y_{dn} - \boldsymbol{v}_d^T \boldsymbol{u}_n)^2 + \frac{1}{2}\lambda_u \sum_n \boldsymbol{u}_n^T \boldsymbol{u}_n + \frac{1}{2}\lambda_v \sum_d \boldsymbol{v}_d^T \boldsymbol{v}_d.$$

If we consider $\boldsymbol{V}$ as fixed, we can minimize the above objective function with respect to each $\boldsymbol{u}_n$ independently:

$$f(\boldsymbol{u}_n) = \frac{1}{2}(\boldsymbol{y}_n - \boldsymbol{V}^T\boldsymbol{u}_n)^T(\boldsymbol{y}_n - \boldsymbol{V}^T\boldsymbol{u}_n) + \frac{1}{2}\lambda_u \boldsymbol{u}_n^T\boldsymbol{u}_n. \tag{5.4}$$

Again, Equation 5.4 is the objective function of ridge regression, whose solution is

$$\boldsymbol{u}_n = (\boldsymbol{V}\boldsymbol{V}^T + \lambda_u \boldsymbol{I})^{-1}\boldsymbol{V}\boldsymbol{y}_n.$$

With $\boldsymbol{V}$ fixed, we can thus find each $\boldsymbol{u}_n$ using the above equation. Then, we fix $\boldsymbol{U}$ and apply the same procedure to find each $\boldsymbol{v}_d$, alternating the two until convergence. We summarize the resulting procedure in Algorithm 5.

The second model MF + LIN(r) is an extension of MF, obtained by adding LIN(r). We train it using a combination of the two ALS methods presented above, where we append

Algorithm 5: Alternating least-squares algorithm for the MF model

Input: Vote outcomes $\boldsymbol{Y}$, latent features precision parameters λ_u and λ_v
Output: Latent feature matrices $\boldsymbol{V}$ and $\boldsymbol{U}$

Initialize $\boldsymbol{V}$
while *convergence criterion is not met* **do**
  $\boldsymbol{U} = \left(\boldsymbol{V}\boldsymbol{V}^T + \lambda_u \boldsymbol{I}\right)^{-1} \boldsymbol{V}\boldsymbol{Y}$
  $\boldsymbol{V} = \left(\boldsymbol{U}\boldsymbol{U}^T + \lambda_v \boldsymbol{I}\right)^{-1} \boldsymbol{U}\boldsymbol{Y}^T$

the explicit features of regions to their latent features. This model is expected to perform better than MF, but again we will show that CV leads to a sub-optimal performance.

LIN(r) is replaced with GP(r) in the third model, in order to enable the use of non-linear kernels. For inference, we will use a Bayesian method that works directly on z_{dn} and show that this model does not suffer from the problem of hyperparameters setting that affects MF + LIN(r). This is made possible by adapting the EM algorithm of Khan et al. [68] to select the hyperparameters using ARD. Most importantly, this method automatically chooses a good value for the hyperparameter σ_s in order to combine the MF and GP(r) terms optimally. It thus automatically finds the proper combination of the MF and GP(r) terms. We give the outline of this method in Algorithm 6.

The fourth model adds the component LIN(v) to address the cold-start problem of new votes. However, we cannot easily integrate this component into the EM algorithm presented above. Thus, we first remove its contribution from $\boldsymbol{Y}$ and then apply the EM algorithm on the remainder. As we will show, this model obtains the best performance of all the models, while remaining computationally simple.

Hyperparameter Learning

We keep the last 50 votes of our dataset as the test set and train our models on the first 231 votes. We implement all our models using Matlab.

For the non-Bayesian models, we use 10-fold cross validation to set the hyperparameters and we monitor the validation root-mean-square error (RMSE) to test the convergence. For each fold, we select 10 % of the outcomes as our validation data and use the rest to train the model. This means that, on average, we observe the results of 90 % of the regions for each vote of the training set, and we predict the outcome of the remaining 10 % of regions. The models are thus trained for weak generalization, as opposed to strong generalization. This is not optimal, as it means that the hyperparameters selected by this procedure will give the best results when we observe many outcomes for a vote, but not necessarily with only a few observations.

Algorithm 6: EM algorithm for the MF + GP(r) model

Input: Vote outcomes $\boldsymbol{Y}$, region features $\boldsymbol{X}$
Output: Latent feature matrix $\boldsymbol{V}$

Initialize $\boldsymbol{V}, \sigma_s^2, \boldsymbol{\theta} = \{\sigma_o^2, \boldsymbol{l}\}$
while *convergence criterion is not met* **do**
  `// Initialization`
  Compute the covariance matrix $\boldsymbol{K}$ using $\boldsymbol{X}$ and $\boldsymbol{\theta}$
  Compute its Cholesky $\boldsymbol{L} = \text{chol}(\boldsymbol{K})$
  Let $\boldsymbol{\Sigma} = \boldsymbol{V}^T\boldsymbol{V} + \sigma_s^2\boldsymbol{K}$
  Initialize $\boldsymbol{C} = \mathbf{0}$
  `// E step`
  for *each vote* n **do**
    $\mathbb{E}(\boldsymbol{t}_n) = \boldsymbol{\Sigma}(\boldsymbol{\Sigma} + \sigma_o^2\boldsymbol{I})^{-1}\boldsymbol{y}_n$
    $\text{cov}(\boldsymbol{t}_n) = \boldsymbol{\Sigma} - \boldsymbol{\Sigma}(\boldsymbol{\Sigma} + \sigma_o^2\boldsymbol{I})^{-1}\boldsymbol{\Sigma}$
    $\boldsymbol{C} = \boldsymbol{C} + \frac{1}{N}\left(\text{cov}(\boldsymbol{t}_n) + \mathbb{E}(\boldsymbol{t}_n)\mathbb{E}(\boldsymbol{t}_n)^T\right)$
  Let $\tilde{\boldsymbol{C}} = \boldsymbol{L}^{-1}\boldsymbol{C}\boldsymbol{L}^{-T}$
  Compute $\boldsymbol{R}$, the matrix of eigenvectors of $\tilde{\boldsymbol{C}}$, and $\boldsymbol{\Lambda}$, the corresponding diagonal matrix of eigenvalues
  `// M step`
  $\sigma_s^2 = \frac{\text{Tr}(\boldsymbol{R}) - \text{Tr}(\boldsymbol{\Lambda})}{D - L}$
  $\boldsymbol{V} = \boldsymbol{L}\boldsymbol{R}(\boldsymbol{\Lambda} - \sigma_s^2\boldsymbol{I})^{\frac{1}{2}}$
  Update $\boldsymbol{\Sigma} = \boldsymbol{V}^T\boldsymbol{V} + \sigma_s^2\boldsymbol{K}$
  Find $\boldsymbol{\theta}$ maximizing the likelihood of $\boldsymbol{y}_n \sim GP(0, \boldsymbol{\Sigma} + \sigma_o^2\boldsymbol{I})$ for all n

We could repeat this procedure for several percentage of observed results, e.g., with only 5 % of observed result for each vote. Our goal however is to have a single model that is able to make predictions with any number of observed results. We thus choose to train the ALS models with almost all the results observed, so that they have enough data to learn the global patterns of results. We show in Table 5.4 the hyperparameter values selected using 10-fold cross-validation for the non-Bayesian methods.

For the Bayesian models, we select the hyperparameters by maximizing the marginal likelihood [97, Section 5.4]. We can thus use all of the 231 train votes and do not need to use cross-validation[8]. We rely on the GPML toolbox [96] for the GP computations.

We use $L = 25$ as the dimension of the latent features of all models that have a MF component, as we empirically found that higher values do not increase the performances but result in longer training and evaluation times.

[8] As the hyperparameter of the LIN(v) component of MF + GP(r) + LIN(v) is not automatically set by the EM algorithm, we set it to $\lambda_\gamma = 200$. We empirically found that this value is small enough to take advantage of the vote features for early predictions, while not damaging the end performances.

Model	λ_u	λ_v	λ_β	λ_γ
LIN(r)	—	—	34	—
LIN(v)	—	—	—	32
LIN(r) + LIN(v)	—	—	36	80
MF	0.03	31	—	—
MF + LIN(r) (CV)	0.08	28	100	—
MF + LIN(r) (hand)	0.03	31	34	—

Table 5.4 – Hyperparameters selected for each of the non-Bayesian models, using 10-fold cross validation. We also show the hyperparameters of the hand-tuned version of MF + LIN(r) shown in Figure 5.7, that are simply the hyperparameters found by CV for its individual components.

5.5.5 Results

Our goal is to estimate the accuracy of the online predictions. We thus proceed as follows. First, we pick a random reveal order, which specifies the order in which we observe the results of 90 % of the regions (2116). Then, we define the last 10 % of these regions (236) as the test regions, on which we will evaluate the error. We report the RMSE on the last 10 % of regions, averaged over the 50 test votes and 500 random reveal orders. As the error over the 50 test votes and 500 orders does not vary significantly, we do not show error bars on the figures below.

We first show in Figure 5.6 the results of the baseline, BIAS, and of the models that only use linear regression. As the standard error of our results is small, we do not show error bars on the figures presented below. All the difference in performances are significant.

LIN(v) gets the best early performances, meaning that the voting recommendations of parties are useful when few observations are available. With many observed regions, however, LIN(v) quickly reaches its limit. LIN(r) starts with performances similar to those of BIAS and worse than those of LIN(v), but quickly outperforms both BIAS and LIN(v). LIN(r) + LIN(v) gets a slight advantage over LIN(r) by taking into account the voting recommendations, but is not able to reach the same early performance as LIN(v), as the hyperparameters selected by CV tend to overpenalize the component that uses vote features.

The same problem occurs with MF + LIN(r), as shown in Figure 5.7. First, we see in this figure that MF needs a few hundred observed regions before it can get performances better than LIN(r), as it needs to properly estimate the latent features of the new vote. By combining the MF and LIN(r) components, we obtain performances better than with MF only, but the hyperparameters obtained by CV again do not result in performances matching those of the individual models at all time.

