Preliminary models for predicting the political parties and election outcome of the Finnish parliamentary elections

T-61.3050 final report

Etienne Thuillier (409261)
etienne.thuillier@aalto.fi

Robin Rajamäki (81970E)
robin.rajamaki@aalto.fi

November 17, 2013

Abstract

This preliminary study describes the application of some simple classification algorithms in a political party and election outcome prediction problem. The implemented methods were first validated and optimized using techniques such as K-fold cross-validation, stratification and forward feature selection. After this, their performance was evaluated using metrics such as F-score and accuracy. The final model used for prediction of the political party was based on normally distributed classes with Naive Bayesian assumptions and maximum likelihood parameters. The generalization F-score/accuracy of this model was 0.47/0.96 on the test set. Mainly due to the heavily skewed nature of the data, it was found that none of the implemented classifiers clearly beat the single-feature dummy function (0.48/0.91) in the election outcome task, although the logistic discriminant using gradient ascent showed some improvement in terms of accuracy (0.44/0.92).

1 Introduction

Election forecast is a relatively challenging machine learning problem given the scarceness of the data: an election indeed occurs only once every few years and political surveys are often significantly biased. In particular, voters may feel uneasy about honestly expressing their political views under scrutiny of a interviewer [1].

Thus there exists a need for a different approach for investigating election processes, that does not depend on public opinion polls. Such investigations, with help of modern machine learning methods, could indeed lead to a better understanding of the election process. To the least, they could provide a complementary viewpoint.

The performance of simple preliminary models for predicting the the outcome of the 2011 Finnish parliamentary elections (1) is studied in this paper. Moreover, this study discusses the performance of these models in inferring the party

memberships of each candidate in a first data subset, given known party labels of a second subset.
The discussed models are mainly parametric. Indeed, the parameter values that is inferred from the data set with such model can provide information on the election process. Analysis of such information is left to specialists in politics.
The study is divided into four parts. In the first section, the theory behind the methods used in this paper are summarized. The second section presents the application of these methods to the classification problems mentioned above. The third section presents the results obtained for each of the studied models. Finally, the fourth section discusses the performance of each studied model in terms of validation and generalization performance with regards to an unseen data set (hereafter called the test data set).

Table 1: Official count of the number of candidates and number of seats in ballot for the 2011 Finnish parliamentary elections, derived from [2].

Constituency	# of Candidates	# of Seats
South Savo	94	6
Helsinki	267	21
Häme	132	14
Central Finland	141	10
Kymi	118	12
Lapland	114	7
Oulu	156	18
Pirkanmaa	207	18
North Karelia	108	6
North Savo	118	9
Satakunta	97	9
Uusimaa	406	35
Vaasa	158	17
Varsinais-Suomi	191	17
Åland	8	1

2 Methods

2.1 Multivariate Gaussian class distributions

As mentioned in section 5.1, the problems with the given data set suggest that parametric learning methods have potentially the best chances at succeeding in the classification tasks. Perhaps one of the simplest parametric classifiers is obtained by assuming normally distributed classes. In case of a data set $\mathcal{X}$ of dimensions $[N\ x\ d]$ this means that each class is modeled as a d-dimensional Gaussian, whose discriminant lies on the hyperplane intersecting with the neighboring class distributions. Knowing the parameters which fully describe the distributions allows estimating the class of a unknown test point. This is simply done by selecting the most probable class using the aforementioned hyperplane as the decision boundary.

The multivariate normal distribution [3] is given by

$$p(\mathbf{x}) = \frac{1}{2\pi^{0.5d}|\mathbf{\Sigma}|^{\frac{1}{2}}} e^{-0.5(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}. \tag{1}$$