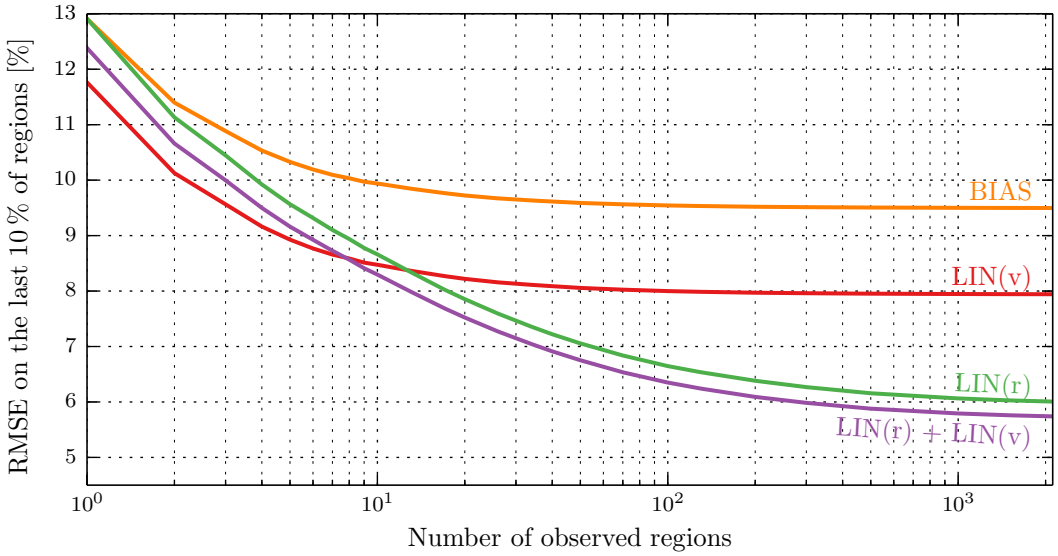


Figure 5.6 – Results of the BIAS, LIN(r), LIN(v), and LIN(r)+LIN(v) models. We show the RMSE on the predicted result of the last 10 % of regions, averaged over 500 random reveal orders and 50 test votes. While LIN(r) + LIN(v) achieves the best performance with many observed regions, its early performances do not match those of LIN(v).

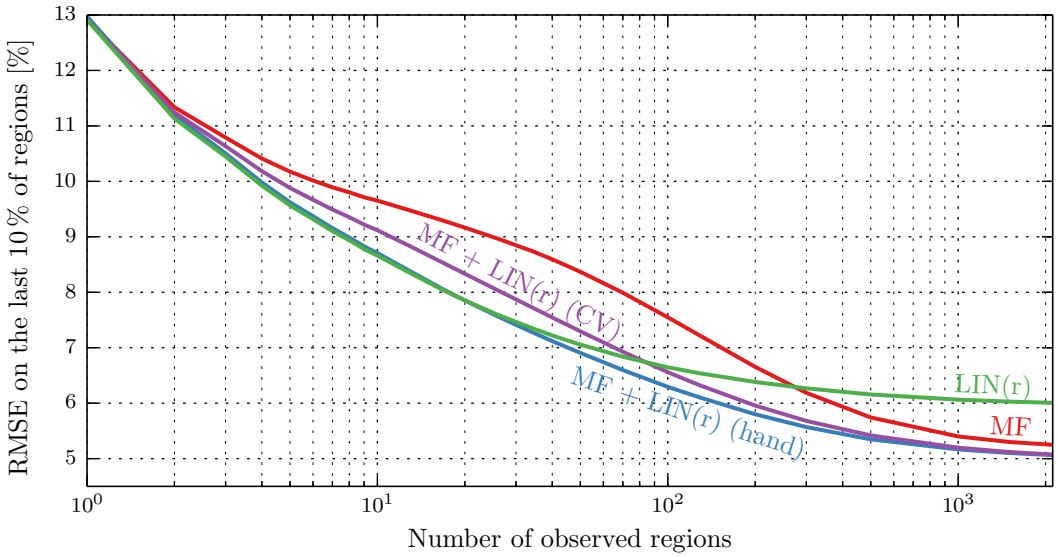


Figure 5.7 – Results of the LIN(r), MF, and MF + LIN(r) models. We show the RMSE on the predicted result of the last 10 % of regions, averaged over 500 random reveal orders and 50 test votes. Similar to Figure 5.6, the hyperparameters selected by CV for MF + LIN(r) do not result in performances matching those of the individual models with all numbers of observed regions. By hand-tuning the hyperparameters, however, we are able to obtain a model that has good performances all along, suggesting that there is room for improvement over CV.

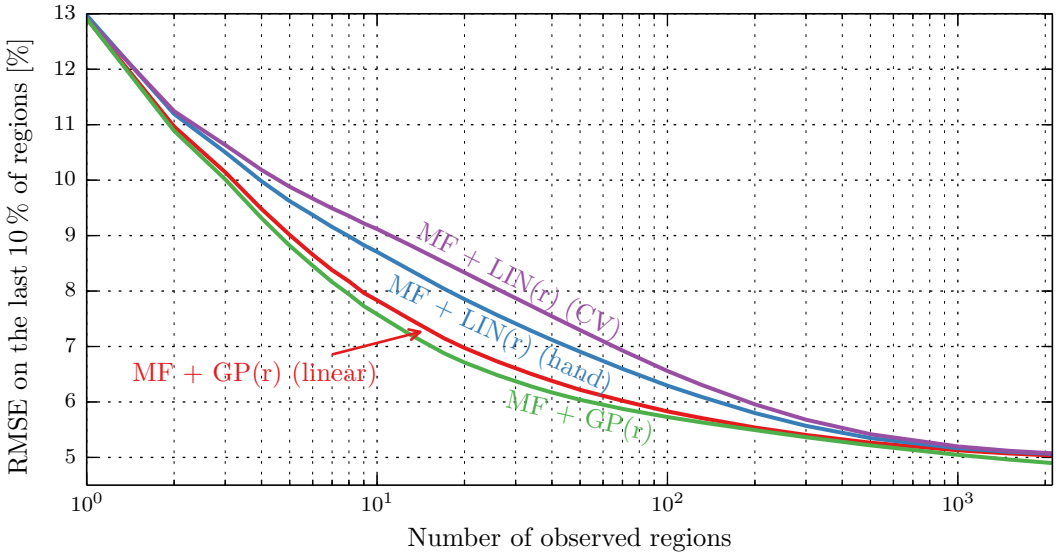


Figure 5.8 – Results of the MF+LIN(r) and MF+GP(r) models. We show the RMSE on the predicted result of the last 10 % of regions, averaged over 500 random reveal orders and 50 test votes. The Bayesian model MF+GP(r) gets better performances with few observed regions than both the cross-validated and hand-tuned MF + LIN(r) models, even with a simple linear kernel.

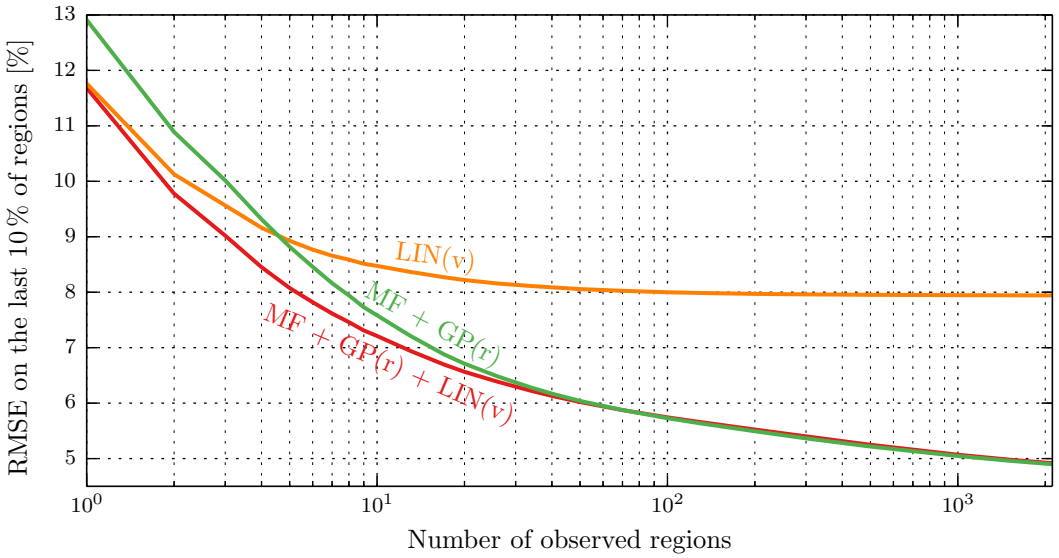


Figure 5.9 – Results of the LIN(v), MF + GP(r), and MF + GP(r) + LIN(v) models. We show the RMSE on the predicted result of the last 10 % of regions, averaged over 500 random reveal orders and 50 test votes. MF + GP(r) + LIN(v) is able to properly combine the LIN(v) component with MF + GP(r) to obtain both good early performances and good results with many observed regions.

However, by hand-tuning these parameters (using the parameter values shown in Table 5.4), we obtain a model that matches the early results of LIN(r) and gets performances better than MF with many observed regions. This means that there is still room for improvement over the model obtained with CV.

We show in Figure 5.8 that the solution to this problem is to use a Bayesian model. Indeed, the MF + GP(r) model is able to beat both the cross-validated and the hand-tuned versions of MF + LIN(r). To make sure that the improved performance does not come from the non-linearity of the SE kernel of the GP(r) component, we also show a variant of the MF + GP(r) that uses a linear isotropic kernel, which also beats the two non-Bayesian models.

Finally, we show in Figure 5.9 that adding the LIN(v) component to MF + GP(r) enables us to obtain the same early performances as those of LIN(v), completing the model. The final combination MF + GP(r) + LIN(v) thus obtains the best overall performances.

5.5.6 National Results Prediction

We can use the models presented above to predict the national result of a vote. To do so, we first predict the result in all unobserved regions, using the result of those observed. Then, we simply compute the average of the results of all regions (observed and predicted), weighted by their population. To achieve the most accurate predictions of the national result, we should use the turnout in each region as the weights, instead of their population. However, we do not have access to this information on the day of the vote, hence cannot use it for the prediction.

Similarly to the results presented above, we show in Figure 5.10 the absolute error on the national result, averaged over 50 test votes and 500 random reveal orders. The difference in performance between MF + GP(r) and MF + GP(r) + LIN(v) is smaller than when predicting individual regions. They can both predict the national outcome of a vote with an absolute error smaller than 1 %, after having observed the result of only 50 regions.

As we show in Figure 5.11, the national outcome of the votes in our dataset are very diverse, spanning nearly the entire interval of possible results. To investigate whether votes whose outcome is close to 50 % have a larger error than others, we show in Figure 5.12 the relationship between the true outcome of votes and the error made by the MF + GP(r) + LIN(v) model. We first group the 50 test votes by their national result into eight bins, each 10 % wide. We then consider these bins separately, and show—for each bin—the distribution of the RMSE on the national result with 50 observed regions, over 500 random reveal orders. We see that there is no systematic relationship between the outcome of a vote and the error made by MF + GP(r) + LIN(v), with very similar distribution of errors over all bins. Therefore, our model is not biased towards any particular type of vote and can predict all votes equally well.

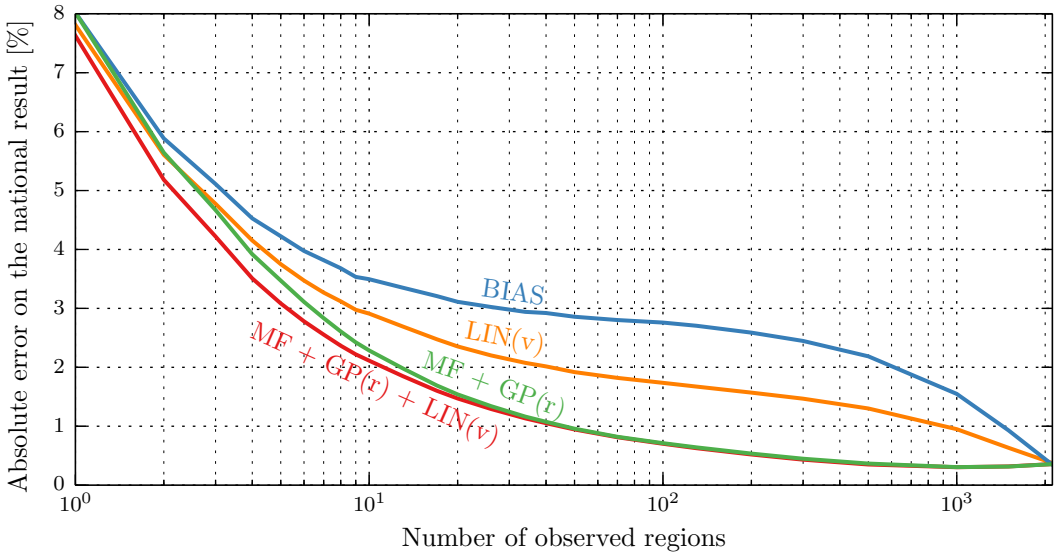


Figure 5.10 – Results of the national prediction for selected models. We show the absolute error of the predicted national result, averaged over 500 random reveal orders and 50 test votes. Both MF + GP(r) and MF + GP(r) + LIN(v) are able to predict the national outcome of a vote within 1 % using the results of only 50 municipalities.

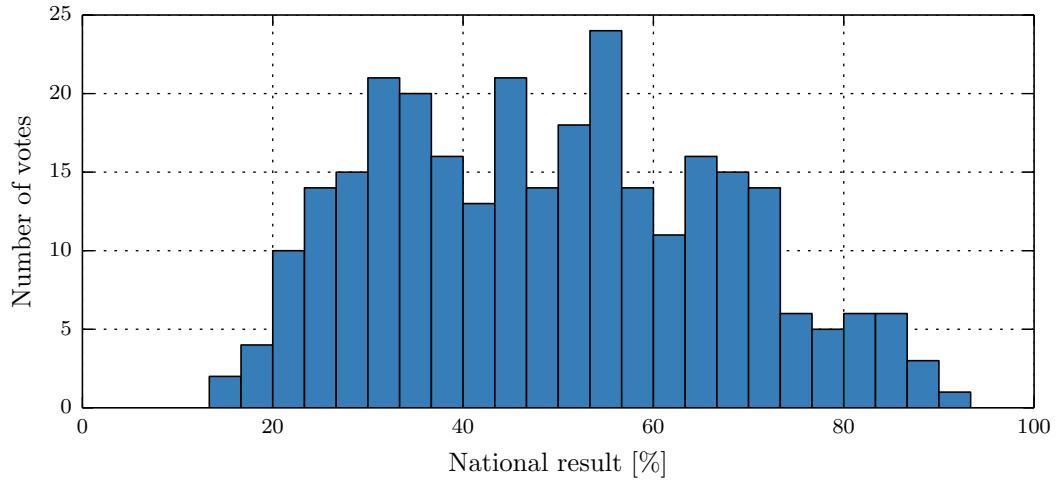


Figure 5.11 – Distribution of the national results of the 281 votes in our dataset. The national results span nearly the whole range of possible results and are not biased towards the extremes.

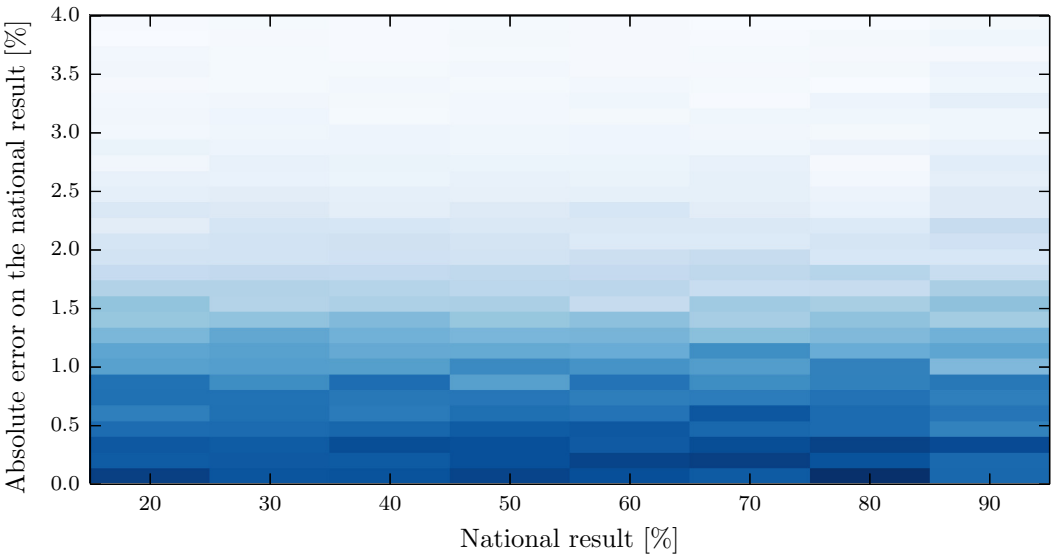


Figure 5.12 – Distribution of the absolute error of national predictions with respect to the true national outcome of the vote. We group the 50 test votes by their national result into eight bins. For each bin (shown as columns), we show the distribution of the absolute errors made by the MF + GP(r) + LIN(v) model with 50 observed results, over 500 random reveal orders for each vote of the bin. There is no systematic relationship between the national result and the errors made by the model.

Finally, we compare the accuracy of the models when predicting the binary outcome of a vote, i.e., whether it is accepted or not. To do so, we simply predict the national result as explained above, and then convert this predicted to a binary outcome. We show in Figure 5.13 the accuracy of these predictions for several models. With binary predictions, we see that the models with the LIN(v) component get a slight advantage when observing only a few regions, similar to what we observed in Figure 5.9. For example, MF + GP(r) + LIN(v) obtains an accuracy of 99 % with just 100 observed regions. We see a drop in accuracy with many observed regions, which could result from using the population of regions instead of the true turnout when computing the national result.

5.5.7 Model Interpretation

As we already mentioned, one of the advantages of the models presented in this section is that they are easily interpretable. This means that political scientists, for example, could use such models to study the voting behavior of a country and verify the effect of some characteristics of the regions.

To illustrate the interpretability of these models, we show in Figure 5.14 the relative importance of the features of the regions, as learned by the MF + GP(r) + LIN(v) model. These weights can be directly obtained from $\boldsymbol{l}$, one of the hyperparameters of the GP(r)

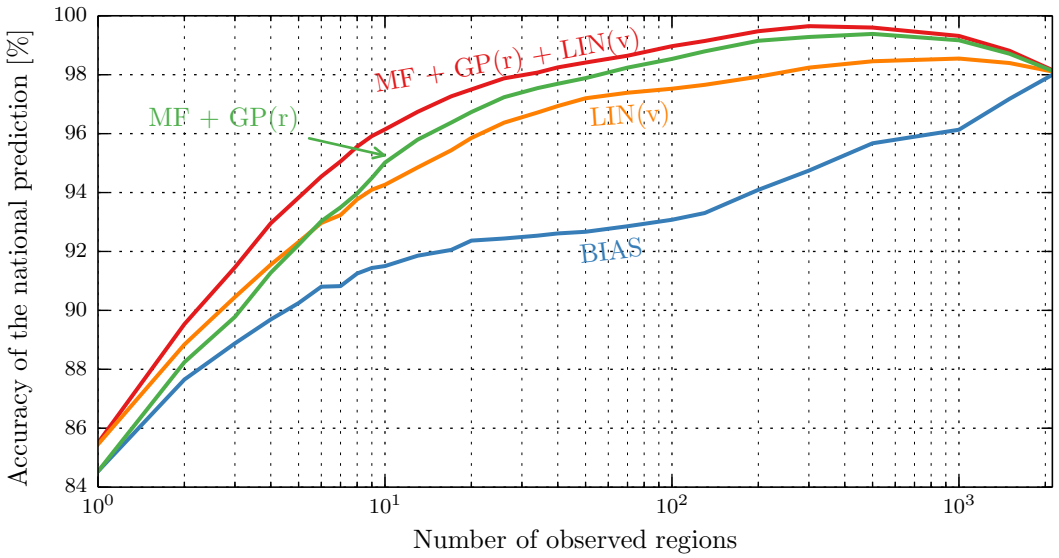


Figure 5.13 – Accuracy of the national binary predictions for selected models. We show the proportion of the national binary outcomes that are correctly predicted, over 500 random reveal orders and 50 test votes. MF + GP(r) + LIN(v) obtains an accuracy of 99 % with just 100 observed regions.

component. We see that this component mostly explains the correlation between the results of two regions using their geographical proximity, and then their election results, i.e., their political orientations.

To explore further the correlation between regions, we show in Figure 5.15 a map of Switzerland with its cantons outlined. On this map, we draw a line between two municipalities if their correlation according to the MF + GP(r) + LIN(v) model is higher than 0.8. Again, such a map could lead to interesting interpretations. For example, we see that the Zürich area, in the North, is heavily clustered. We also clearly see a separation between the French-speaking and the German-speaking parts of the canton of Valais. This difference of voting behaviors in the canton of Valais could already be seen in Figure 5.1.

5.6 Related Work

To the best of our knowledge, we are the first to investigate this dataset of *issue votes* and to make a large-scale study of the voting behaviors in a country, at such a fine geographical level. While issue votes have not been extensively studied, there is a large body of work that studies *elections* [98, 67, 54]. Similar to the analysis presented in Section 5.3, Agnew [22] investigates the geographical voting patterns of Italy, in order to assess the homogeneity of the country.

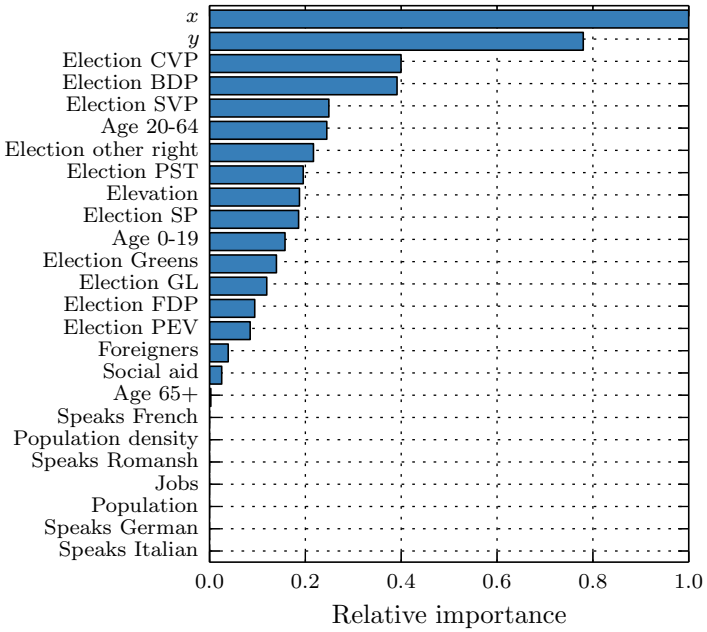


Figure 5.14 – Relative importance of the region features learned by the GP(r) component of the MF+GP(r)+LIN(v) model. We see that most of the correlation between two municipalities is explained by their geographical proximity, with more weight given to the X-axis as it also partially encodes the language dimension (a large distance on the X axis usually implies different languages).