Eq. 1 shows that the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\mathbf{\Sigma}$ fully characterize the multivariate normal distribution. Since the final posterior probability of an observation belonging to class C_i is given by Bayes' rule

$$P(r^t = C_i|\mathbf{x}^t) = \frac{P(\mathbf{x}^t|r^t = C_i)P(r^t = C_i)}{P(\mathbf{x}^t)}, \tag{2}$$

one needs to know the class prior probabilities in addition to the parameters of the normal distribution in order to be able to evaluate the discriminant function

$$g(\mathbf{x}) \propto P(\mathbf{x}^t|r^t = C_i)P(r^t = C_i). \tag{3}$$

2.1.1 Maximum likelihood parameter estimation

A central task of any parametric method is to find the model's parameters. One way to do this is to find the maximum likelihood (ML) estimates, which conveniently have analytical expressions for the multivariate normal distribution. The ML parameters are found by maximizing the likelihood, or log likelihood function, which expresses the likelihood the data given a certain set of parameters ($\boldsymbol{\theta}$) of the assumed model. Assuming N i.i.d. observations, the log likelihood function can be written as

$$\mathcal{L}(\boldsymbol{\theta}|\mathcal{X}) = \sum_{t=1}^{N} log(p(\mathbf{x}^t|\boldsymbol{\theta})). \tag{4}$$

The ML parameters (in this case $\boldsymbol{\theta}$ consists of $\boldsymbol{\mu}_i$, $\mathbf{\Sigma}_i$ and $P(C_i)$) are found by taking the derivative of Eq. 4 w.r.t. each parameter and setting the resulting expression to zero. Under the assumption of normally distributed classes the ML estimates for class i become [3]

$$\begin{cases} \hat{\boldsymbol{\mu}}_i = \frac{\sum_t r_i^t \mathbf{x}^t}{\sum r_i^t} \\ \hat{\mathbf{\Sigma}}_i = \frac{\sum_t r_i^t (\mathbf{x}^t - \boldsymbol{\mu}_i)(\mathbf{x}^t - \boldsymbol{\mu}_i)^T}{\sum r_i^t} \\ \hat{P}(C_i) = \frac{\sum_t r_i^t}{N}. \end{cases} \tag{5}$$

Eq. 5 assumes that $r_i^t = 1$ when $x^t \in C_i$, and $r_i^t = 0$ else.

2.1.2 Model variations

The model outlined in the previous sections can be varied a few ways using further simplifying assumptions. The most straightforward discriminant function is obtained by directly substituting the ML estimates on Eq. 5 into Eq. 3. This yields [3]

$$g_i(\mathbf{x}) = -0.5log(|\mathbf{\Sigma}_i|) - 0.5(\mathbf{x} - \hat{\boldsymbol{\mu}}_i)^T \mathbf{\Sigma}_i^{-1}(\mathbf{x} - \hat{\boldsymbol{\mu}}_i) + \hat{P}(C_i). \tag{6}$$

The matrix inversion and discriminant computation of Eq. 6 are very sensitive to ill-conditioned (read: real-life) data. If the modeled data is only approximately normally distributed, it makes sense to make further simplifications to the model. One alternative described in [3] is to calculate a weighted common covariance matrix using the class priors probabilities and covariance matrices:

$$\hat{\mathbf{\Sigma}} = \sum_i \hat{P}(C_i)\hat{\mathbf{\Sigma}}_{\boldsymbol{i}}. \tag{7}$$

The discriminant function then becomes linear [3]:

$$g_i(\mathbf{x}) = (\mathbf{\Sigma}^{-1}\hat{\boldsymbol{\mu}}_i)^T\mathbf{x} - 0.5\hat{\boldsymbol{\mu}}_i^T\mathbf{\Sigma}^{-1}\hat{\boldsymbol{\mu}}_i + log(\hat{P}(C_i)). \tag{8}$$

Alternatively, the common covariance matrix can be assumed to be diagonal, which implies independent covariates in the case of the normal distribution. This leads to naive Bayes' classifier [3]

$$g_i(\mathbf{x}) = -0.5\sum_{j=1}^{d}(\frac{x_j^t - \hat{\boldsymbol{\mu}}_{j,i}}{\sigma_{j,i}})^2 + log(\hat{P}(C_i)). \tag{9}$$

One can see that in case of unit variance covariates ($\sigma_{j,i} = 1$), the squared expression in Eq. 9 becomes the Euclidian distance. Also, if the priors $\hat{P}(C_i)$ are uniform, Eq. 9 reduces to the nearest mean classifier [3]

$$g_i(\mathbf{x}) = ||\mathbf{x} - \boldsymbol{\mu}_i||^2. \tag{10}$$

2.2 Logistic discrimination

A discriminative model such as the logistic discriminant (using gradient ascent) avoids some of the problems produced by generative models such as the Gaussian ML of section 2.1. It has been argued that discriminative models are robuster on small data sets, since the assumptions associated with them are not as strong as for generative models [3].

In logistic discrimination, rather than inspecting the class-conditional distributions separately, their ratio is modeled. In the binary case the discriminant function can be directly derived from the probability

$$\hat{P}(C_0|\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w}^T\mathbf{x} + w_0)}}, \tag{11}$$

where $\mathbf{w}$ and w_0 can be viewed as model parameters. Under specific assumptions, e.g. maximum likelihood expressions can be derived for $\mathbf{w}$ and w_0. However, when viewing Eq. 11 as a purely discriminative model one can attempt

to directly estimate the model parameters without any generative assumptions. This can for instance be done using the gradient ascent algorithm, which iteratively attempts to find a local maximum by incrementing the parameters of the function in the opposite direction of the gradient. The details of the gradient ascent algorithm are beyond the scope of this paper, so the interested reader is referred to [3] for more on the subject.

2.3 k-Nearest Neighbors

Non-parametric methods are particularly useful when no prior assumption can be made with regards to the underlying distribution of the problem to be solved. A non-parametric method was implemented in this study in part to provide a baseline to compare the parametric methods described in section 2.1 and 2.2 to. k-NN is a method that is easy to implement, which is why it was chosen for this task.

The k-NN is a extension of the Nearest Neighbor method in which the predicted class of an input element is set to the the class value of the closest element of a training data set. In practice:

1. A distance, for example euclidian, between each element of the training dataset and the input element is calculated
2. Identification of the closest element is performed by a search of the smallest distance
3. The class label of the closest training set item is outputted.

In the k-NN algorithm the search step (step 2) is performed to identify the set of the k elements of the training data set that are the closest to the input element. Moreover, the class value that is most frequent among the k closest elements is outputted as the predicted class value for the input element.

In k-NN, the identification of the most frequent class in the surroundings of the input element allows, at least partly, to mitigate the effects of noise. Feature selection, as described in section 2.6, is used to further decrease the influence of noise.

2.4 Metrics

The prediction result of any classifier belongs to one of four elementary categories of the confusion matrix, which is a well established concept of estimation theory. The estimate is said to be a true positive (TP) if it was correctly identified to belong to class C_i and true negative (TN) if it was correctly identified not to belong to C_i. Analogously, the errors the estimator can make are divided into false positives (FP) and false negatives (FN). More descriptive metrics can obviously be derived using these four basic categories. The most common of these include accuracy, sensitivity, precision (a.k.a. positive predictive value) and false positive rate, which all describe different performance aspects of the classifier. Perhaps the most useful measure is given by the F-score (more precisely the F_1-score), which is an attempt at compressing the overall performance into a single metric. The metrics used in this paper along with their descriptions are given in Tab. 2. For the sake of comprehensibility, the formulas in the table

apply only for the binary classification task. The respective expressions for the multi-class problem are conceptually the same, but slightly more complicated due to the fact that the errors (FP and FN) are no longer as simple to define as in the binary case. For an in-depth review of different (multi-class) classification metrics, the interested reader is referred to [4].

Table 2: Evaluation metrics and descriptions. The number of estimates belonging to each category is marked by capital letters (i.e. $N = TP+TN+FP+FN$).

Metric	**Formula**	**Description**
Accuracy	$\frac{TP+TN}{N}$	Fraction of correct classifications
Precision	$\frac{TP}{TP+FP}$	Confidence in the correctness of the classification
Sensitivity	$\frac{TP}{TP+FN}$	Sensitivity to false estimates
FP rate	$\frac{FP}{FP+TN}$	False positive rate
F-score	$2\frac{Precision \cdot Sensitivity}{Precision+Sensitivity}$	"overall performance"

2.5 K-fold cross-validation

Since the test set should only be used for generalization error estimation of the final classifier, model order / optimal feature set selection must be performed on the training data. An effective way of avoiding strong training set bias in this decision process is to divide the training set into two separate parts. One set acts as training data, whereas the generalization performance of the classifier is evaluated using the other set (i.e. the validation set). An even more robust alternative is to do this many times using different (e.g. random) divisions of the training data and averaging over all trials. K-fold cross-validation is such a method, where the training data is divided into K subsets of approximately equal size. The classifier is then trained using $K-1$ of these subsets and evaluated using the remaining subset. This is repeated K times and the performance of the classifier is obtained as the average performance over these K trials.