Armstrong and Graefe [24] use biographical information about candidates to predict U.S. election results. Recently, several studies have focused on Twitter data to predict the outcome of elections, from Germany [115] to the Netherlands [101] and Singapore [104]. However, some researchers (see e.g., Gayo [53]) have warned against relying only on tweets to predict election results, arguing that the data is inherently biased and that missing signals could be more important than observed ones.

The models we introduce in Section 5.5 are not new. Matrix factorization and regression have been applied in similar collaborative settings, both individually [80, 126, 100] and combined with each other [21, 110]. However, we believe we are the first to apply such models to the problem of predicting the outcome of issue votes.

5.7 Summary

In this chapter, we introduce several applications of a dataset of issue votes. We first use this dataset to study the voting behavior of Swiss municipalities, and its evolution over time. We show that simple dimensionality-reduction techniques enable us to highlight well-known characteristics of the political landscape of Switzerland, such as the difference in voting behavior between the French-speaking and the German-speaking regions.

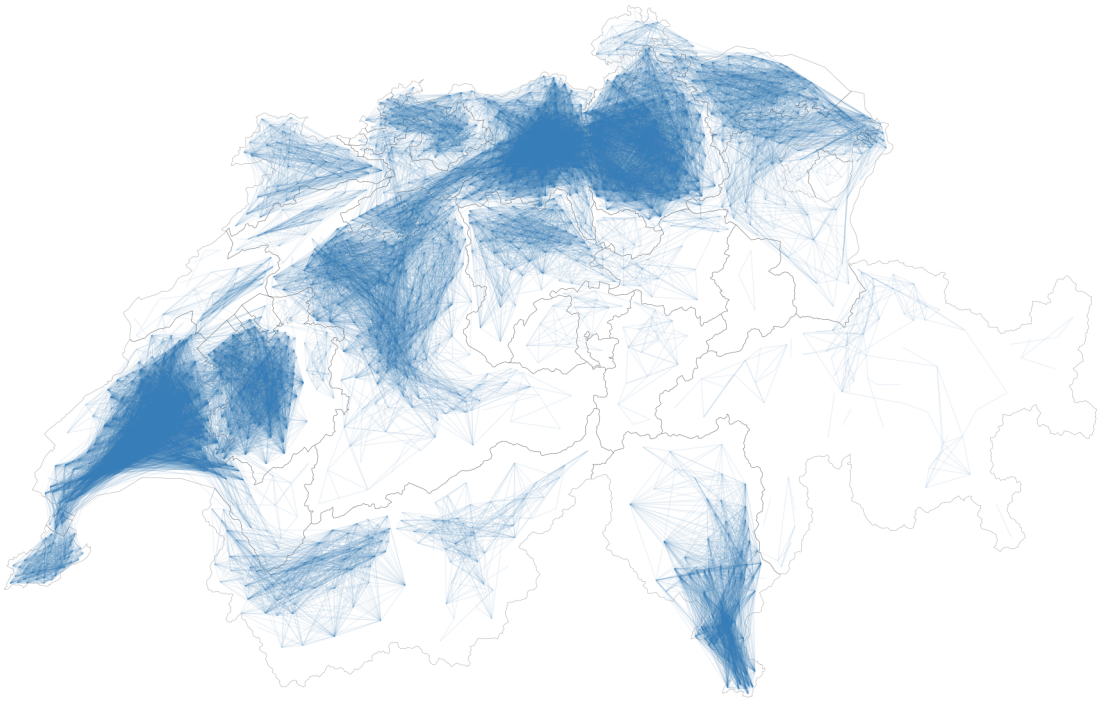


Figure 5.15 – Visualization of the correlation between the results of municipalities captured by the MF + GP(r) + LIN(v) model. We show a link between two regions if their correlation is larger than 0.8, and add the boundaries of the cantons to have a frame of reference. There are clusters that correspond to cantons, but we also see many correlated regions around Zürich, in the North. Interestingly, we can clearly see the separation between the French-speaking and the German-speaking parts of the canton of Valais, in the southwestern part of the country.

We then introduce the problem of predicting the outcome of a vote. Such a prediction could be useful to the media, polling agencies, and the political actors of the country. We first show that it is possible to identify individual regions that have a high predictive power of the binary national outcome.

We then reformulate the vote prediction problem into a more general setting, where we jointly predict the outcome in all regions, given a few observed results. We take a principled approach and show that combining a matrix-factorization component with regression terms that use features about both the regions and the votes enables us to predict vote results with a high accuracy. Moreover, we demonstrate that Bayesian methods are superior to non-Bayesian ones for this application, with proper hyperparameter setting leading to better performances. Finally, we illustrate the interpretability of the resulting model with two simple examples.

6 Conclusion

> Getting an education was a bit like a communicable sexual disease. It made you unsuitable for a lot of jobs and then you had the urge to pass it on.
>
> *Terry Pratchett*

In this thesis, we showed that there is a great potential for innovation at the intersection of computer science and fields such as social science, economics, and political science. Nowadays, these fields generate large amounts of data, that computer scientists are sometimes better equipped to deal with than most other researchers. We demonstrated that the proper application of techniques from the data-mining and machine-learning communities enables us to bring significant contributions to practical problems. In particular, we showed that combining models and relevant datasets is key to the quality and usefulness of the results we obtain.

In Chapter 2, we presented our work on human-mobility prediction. We described our participation to the Nokia Mobile Data Challenge, that enabled us to win the Next-Place Prediction Task. The reasons for our winning are twofold: First, we identified non-stationarity as one of the key characteristics of the data that rendered the task difficult, and implemented a home-change detection algorithm for dealing with non-stationary users. Second, we developed three different types of predictors, that obtain different results for the same users but have globally similar performances, and we combined them by taking this diversity into account. The combined predictions all outperformed the submissions of other competitors in the challenge.

In Chapter 3, we introduced a novel dataset of crowdfunding campaigns that we extracted from Kickstarter. We showed that using information about the money pledged to campaigns is sufficient to predict accurately after a couple of days if the campaigns are going to reach their funding goal or not. However, this information does not enable

us to distinguish properly, after a few hours, the campaigns that are going to succeed from those that are likely to fail. By adding data extracted from the social network of Kickstarter and from Twitter, and by combining the resulting models, we were able to significantly improve early predictions. In particular, predictions made by the combined model on average four hours after the beginning of a campaign are 4 % more accurate. We made the dataset available to the scientific community and built an online platform (described in Appendix A) to make the predictions available to the public.

In Chapter 4, we studied the ideology of the citizens and the politicians of Switzerland. Using data extracted from a voting advice application, we showed that the traditional left-/right-wing and liberal/conservative views of the political system are indeed the most efficient two-dimensional representations of political opinions. In order to raise awareness about the potential misuse of such data, we introduced a method for "abusing" voting advice applications: We crafted a fake candidate profile that would have obtained twice as many voting recommendations as any other real candidate. We also proposed a technique for monitoring elected politicians, by combining their pre-electoral opinions with the votes that they cast once elected, in order to detect potential changes of opinion after the elections.

Finally, in Chapter 5, we presented a novel dataset of issue votes. We showed that such publicly available data can be used to automatically uncover typical patterns in the voting behaviors of small geographical regions. Moreover, we identified regions that have a high predictive power of the binary national outcome of votes. We then showed in a principled way that we obtain accurate predictions of the outcome of votes in all regions by combining matrix factorization with regression using features about both votes and regions. We also demonstrated how taking a Bayesian approach enabled us to properly choose the hyperparameters of our models, and how the resulting models could lead to useful interpretations.

On the Need for Collaboration

As we mentioned several times, the problems we solved in this thesis have their roots in fields quite far from computer science, such as social science and economics. As computer scientists, we are most of the time better equipped to deal with the data generated by these fields: The data-mining and machine-learning communities have produced numerous techniques to process, analyze, and predict quantities extracted from these datasets. One problem remains, however: we usually lack the deep understanding of these fields that is required to go beyond the sometimes simplistic interpretations presented in this thesis. We thus need to complete the journey: After having looked in these fields for problems to solve, we must go back there with our solutions. Collaborating with political scientists, for example, is the logical next step after the analysis presented in Chapters 4 and 5.

Automated Machine Learning

There are only a few of us, computer scientists, but so many of them, researchers from political science, biology, social science, history, etc. It is thus not sustainable to hope that one of us will be here for each of them, to help them make sense of their data and extract the information they need. The recent developments in automated machine learning [114, 48] are in our opinion a good solution to this problem. The goal of this emerging discipline is to "take the human expert out of the loop" [2] and to design machine-learning frameworks that are directly useable by end users, i.e., the people that understand the data. Such frameworks would enable researchers from other fields to directly use data-mining and machine-learning techniques on their dataset, without needing our intervention.

Keeping the Users in the Loop

The automated machine learning effort is a step in the right direction: It puts the outcome of the research of computer scientists back into the hands of people, for them to use. This is the reason why we built the two websites described in Appendices A and B: to make it possible for *regular* people to use the outcome of our research. While we understand that it is not always possible, and that theoretical research is sometimes hard to apply in practice, we strongly believe that all researchers should take the extra steps needed to produce useable and understandable results. This requires of course some additional efforts that are sometimes hard to justify in the competitive world of scientific research, but we are deeply convinced that they are worth the investment.

A Sidekick: Real-Time Success Prediction of Kickstarter Campaigns

To make the research presented in Chapter 3 available to the public, we developed Sidekick [13], a website showing real-time success predictions of Kickstarter campaigns.

Live Campaigns

The data collected through the process described in Section 3.2 is directly available on Sidekick. We show all live campaigns with their characteristics (category, funding goal, etc.), current status (number of backers and amount of pledged money), and predicted probability of success. The predictions are obtained using the combined model described in Section 3.4. Figure A.1 shows the home page of Sidekick, which lists campaigns that are close to their end. Visitors can search, filter, and sort campaigns, based on their attributes.

Campaign Page

Each live campaign also has a dedicated page, with more detailed information. We show on this page the evolution of pledges over time, as well as the history of the predictions the model made for this campaign.

Figure A.2 shows an example of a campaign page. We see that the probability of success went from close to 0 to nearly 100 % after roughly one third of the campaign. While this campaign only has two days left and still needs to raise another 30 % of its goal, the algorithm gives it a success probability of 84 %.