If the class frequencies are nonuniformly distributed (as in the case of this data set), it may be useful to assure that each of the K subsets used for cross-validation have approximately the same proportions of class occurrences. If this is not the case, the classification performance becomes unstable due to possible data mismatches between the training and validation sets. For example, with bad luck the training data may not contain a single instance of a certain class, which obviously makes it impossible to correctly classify that class when using supervised learning methods. Ensuring equal class distributions between subsets is called stratification [3]. Due to the heavily skewed distributions of both party and election results, stratification was used along with K-fold cross-validation in the classification tasks presented in this paper.

2.6 Model order and feature selection

As stated in section 2.5, the training set should be used for selecting the order and used features of the model under inspection. In practice, the optimal model order can be selected through K-fold cross validation by choosing the complexity

which minimizes the validation error. Alternatively, one (or many) of the metrics presented in section 2.4 (e.g. the F-score) can be used as the criterion to be maximized or minimized. A reasonably good feature subset can be found in a similar manner by e.g. forward feature selection. In forward feature selection one starts with an empty feature subset, to which each feature not already in the subset is iteratively added to. The performance of the feature subset is then calculated (e.g. using cross-validation), where after the new feature is removed and the next one added to the subset. The feature yielding the best subset performance is permanently added to the set of selected features. The inverse procedure, that is starting from the full feature set and eliminating the worst features one-by-one is called backward feature selection [3]. In practice the two yield identical results in equal time, provided the entire feature set is evaluated. Through feature selection, one can both decrease the model complexity and improve the performance of the classifier.

3 Experiments

3.1 Pre-processing

The data set consisted of 1563 data vectors of 199 covariates. The data was divided into a training and test set of size 1037 and 526 entries respectively. In addition, each covariate was normalized to zero-mean and unit variance for compatibility reasons.

3.2 Validation and evaluation

The performance of all classifiers was evaluated using the metrics presented in section 2.4. Most emphasis was given to the F-score, which was generally used for decision making in the evaluation. For example model order, used features and final models were all chosen to maximize the F-score.

A 10-fold cross validation scheme with stratification was employed in the experiments. $K = 10$ is a common choice in machine learning tasks [3], although it in this case further reduces the amount of training data in a an initially small data set. On the other hand, the number of features is large in comparison to the number of observations already to begin with.

The cross-validation scheme described above was used to train each of the studied models, both in terms of model complexity and feature selection. It was also used in a later step for comparing the performance of the models on a common ground, i.e. using exactly the same K-fold segmentation for the validation of all models. This was done in order to avoid any bias in favor of one model over the others, which could occur from a favorable segmentation of the data set.

In this study, forward selection was used. Obviously, this step cannot be be undertaken without first defining the complexity of the model. Inversely, this also holds for complexity selection which depends on the selected feature set. Thus, both of these meta parameters should ideally be searched for simultaneously. An exhaustive search through all possible feature and model order permutations would however be highly impractical due to the exponentially increasing computational cost. Thus these two optimization tasks were performed separately. Although there is no guarantee of finding globally optimal parameters this way, a too pedantic fine-tuning of the model's parameters could lead to overfitting the model to the training data.

As previously mentioned, the models are compared in this study following their tuning in the feature and complexity selection steps. This allows selecting the model with the best performance to predict election outcome and party membership for candidates of the test data set.

To summarize, the final experiment procedure was roughly as follows:
For each classifier

- find the model order yielding the highest F-score in validation
- find the feature set yielding the highest F-score in validation.

Once the above steps are completed

- select the model yielding the highest F-score in validation as the final classifier

- evaluate the generalization performance of the final classifier.

In practice it was found that varying the order of the first two steps sometimes gave better validation results. Also, for reasons discussed in section 5.3 the generalization performance was actually evaluated for all optimized classifiers, instead of only the best one of the final validation round.

3.3 Parameters of the logistic discrimination

Since the gradient ascent algorithm was utilized in the case of the logistic discriminant, some parameters needed to be specified to assure the convergence of the algorithm. First of all, gradient ascent needs an initial starting point or "guess" for its internal parameters $\mathbf{w}$ and w_0 (see Eq. 11). A common way to do this is to initialize by zeros [3], which was also done in this study. Secondly, the step size has to be set. In practice, the adequateness of the selected step size can only be determined by checking if the algorithm converges for the particular problem or not. A step size of 0.01 was found lead to convergence in all tested cases of this study. Finally, the algorithm needs a stopping condition. Usually this is selected to be a maximum error between iterations, a maximum number of iterations or a combination of both. The maximum number of iterations was left as the optimizable model parameter, since it roughly controls how exactly the data is fitted. This gives an additional means to avoid possible over-/underfitting of the model to the training data.

3.4 Implementation particularities for the k-NN classifier

It was found that executing feature selection before complexity selection proved critical in the case of the k-NN method, which is very sensitive to noise. Such noise primarily results from features that are insignificant with regards to the problem at hand. The feature selection algorithm is thus executed first in the case of k-NN. This required taking into account all considered values of k. In order to keep the calculation time in a practical limit, the F-score was calculated at each iteration of the forward feature selection method for a large set of predictions. These predictions comprised of a prediction for each considered value of k and element of the validation data set. Thus the feature selection could be undertaken in a somewhat agnostic fashion with regard to the model complexity. Following this step, a complexity was selected starting from the data set with a pruned feature set. Finally, the feature selection was repeated specifically for the selected complexity value in order to provide a refined selection features.

The k-NN method was implemented throughout the study using MATLAB's built in *knnsearch* function which outputs a set of indexes designating the nearest neighbors in the training data set for each input element.

4 Results

In this section the results of each prediction task are presented. Both the political party and election outcome sections begin with model order and feature selection of each model. After this the optimized models are compared with each other in a common validation. The best classifier is then selected for each tasks, and its generalization error is evaluated on the test set.

4.1 Political party

Tab. 3 shows the average performance of the Gaussian ML classifier on the party prediction task using all available features and the different model assumptions discussed in section 2.1.2. Clearly, the most general model assumption (i.e. the most complex discriminant) with class-specific covariance matrices (Eq. 6) gives the worst overall performance. Gradual model simplifications seem to improve the overall performance until the nearest mean classifier. The best performance on all metrics is given the naive Bayes classifier (Eq. 9) with a common unit covariance matrix (denoted "Naive Bayes II" in Tab. 3). This leads to it being chosen as the suitable model complexity for the Gaussian ML estimator.

Table 3: Model order selection of the Gaussian ML using all features for the party prediction task.

	F-score	**Accuracy**	**FP rate**	**Sensitivity**	**Precision**
Perfect	1	1	0	1	1
Original	0.003	0.874	0.065	0.065	0.002
Linear discriminant	0.255	0.916	0.045	0.275	0.32
Naive Bayes I	0.432	0.947	0.029	0.445	0.45
Naive Bayes II	0.437	0.949	0.028	0.449	0.456
Nearest mean	0.438	0.948	0.028	0.454	0.452

Fig. 4.1 shows the results of the forward feature selection using the Gaussian ML model obtained in the previous paragraph. It is hard to reliably say that the performance of the classifier is better for any feature set when looking at the graph. In fact, it was found that the performance of the classifier when using the reduced feature set (31 features) was worse than when using all (199) features. It was thus decided to disregard any feature selection and keep the data intact for the political party estimator.

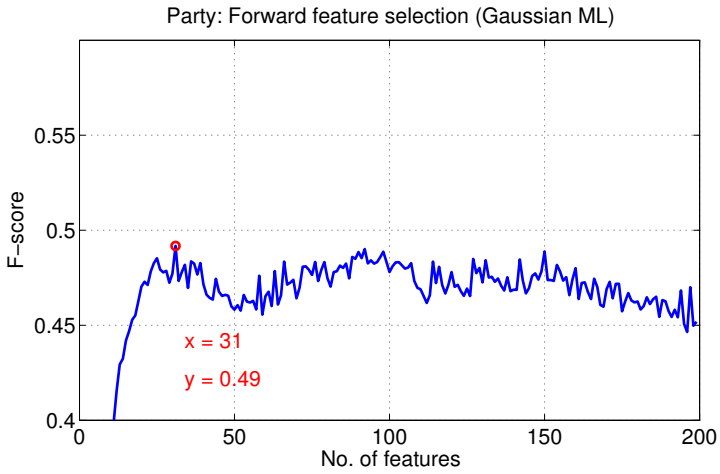

Figure 1: Forward feature selection using the Gaussian ML classifier for the party prediction task. The red circle and text specify the optimal feature set (length and F-score).