Appendix A. Sidekick: Real-Time Success Prediction of Kickstarter Campaigns

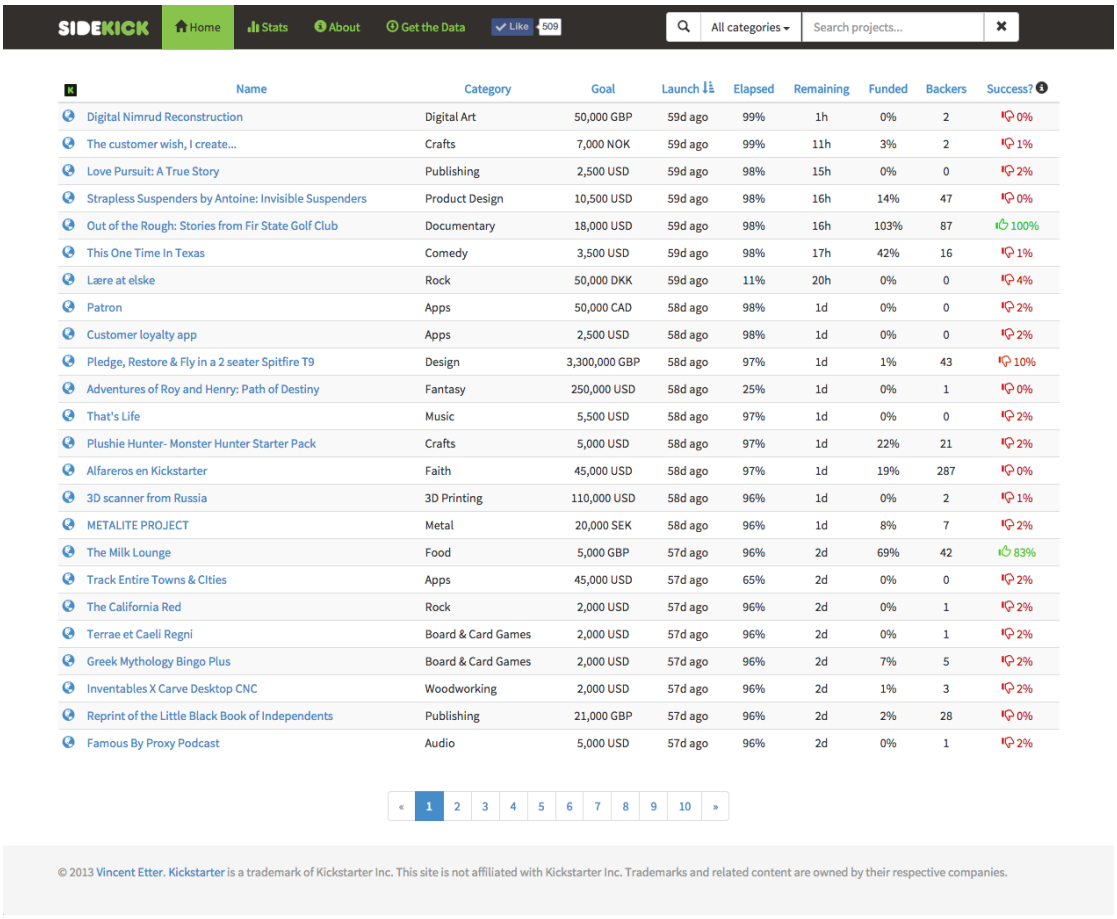


Figure A.1 – Home page of Sidekick, the platform showing real-time success predictions of Kickstarter campaigns. It shows live campaigns that are close to their end, with their current status and the corresponding predicted probability of success.

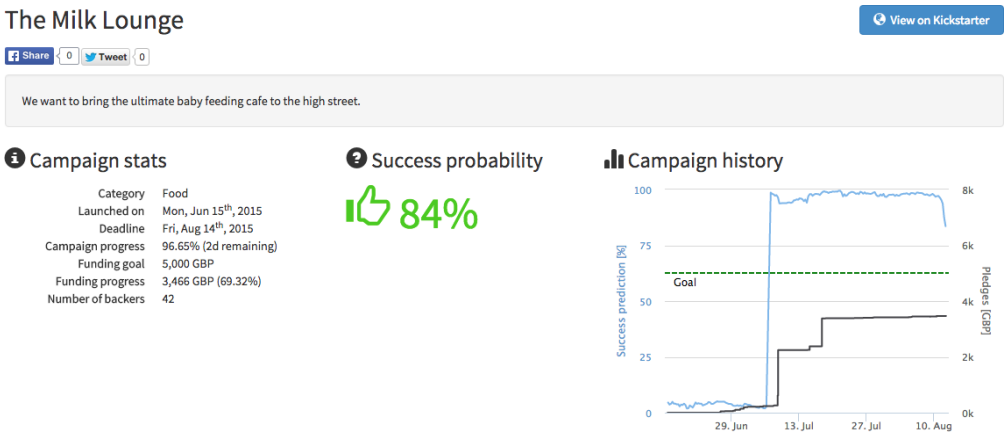


Figure A.2 – Example of a page dedicated to a campaign. It shows some information about the campaign, the evolution of its pledges over time, as well as the evolution of the predicted probability of success.

B Predikon: Visualizing Vote Results and Voting Patterns

To share some of the results of our exploration of political data, we developed Predikon [10], a website enabling people to visualize the result of issue votes in Switzerland, as well as the voting patterns presented in Section 5.3.

Data

We collect the outcome of votes directly on the website of the Swiss Federal Statistical Office [47]. We show municipalities as they currently exist, and interpolate the past results of those that are the outcome of a fusion or a division of municipalities, as explained in Section 5.2.1.

In addition to the outcome of each vote, we display some information about each municipality. This information was extracted from the Wikipedia page of the municipalities, as well as from municipality portraits published by the Swiss Federal Statistical Office [46].

Implementation

The back end of the website is implemented using *Python* and the *Flask* microframework [6]. The front end of the website is implemented using *HTML5* and *Javascript*. All the interactive visualizations use the *D3.js* library [3].

Vote Results

The result of all Swiss issue votes can be visualized, from 1848 to today. Votes that took place before 1981 only have results at the cantonal level, whereas recent votes have results at the municipal level. For each vote, we show a map of Switzerland with cantons (or municipalities) colored according to their outcome: green regions accepted the issue, yellow regions were undecided, and red regions rejected the issue. Figure B.1 shows an

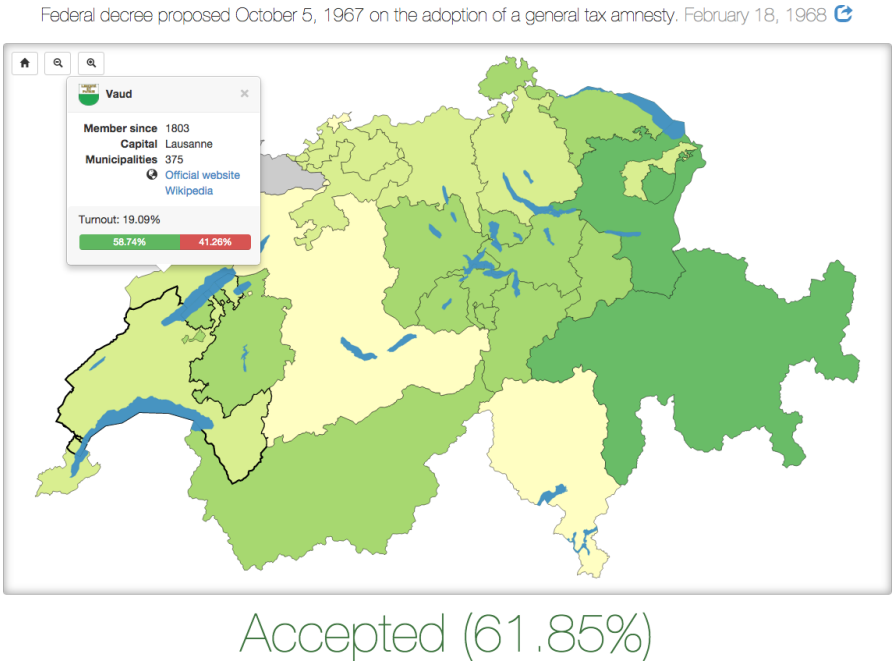


Figure B.1 – Example of the visualization of the outcome of a vote that took place before 1981 and for which results are only available at the cantonal level. We include some basic information about cantons, as well as links to their Wikipedia page and official website.

example of a vote with results available only at the cantonal level and Figure B.2 shows a recent vote with municipality results.

Voting Patterns

In addition to vote results, the platform also presents an interactive version of the voting patterns visualization shown in Figure 5.1. It shows the position of each municipality in the two-dimensional space illustrated in Figure 5.1(a), as well as the colored map presented in Figure 5.1(c). Visitors can chose to color the municipalities in the two-dimensional space according to various factors: language, population, strength of political parties, etc. In Figure B.3, municipalities are colored on the right according to their population density, highlighting the fact that cities are at one end of their "cloud", and rural regions at the other.

Below the interactive map shown in Figure B.3, we show the 10 votes that have the most weight in the first (Figure B.4, top) and second (Figure B.4, bottom) eigenvectors used to compute the two-dimensional representation of municipalities. The map on the left side gives an overview of the pattern of results for each vote. The bar on the right side shows the relative importance of each vote, as well as to which side of the axis a "yes" to this vote corresponds.

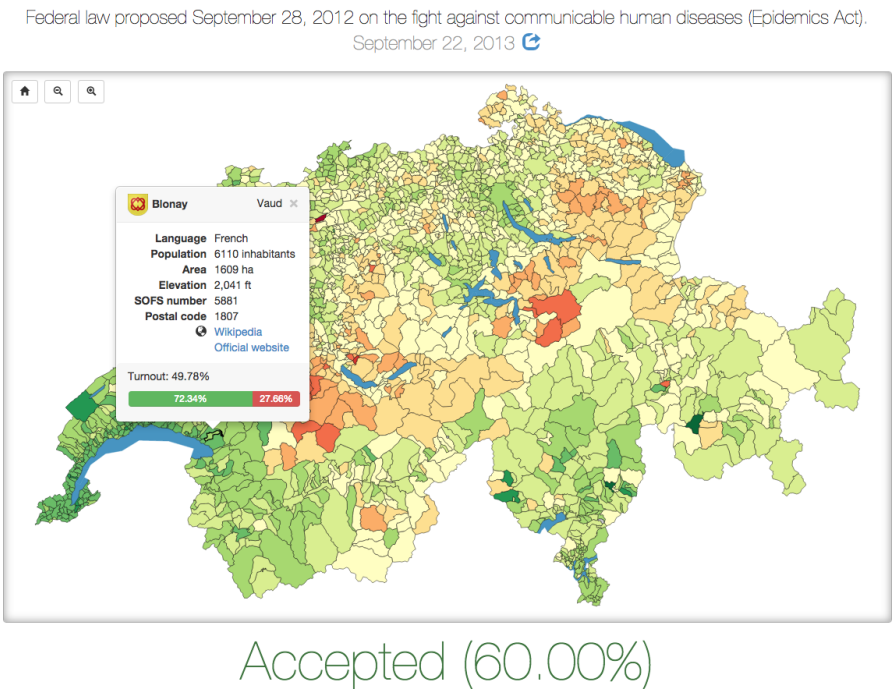


Figure B.2 – Example of the visualization of the outcome of a vote that took place after 1981 and for which results are available at the municipal level. We include some basic information about municipalities, as well as links to their Wikipedia page and official website.