The generalization performance of the final political party classifier is shown in Tab. **??** and 5. The former displays the party-wise performance on the test data, whereas the latter shows essentially the same but averaged over all classes, along with the performance of the perfect classifier and some dummy methods. The analysis and discussion of the final classifier is saved for section 5.2.

Table 4: Party specific generalization performance of the Gaussian ML classifier.

	F-score	**Accuracy**	**FP rate**	**Sensitivity**	**Precision**
IPU	0.296	0.964	0.021	0.333	0.267
KA	0	0.994	0	0	0
KD	0.473	0.907	0.052	0.478	0.468
KESK	0.677	0.924	0.047	0.7	0.656
KOK	0.718	0.937	0.026	0.667	0.778
KTP	0	0.989	0.006	0	0
M2011	0.667	0.985	0	0.5	1
PIR	0.743	0.983	0.004	0.65	0.867
PS	0.85	0.968	0.023	0.889	0.814
RKP	0.596	0.964	0.03	0.778	0.483
SDP	0.75	0.935	0.043	0.785	0.718
SEN	0	0.989	0	0	0
SKP	0.5	0.947	0.028	0.5	0.5
STP	0	0.964	0.027	0	0
VAS	0.557	0.903	0.055	0.561	0.552
VIHR	0.672	0.924	0.039	0.651	0.695
VP	0.444	0.99	0	0.286	1

Table 5: Average generalization performance on test data.

	F-score	**Accuracy**	**FP rate**	**Sensitivity**	**Precision**
Random	0.037	0.888	0.06	0.044	0.046
LP dummy	0.013	0.897	0.059	0.059	0.007
Perfect	1	1	0	1	1
Naive Bayes II	0.467	0.957	0.024	0.457	0.517

4.2 Election outcome

4.2.1 Gaussian class distributions with ML parameters

The results of the Gaussian ML model order selection procedure for the election results task are displayed in Tab. 6. For largely the same reasons as in section 4.1, the Naive Bayesian with a common unitary covariance matrix (Naive Bayes II) prevails once again as the winning model in validation. However, contrary to the model used for party prediction, the Gaussian ML shows a clear decrease in performance for increasing feature set length (Fig. 4.2.1). The forward feature selection suggests a final feature set of length 9, although when inspecting Fig. 4.2.1 it could arguably be any combination of the 50 or so first features picked by the method. In order to keep the final Gaussian model as simple as possible, it was decided to use the Naive Bayes II assumption and the 9 features suggested by forward feature selection.

Table 6: Model order selection of the Gaussian ML using all features for the election results prediction task.

	F-score	**Accuracy**	**FP rate**	**Sensitivity**	**Precision**
Perfect	1	1	0	1	1
Original	0.145	0.078	1	1	0.078
Linear discriminant	0.208	0.703	0.279	0.499	0.132
Naive Bayes I	0.348	0.855	0.115	0.494	0.272
Naive Bayes II	0.389	0.864	0.109	0.549	0.306
Nearest mean	0.343	0.806	0.18	0.64	0.236

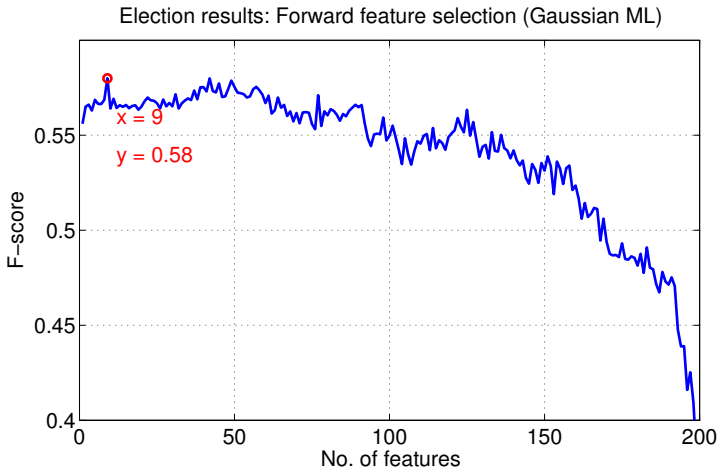

Figure 2: Forward feature selection using the Gaussian ML classifier for the election outcome prediction task. The red circle and text specify the optimal feature set (length and F-score).