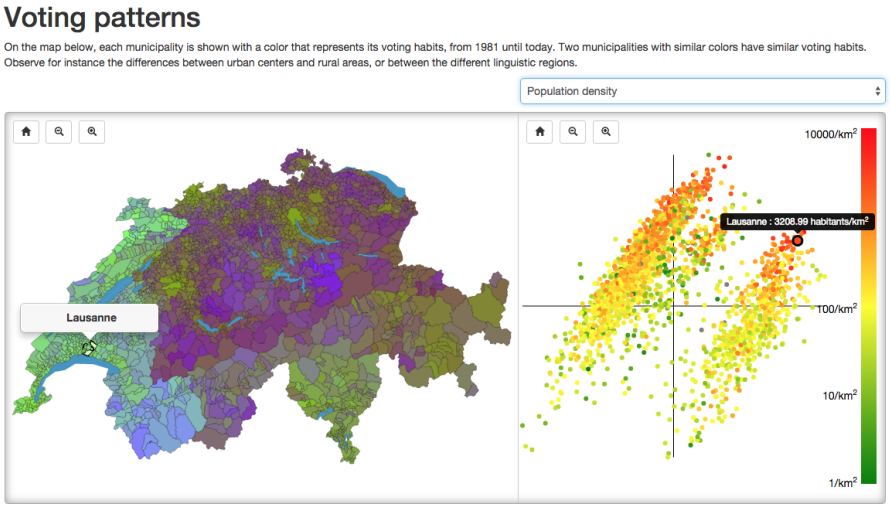


Figure B.3 – Visualization of the voting patterns described in Section 5.3. The left side of the page shows the map of Switzerland with municipalities colored according to their position in the two-dimensional representation on the right. Visitors can color municipalities on the right side according to several features, such as their population density.

Composition of the horizontal axis:

Results	Vote date	Vote name	Relative importance
	December 6, 1992	Federal decree on the European Economic Area (EEA) of October 9, 1992	100.0%
	September 22, 1985	Federal decree proposed October 5, 1984 on the popular initiative "calling for the harmonization of the beginning of the school year in all cantons" (counter-proposal)	98.13%
	June 13, 1999	Federal law on maternity insurance	97.33%
	September 26, 2004	Change proposed October 03, 2003 of the federal law on the income loss allowances for people serving in the military, civil service or civil protection (Allowances act for income loss)	92.94%
	September 26, 2004	Federal decree proposed October 03, 2003 on the acquisition of nationality by third-generation foreigners	79.88%
	April 18, 1999	Federal decree on an update of the Federal Constitution	79.73%
	September 26, 2004	Federal decree proposed October 03, 2003 on the regular naturalization and the facilitated naturalization of second-generation young foreigners	78.18%
	June 1, 2008	Popular initiative proposed November 18, 2005 « For democratic naturalizations »	69.89%
	March 3, 2013	Federal decree proposed June 15, 2012 on family policies	68.69%
	December 4, 1994	Federal act on health insurance of March 18, 1994	64.86%

Composition of the vertical axis:

Results	Vote date	Vote name	Relative importance
	February 20, 1994	Federal decree on the continuation of the fee on heavy vehicles traffic from June 18, 1993	100.0%
	February 20, 1994	Federal law regarding the introduction of a fee on the traffic of heavy vehicles, related either to services or to the consumption, of June 18, 1993	98.64%
	May 17, 1992	Federal law on the protection of waters (Water Protection Act) of January 24, 1991	91.54%
	September 28, 1986	Federal decree on the domestic sugar economy, change of June 21, 1985	87.96%
	February 20, 1994	Federal decree on the continuation of the national road tax	86.57%
	June 4, 1989	Popular initiative "for the protection of farms and against animal factories (Initiative for small farmers)"	86.44%
	September 27, 1998	Federal law regarding a fee on the traffic of heavy vehicle related to services (Law related to a fee on the traffic of heavy vehicles)	84.64%
	February 26, 1984	Federal decree proposed June 24, 1983 concerning the collection of a fee on the traffic of heavyweight trucks	78.36%
	June 12, 1988	Federal decree proposed March 20, 1987 concerning the amendment of the federal constitution for a coordinated transport policy	77.37%
	March 12, 1995	Federal Decree on the popular initiative 'for an environmentally sound and efficient peasant agriculture' (counter-proposal)	71.7%

Figure B.4 – List of the 10 most important votes composing each axis of the two-dimensional representation on the right side of Figure B.3. The map on the left side shows the results for each vote, and the bar on the right side indicates the relative importance of each vote in the axis.

References

[1] ArtistShare. http://www.artistshare.com. [Cited on page 31]

[2] AutoML. http://www.automl.org. [Cited on page 103]

[3] D3.js — data-driven documents. http://d3js.org. [Cited on page 107]

[4] Data portals. http://dataportals.org. [Cited on page 71]

[5] Federal Statistical Office. http://www.bfs.admin.ch. [Cited on page 72]

[6] Flask (a python microframework). http://flask.pocoo.org. [Cited on page 107]

[7] Kickstarter. http://www.kickstarter.com. [Cited on page 31]

[8] Kicktraq. http://www.kicktraq.com. [Cited on page 47]

[9] Kickstarter – Recently Launched. http://www.kickstarter.com/discover/recently-launched. [Cited on page 32]

[10] Predikon. http://www.predikon.ch. [Cited on page 107]

[11] Preference Matcher. http://www.preferencematcher.org. [Cited on page 49]

[12] Sidekick data. http://sidekick.epfl.ch/data. [Cited on pages 32 and 47]

[13] Sidekick. http://sidekick.epfl.ch. [Cited on page 105]

[14] Smartvote. http://www.smartvote.ch. [Cited on pages 49 and 51]

[15] StemWijzer. http://www.stemwijzer.nl. [Cited on page 49]

[16] Vote Compass. http://www.votecompass.com. [Cited on page 49]

[17] Vote Match Europe. http://www.votematch.net. [Cited on page 49]

[18] Wahl-O-Mat. http://www.bpb.de/politik/wahlen/wahl-o-mat. [Cited on page 49]

[19] 24 heures. Sur smartvote, certains partis aident les candidats à répondre. http://www.24heures.ch/suisse/smartvote-certains-partis-aident-candidats-repondre/story/29398477, August 2011. Accessed on 2014-05-19. [Cited on page 56]

[20] Scott Aaronson. Why philosophers should care about computational complexity. In B. Jack Copeland, Carl J. Posy, and Oron Shagrir, editors, *Computability: Turing, Gödel, Church, and Beyond*. The MIT Press, 2013. [Cited on page 1]

[21] Deepak Agarwal and Bee-Chung Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09)*, pages 19–28. ACM, 2009. [Cited on pages 81 and 98]

[22] John Agnew. Remaking italy? place configurations and italian electoral politics under the 'second republic'. *Modern Italy*, 12(1):17–38, 2007. [Cited on page 97]

[23] Ajay Agrawal, Christian Catalini, and Avi Goldfarb. Crowdfunding: Geography, social networks, and the timing of investment decisions. *Journal of Economics & Management Strategy*, 24(2):253–274, 2015. [Cited on page 47]

[24] J Scott Armstrong and Andreas Graefe. Predicting elections from biographical information about candidates: A test of the index method. *Journal of Business Research*, 64(7):699–706, 2011. [Cited on page 98]

[25] Daniel Ashbrook and Thad Starner. Using gps to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing*, 7(5):275–286, 2003. [Cited on page 28]

[26] Vitaly Belik, Theo Geisel, and Dirk Brockmann. Natural human mobility patterns and spatial spread of infectious diseases. *Physical Review X*, 1:011001, August 2011. [Cited on page 9]

[27] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, September 1975. [Cited on page 6]

[28] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *Proceedings of the 7th International Conference on Database Theory (ICDT'99)*, volume 1540 of *Lecture Notes in Computer Science*, pages 217–235. Springer, 1999. [Cited on page 6]

[29] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006. [Cited on page 21]

[30] Daniel Broderick. Crowdfunding's untapped potential in emerging markets. `http://www.forbes.com/sites/hsbc/2014/08/05/crowdfundings-untapped-potential-in-emerging-markets`, August 2014. Accessed on 2015-07-13. [Cited on page 31]

[31] CERN. Computing. `http://home.web.cern.ch/about/computing`, . Accessed 2015-08-24. [Cited on page 2]

[32] CERN. Processing: What to record? `http://home.web.cern.ch/about/computing/processing-what-record`, . Accessed 2015-08-24. [Cited on page 2]

[33] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. Friendship and mobility: User movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'11)*, pages 1082–1090. ACM, 2011. [Cited on page 28]

[34] Joshua Clinton, Simon Jackman, and Douglas Rivers. The statistical analysis of roll call data. *American Political Science Review*, 98(02):355–370, 2004. [Cited on page 68]

[35] Ronan Collobert. *Large Scale Machine Learning*. PhD thesis, Université Paris VI, 2004. [Cited on page 19]

[36] Ronan Collobert. Torch. NIPS Workshop on Machine Learning Open Source Software, 2008. [Cited on page 19]

[37] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3): 273–297, 1995. [Cited on pages 7 and 39]

[38] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967. [Cited on pages 6 and 38]

[39] Paolo Crosetto and Tobias Regner. Crowdfunding: Determinants of success and funding dynamics. Technical report, Jena Economic Research Papers, 2014. [Cited on page 47]

[40] Anthony Downs. An economic theory of political action in a democracy. *The Journal of Political Economy*, pages 135–150, 1957. [Cited on page 63]

[41] Nathan Eagle, Alex Sandy Pentland, and David Lazer. Inferring friendship network structure by using mobile phone data. *Proceedings of the National Academy of Sciences (PNAS)*, 106(36):15274–15278, 2009. [Cited on page 9]

[42] James M. Enelow and Melvin J. Hinich. *The spatial theory of voting: An introduction*. Cambridge University Press, 1984. [Cited on page 68]

[43] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research (JMLR)*, 9:1871–1874, 2008. [Cited on page 65]

[44] Federal Office of Topography. Swiss reference systems. `http://www.swisstopo.admin.ch/internet/swisstopo/en/home/topics/survey/sys/refsys/switzerland.html`. Accessed on 2015-07-27. [Cited on page 73]

[45] Federal Statistical Office. Liste historisée des communes de suisse. `http://www.portal-stat.admin.ch/gde-tool/core/xshared/gewo.php`, . Accessed on 2015-07-28. [Cited on page 73]

[46] Federal Statistical Office. Regional portraits: Communes. `http://www.bfs.admin.ch/bfs/portal/en/index/regionen/02/daten.html`, . Accessed on 2015-07-28. [Cited on pages 72 and 107]