4.2.2 Logistic discrimination

Contrary to the Gaussian ML, the logistic discriminator showed a better performance when the model order and feature selection steps were inverted. The model order in this case was the number of iterations of gradient ascent algorithm (N_{iter}). Experimentation showed that $N_{iter} = 100$ was a suitable value for the feature selection. The forward feature selection F-score curve for $N_{iter} = 100$ is shown in Fig 4.2.2. The decrease in the F-score with increasing feature set length is not as exaggerated as in Fig. 4.2.1, however arguably not as ambiguous as in Fig 4.2.1. The feature set was thus chosen as the one of length 35 maximizing the F-score in Fig. 4.2.2.

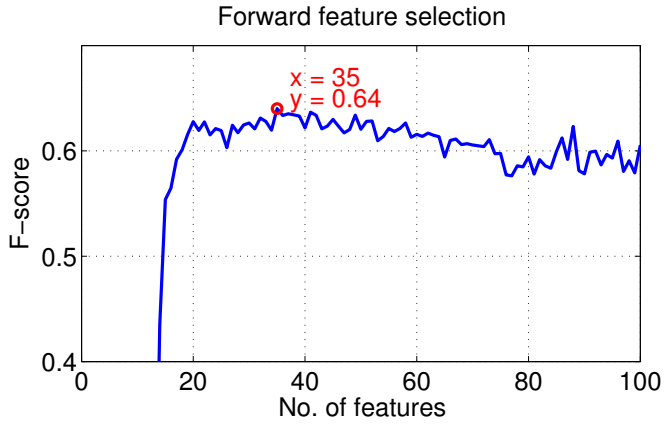

Figure 3: Forward feature selection using the logistic discriminator for the election outcome prediction task. The red circle and text specify the optimal feature set (length and F-score).

The subsequent model order selection is displayed in Fig. 4.2.2. The F-score converges to ≈ 0.6, due to the order in which the feature and model order selections were performed. Fig. 4.2.2 suggests however that $N_{iter} = 100$ is a unnecessary large parameter value for ensuring convergence of the discriminant function. This is why $N_{iter} = 50$ was selected as the final model order.

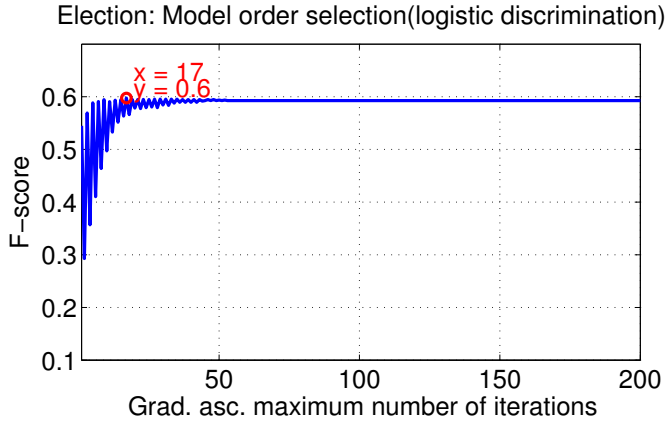

Figure 4: Model order selection of the logistic discriminator using the features set specified by Fig. 4.2.2 for the election outcome prediction task.

4.2.3 k-NN

The graph of figure 4.2.3 depicts the result of the preliminary feature selection step as described in section 3.4. As indicated in the graph, 42 features are retained as significant following this step.