[47] Federal Statistical Office. Votations — données détaillées. `http://www.bfs.admin.ch/bfs/portal/fr/index/themen/17/03/blank/data/01.html`, . Accessed on 2015-07-28. [Cited on pages 72 and 107]

[48] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost T. Springenberg, Manuel Blum, and Frank Hutter. Methods for improving bayesian optimization for automl. In *Proceedings of the AutoML Workshop at ICML'15*, 2015. [Cited on page 103]

[49] Jerome H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001. [Cited on pages 7, 20, and 80]

[50] Jerome H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002. [Cited on pages 20 and 80]

[51] Huiji Gao, Jiliang Tang, and Huan Liu. Mobile location prediction in spatio-temporal context. In *Proceedings of the Mobile Data Challenge by Nokia Workshop, in Conjunction with the International Conference on Pervasive Computing*, Newcastle, UK, 2012. [Cited on page 29]

[52] Diego Garzia and Stefan Marschall. Voting advice applications under review: the state of research. *International Journal of Electronic Governance*, 5(3-4):203–222, 2012. [Cited on page 49]

[53] Daniel Gayo-Avello. Don't turn social media into another 'literary digest' poll. *Communications of the ACM*, 54(10):121–128, October 2011. [Cited on page 98]

[54] Andrew Gelman and Gary King. Why are american presidential election campaign polls so variable when votes are so predictable? *British Journal of Political Science*, 23(04): 409–451, 1993. [Cited on page 97]

[55] Kostas Gemenis and Martin Rosema. Voting advice applications and electoral turnout. *Electoral Studies*, 36:281–289, 2014. [Cited on page 64]

[56] Micha Germann, Fernando Mendez, Uwe Serdult, and Jonathan Wheatley. Exploiting smartvote data for the ideological mapping of swiss political parties. In *XXVI Congress of the Italian Political Science Association*, pages 13–15, 2012. [Cited on pages 56 and 68]

[57] Sean Gerrish and David M. Blei. How they vote: Issue-adjusted models of legislative behavior. In *Advances in Neural Information Processing Systems 25*, pages 2753–2761, 2012. [Cited on page 68]

[58] Gene H. Golub and Christian Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, 1970. [Cited on page 52]

[59] Marta C. González, César A. Hidalgo, and Albert-László Barabási. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, June 2008. [Cited on pages 9 and 27]

[60] Michael D. Greenberg, Bryan Pardo, Karthic Hariharan, and Elizabeth Gerber. Crowdfunding support tools: predicting success & failure. In *Extended Abstracts on Human Factors in Computing Systems (CHI'13)*, pages 1815–1820. ACM, 2013. [Cited on pages 43, 44, and 47]

[61] Martin Ejnar Hansen and Niels Erik Kaaber Rasmussen. Does running for the same party imply similar policy preferences? evidence from voting advice applications. *Representation*, 49(2):189–205, 2013. [Cited on page 68]

[62] Sariel Har-Peled. *Geometric approximation algorithms*, volume 173. American Mathematical Society, 2011. [Cited on page 62]

[63] Mark O. Hill. Correspondence analysis: A neglected multivariate method. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 23(3):340–354, 1974. [Cited on page 53]

[64] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. [Cited on page 6]

[65] David W. Hosmer and Stanley Lemeshow. *Applied logistic regression*. John Wiley & Sons, 2004. [Cited on page 5]

[66] George Johnson. The world: In silica fertilization; all science is computer science. *The New York Times*, March 2001. [Cited on page 1]

[67] Richard Johnston, André Blais, Henry E. Brady, and Jean Crête. *Letting the people decide: Dynamics of a Canadian election*. Cambridge University Press, 1992. [Cited on page 97]

[68] Mohammad Emtiyaz Khan, Young Jun Ko, and Matthias Seeger. Scalable collaborative bayesian preference learning. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS'14)*, volume 33, pages 475–483, 2014. [Cited on page 89]

[69] Kickstarter. Kickstarter stats. `http://www.kickstarter.com/help/stats`. Accessed on 2015-07-17. [Cited on pages 31 and 33]

[70] Minkyong Kim, David Kotz, and Songkuk Kim. Extracting a mobility model from real user traces. In *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM'06)*, pages 1–13, April 2006. [Cited on page 27]

[71] Niko Kiukkonen, Jan Blom, Olivier Dousse, Daniel Gatica-Perez, and Juha Laurila. Towards rich mobile phone datasets: Lausanne data collection campaign. In *Proceedings of the ACM International Conference on Pervasive Services (ICPS'10)*. ACM, July 2010. [Cited on page 10]

[72] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009. [Cited on page 81]

[73] Venkat Kuppuswamy and Barry L. Bayus. Crowdfunding creative ideas: The dynamics of project backers in kickstarter. *UNC Kenan-Flagler Research Paper*, (2013-15), 2014. [Cited on pages 35 and 47]

[74] Andreas Ladner, Gabriela Felder, and Jan Fivaz. More than toys? a first assessment of voting advice applications in Switzerland. *Voting Advice Applications in Europe. The State of the Art*, pages 91–123, 2010. [Cited on page 64]

[75] Andreas Ladner, Jan Fivaz, and Joëlle Pianzola. Voting advice applications and party choice: evidence from smartvote users in Switzerland. *International Journal of Electronic Governance*, 5(3):367–387, 2012. [Cited on page 64]

[76] Daniel Lathrop and Laurel Ruma. *Open government: Collaboration, transparency, and participation in practice*. O'Reilly Media, Inc., 2010. [Cited on page 71]

[77] Juha K. Laurila, Daniel Gatica-Perez, Imad Aad, Jan Blom, Olivier Bornet, Trinh-Minh-Tri Do, Olivier Dousse, Julien Eberle, and Markus Miettinen. The mobile data challenge: Big data for mobile computing research. In *Pervasive Computing*. Springer, 2012. [Cited on pages 9, 10, and 11]

[78] Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In Genevieve B. Orr and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*. Springer, 1998. ISBN 978-3-540-65311-0. [Cited on page 19]

[79] Philipp Leimgruber, Dominik Hangartner, and Lucas Leemann. Comparing candidates and citizens in the ideological space. *Swiss Political Science Review*, 2010. [Cited on page 56]

[80] Yew Jin Lim and Yee Whye Teh. Variational bayesian approach to movie rating prediction. In *Proceedings of the KDD Cup and Workshop*, 2007. [Cited on page 98]

[81] Marvin L. Minsky and Seymour A. Papert. *Perceptrons - Expanded Edition: An Introduction to Computational Geometry*. The MIT Press, 1987. [Cited on page 6]

[82] Ethan Mollick. The dynamics of crowdfunding: An exploratory study. *Journal of Business Venturing*, 2013. [Cited on page 47]

[83] Nelson Morgan and Hervé Bourlard. Generalization and parameter estimation in feedforward nets: Some experiments. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 630–637. Morgan-Kaufmann, 1990. [Cited on page 19]

[84] Kevin Patrick Murphy. *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, University of California, Berkeley, 2002. [Cited on page 4]

[85] Roger B. Myerson and Robert J. Weber. A theory of voting equilibria. *American Political Science Review*, 87(01):102–114, 1993. [Cited on page 63]

[86] Ivan Noble. Human genome finally complete. *BBC News*, April 2003. [Cited on page 2]

[87] Anastasios Noulas, Salvatore Scellato, Renaud Lambiotte, Massimiliano Pontil, and Cecilia Mascolo. A tale of many cities: Universal patterns in human urban mobility. *PLoS ONE*, 7 (5):e37027, 05 2012. [Cited on page 9]

[88] Christos Papadimitriou. Algorithms, complexity, and the sciences. *Proceedings of the National Academy of Sciences (PNAS)*, 111(45):15881–15887, 2014. [Cited on page 1]

[89] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research (JMLR)*, 12:2825–2830, 2011. [Cited on pages 43, 65, and 80]

[90] Politools. Smartmap. `https://www.smartvote.ch/11_ch_nr/election/index`. Accessed 2015-08-01. [Cited on page 53]

[91] Keith T. Poole. Changing minds? not in congress! *Public Choice*, 131(3-4):435–451, 2007. [Cited on page 68]

[92] Keith T. Poole and Howard Rosenthal. A spatial model for legislative roll call analysis. *American Journal of Political Science*, pages 357–384, 1985. [Cited on page 68]

[93] Keith T. Poole and Howard Rosenthal. Patterns of congressional voting. *American Journal of Political Science*, pages 228–278, 1991. [Cited on page 68]

[94] Marian B. Pour-El and J. Ian Richards. *Computability in analysis and physics*, volume 2 of *Perspectives in Mathematical Logic*. Springer-Verlag, 1989. [Cited on page 1]

[95] Huaming Rao, Anbang Xu, Xiao Yang, and Wai-Tat Fu. Emerging dynamics in crowdfunding campaigns. In *Social Computing, Behavioral-Cultural Modeling and Prediction*, pages 333–340. Springer, 2014. [Cited on page 47]

[96] Carl Edward Rasmussen and Hannes Nickisch. Gaussian processes for machine learning (GPML) toolbox. *Journal of Machine Learning Research (JMLR)*, 11:3011–3015, 2010. [Cited on page 90]

[97] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006. [Cited on pages 84, 86, 88, and 90]

[98] Karlheinz Reif and Hermann Schmitt. Nine second-order national elections: A conceptual framework for the analysis of european election results. *European Journal of Political Research*, 8(1):3–44, 1980. [Cited on page 97]

[99] Injong Rhee, Minsu Shin, Seongik Hong, Kyunghan Lee, Seong Joon Kim, and Song Chong. On the levy-walk nature of human mobility. *IEEE/ACM Transactions on Networking (TON)*, 19(3):630–643, 2011. [Cited on page 27]

[100] Ruslan R. Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems 20*, pages 1257–1264. 2008. [Cited on pages 81 and 98]

[101] Erik Tjong Kim Sang and Johan Bos. Predicting the 2011 dutch senate election results with twitter. In *Proceedings of the Workshop on Semantic Analysis in Social Media*, pages 53–60, 2012. [Cited on page 98]

[102] Salvatore Scellato, Mirco Musolesi, Cecilia Mascolo, Vito Latora, and Andrew T. Campbell. Nextplace: A spatio-temporal prediction framework for pervasive systems. In *Pervasive Computing*, volume 6696 of *Lecture Notes in Computer Science*, pages 152–169. Springer, 2011. [Cited on page 28]

[103] Jonathon Shlens. A tutorial on principal component analysis. *Computing Research Repository (CoRR)*, abs/1404.1100, 2014. [Cited on page 75]

[104] Marko Skoric, Nathaniel Poor, Palakorn Achananuparp, Ee-Peng Lim, and Jing Jiang. Tweets and votes: A study of the 2011 singapore general election. In *Proceedings of the 45th Hawaii International Conference on System Science (HICSS'12)*, pages 2583–2591. IEEE, 2012. [Cited on page 98]

[105] Smartvote. Methodenbeschreibung — smartmap-positionskarte. `https://www.smartvote.ch/downloads/methodology_smartmap_de_CH.pdf`, June 2015. Accessed 2015-08-01. [Cited on page 53]

[106] Lindsay I. Smith. A tutorial on principal components analysis. *Cornell University, USA*, 51:52, 2002. [Cited on page 75]

[107] Chaoming Song, Tal Koren, Pu Wang, and Albert-László Barabási. Modelling the scaling properties of human mobility. *Nature Physics*, 6(10):818–823, 2010. [Cited on page 27]

[108] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, 2010. [Cited on pages 12, 27, 28, and 29]

[109] Libo Song, David Kotz, Ravi Jain, and Xiaoning He. Evaluating next-cell predictors with extensive wi-fi mobility data. *IEEE Transactions on Mobile Computing (TMC)*, 5(12): 1633–1649, 2006. [Cited on page 28]

[110] David Stern, Ralf Herbrich, and Thore Graepel. Matchbox: Large scale bayesian recommendations. In *Proceedings of the 18th International World Wide Web Conference (WWW'09)*, 2009. [Cited on page 98]

[111] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009, January 2009. [Cited on page 81]

[112] Jaakko Talonen and Mika Sulkava. Analyzing parliamentary elections based on voting advice application data. In *Advances in Intelligent Data Analysis X*, volume 7014 of *Lecture Notes in Computer Science*, pages 340–351. Springer, 2011. [Cited on page 68]

[113] Luis Terán, Jan Fivaz, and Stefani Gerber. Using a fuzzy-based cluster algorithm for recommending candidates in e-elections. *Fuzzy Methods for Customer Relationship Management and Marketing*, page 115, 2012. [Cited on page 51]

[114] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'13)*, pages 847–855. ACM, 2013. [Cited on page 103]

[115] Andranik Tumasjan, Timm O. Sprenger, Philipp G. Sandner, and Isabell M. Welpe. Predicting elections with twitter: What 140 characters reveal about political sentiment. In *Proceedings of the Fourth International AAAI Conference on Web and Social Media (ICWSM'10)*, 2010. [Cited on page 98]

[116] Twitter Developers. Frequently asked questions. `https://dev.twitter.com/faq`, . Accessed on 2015-07-13. [Cited on page 32]

[117] Twitter Developers. Twitter streaming api. `https://dev.twitter.com/streaming/overview`, . Accessed on 2015-07-13. [Cited on page 32]

[118] Kristjan Vassil. *Voting Smarter? The Impact of Voting Advice Applications on Political Behavior*. PhD thesis, European University Institute, 2011. [Cited on page 64]

[119] Michael E. Wall, Andreas Rechtsteiner, and Luis M. Rocha. Singular value decomposition and principal component analysis. In *A Practical Approach to Microarray Data Analysis*, pages 91–109. Springer, 2003. [Cited on page 52]

[120] Jingjing Wang and Bhaskar Prabhala. Periodicity based next place prediction. In *Proceedings of the Mobile Data Challenge by Nokia Workshop, in Conjunction with the International Conference on Pervasive Computing*, Newcastle, UK, 2012. [Cited on page 29]

[121] Rick Wash. The value of completing crowdfunding projects. In *Proceedings of the* 7^{th} *International AAAI Conference on Weblogs and Social Media (ICWSM'13)*, 2013. [Cited on page 47]

[122] Amy Wesolowski, Nathan Eagle, Andrew J. Tatem, David L. Smith, Abdisalan M. Noor, Robert W. Snow, and Caroline O. Buckee. Quantifying the impact of human mobility on malaria. *Science*, 338(6104):267–270, 2012. [Cited on page 9]

[123] Jonathan Wheatley. The use VAA-generated data to identify ideological dimensions: the case of Ecuador. In *Proceedings of the Second International Conference on eDemocracy & eGovernment (ICEDEG)*, pages 55–60. IEEE, 2015. [Cited on page 68]

[124] Jonathan Wheatley, Christopher Carman, Fernando Mendez, and James Mitchell. The dimensionality of the scottish political space: Results from an experiment on the 2011 holyrood elections. *Party Politics*, 2012. [Cited on page 68]

[125] Wikipedia. Ballot measure. `https://en.wikipedia.org/w/index.php?title=Ballot_measure&oldid=553302117`, 2013. Accessed on 2014-05-19. [Cited on page 50]

[126] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *Algorithmic Aspects in Information and Management*, volume 5034 of *Lecture Notes in Computer Science*, pages 337–348. Springer, 2008. [Cited on pages 88 and 98]

Vincent Etter

Route de Châtel-St-Denis 59
1807 Blonay, Switzerland
+41 79 278 20 12

Nationality: Swiss
Date of birth: June 12th, 1985
vincent@etter.io

EDUCATION

PHD IN COMPUTER AND COMMUNICATION SCIENCES, SEP 2010 – DEC 2015

Laboratory for Communications and Applications, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland

- Thesis entitled "Combine and Conquer: Mining Social Systems for Prediction"
- Co-supervised by Prof. Matthias Grossglauser and Prof. Patrick Thiran
- Mainly interested in combining models and/or datasets: worked on human mobility prediction, topic distribution of geo-localized tweets, crowdfunding success predictions, spatial analysis of political trends
- Gathered and maintained several datasets of a total of hundreds of millions of entries
- Set up a 15+ nodes Hadoop cluster and designed labs for a 50+ students undergrad course using this cluster

MASTER OF SCIENCE IN COMMUNICATION SYSTEMS, SEP 2006 – APR 2009

Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland

Master Thesis: Semantic Vector Machines, grade 6/6

GPA 5.54/6 – Courses taken include:

Cryptography and Security	Algorithms	Advanced Digital Communications
TCP/IP Networking	Security protocols and applications	Computer Vision
Distributed Information Systems	Pattern Classification and Machine Learning	Information Theory and Coding
Software-defined Radio	Models and Methods for Large-scale Random Networks	

BACHELOR OF SCIENCE IN COMMUNICATION SYSTEMS, OCT 2003 – JUL 2006

Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland

GPA 5.3/6 – Courses taken include:

Algorithmics	Stochastic Models	Principle of Digital Communications
Artificial Intelligence	Computer Networking	Object-Oriented Programming
Computer Graphics	Introduction to Information Systems	

PUBLICATIONS

JOURNAL PAPERS

- V. Etter, M. Kafsi, E. Kazemi, M. Grossglauser and P. Thiran, "Where To Go from Here? Mobility Prediction from Instantaneous Information", *Submitted to Pervasive and Mobile Computing*, 2012

CONFERENCE PROCEEDINGS

- V. Etter, J.Herzen, M. Grossglauser and P.Thiran, "Mining Democracy", *Proceedings of the second ACM conference on Online Social Networks (COSN'14)*, 2014 – **Best paper award**
- V. Etter, M. Grossglauser and P.Thiran, "Launch Hard or Go Home! Predicting the Success of Kickstarter Campaigns", *Proceedings of the first ACM conference on Online Social Networks (COSN'13)*, 2013
- V. Etter, M. Kafsi and E. Kazemi, "Been There, Done That: What Your Mobility Traces Reveal about Your Behavior", *Mobile Data Challenge by Nokia Workshop, in conjunction with Int. Conf. on Pervasive Computing*, Newcastle, UK, 2012 – **1st place winner of the Next-Place Prediction task**
- V. Etter, I. Jovanovic and M. Vetterli, "Use of learned dictionaries in tomographic reconstruction", *Proceedings of SPIE 8138*, 81381C (2011); doi:10.1117/12.894776

MASTER THESIS

- V. Etter, "Semantic Vector Machines", *Laboratory of Nonlinear Systems*, EPFL, 2009.

HONORS AND AWARDS

- Best paper award for "Mining Democracy" at COSN'14
- EPFL School of Computer and Communication Sciences Outstanding Teaching Assistant (2013)
- Winner of the Next-Place Prediction task in the 2012 Nokia Mobile Data Challenge
- EPFL I&C Department Fellowship for 1st year PhD students (2010)

WORK EXPERIENCE

RESEARCH INTERN, JUN 2013 – SEP 2013

Machine Learning Group, Microsoft Research, Redmond, WA, USA

- Implemented efficient sampling algorithms for continuous-time event models and developed a Monte-Carlo scheduling algorithm to plan future actions based on the predicted behavior of the user.
- *Related skills: .Net, Machine Learning, time-inhomogeneous Poisson processes*

RESEARCH ASSISTANT, OCT 2009 – APR 2010

Machine Learning Department, NEC Laboratories America, Princeton, NJ, USA

- Implemented parts of Torch (machine learning framework) using CUDA to run on GPU, in order to scale to even bigger datasets for unsupervised learning (currently using Wikipedia – 650 millions of words).
- *Related skills: CUDA, Lua, Machine Learning*

RESEARCH ASSISTANT, SEP 2008 – MAR 2009

Machine Learning Department, NEC Laboratories America, Princeton, NJ, USA

- Master thesis in Machine Learning
- Using torch5 (http://torch5.sourceforge.net), a Machine Learning framework written in Lua, studied the use of neural networks for machine translation and semantic compression, and developed a novel architecture for embedding sentences of arbitrary length into a higher dimensional space
- *Related skills: Lua, Machine Learning, neural networks, language models*

CO-OWNER AND LEAD DEVELOPER, 2003 – 2014

Dootix Sàrl, Lausanne, Switzerland

- Developed and maintained dynamic websites, as well as an online interface for servers and clients management
- Helped for mail and web servers administration
- *Related skills: PHP, XHTML, CSS, AJAX, MySQL, Linux administration (Debian)*

SUPERVISED STUDENT PROJECTS

- Predicting Vote Outcomes: the Impact of Imperfect Sampling – Spring 2015
- Hybrid Indoor Localization using Smartphones - Spring 2014
- Scraping and Mining Political Data from www.admin.ch - Spring 2014
- Analyzing the Social Graph of Kickstarter - Spring 2014
- Learning Topic Models from Geo-Tagged Tweets - Autumn 2013
- Visualization of Campus Mobility - Spring 2013
- Multiscale Mobility Prediction - Spring 2013
- MapReduce for Data Mining and Visualization - Autumn 2012
- Analysis of Political Data - Spring 2012

TECHNICAL KNOWLEDGE

- Machine Learning, Data Mining, Information Retrieval
- Hadoop (scalable distributed computing), Condor (high-throughput computing)
- Python, Java, .Net Framework (C#, ASP.Net), C/C++, Lua (Torch), CUDA
- PHP (Zend Framework), XHTML, JavaScript (jQuery), CSS, AJAX
- MVC, eXtreme Programming, Unit Testing, Version Control (Git, Subversion)
- MySQL, SQLite, PostgreSQL, SQL Server
- Eclipse, Visual Studio, Zend Studio, LaTeX, Vim
- Windows, Mac OS X, *nix (Debian, Ubuntu)

LANGUAGE SKILLS

- French (mother tongue)
- English (fluent)
- German (High School knowledge)
- Spanish (basic notions)