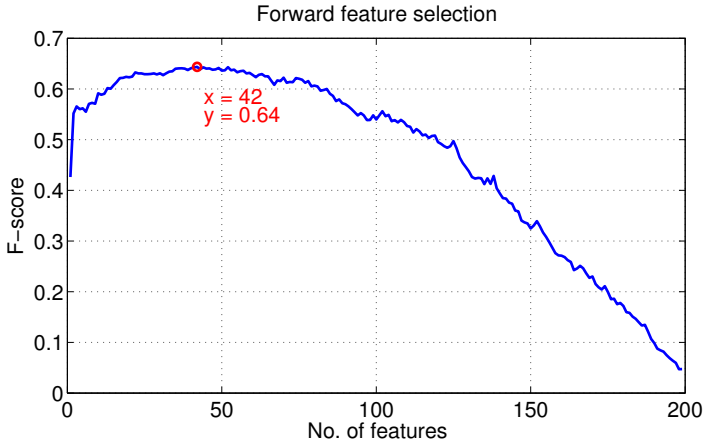

Figure 5: Preliminary feature selection step for the k-NN (election outcome prediction task). The red circle and text specify the optimal feature set (length and F-score).

The graph of figure 4.2.3 represents the F-score achieved by the k-NN model for different values of the complexity parameter k. High values of this parameter provide increasing robustness with regards to noise [3]. An excessively high value of k leads to underfitting and thus to a drop in performance, as made obvious in the right part of the graph. The complexity value of 34 is chosen as it is located at the inflection point of an elbow in the curve, after which the performance of the model drops with increasing k.

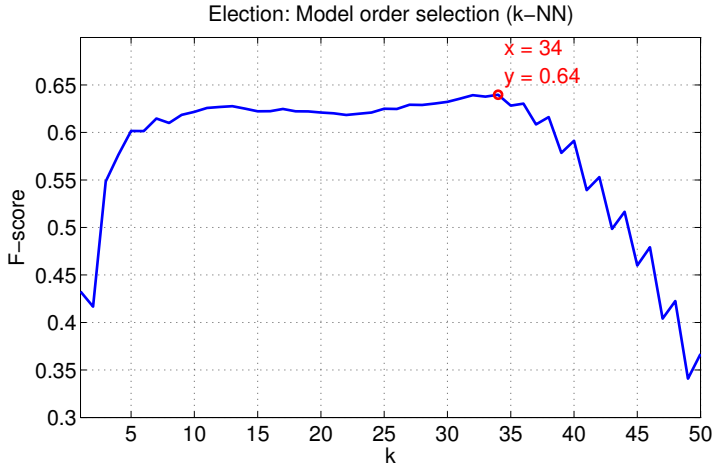

Figure 6: Complexity parameter selection of the k-NN classifier (election results prediction task).

As described in section 3.4, another run of the feature selection algorithm is carried out in order to refine the selection of features specifically for the chosen

value of the complexity parameter k. As depicted in the graph of Fig. 4.2.3, 60 features are selected to form the definitive reduced feature set for the k-NN model. It is to be noted that a slight increase of the F-score is observed for this set, although the validation performance in Tab. 7 is slightly lower. This is caused by the differences between the k-fold segmentation of the refinement feature selection step and that of the comparative evaluation of the models as discussed in section 3.2.

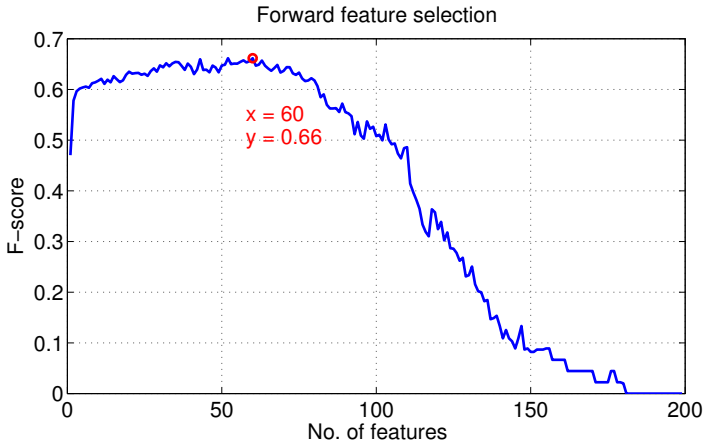

Figure 7: Refinement feature selection step for the k-NN (election results prediction task). The red circle and text specify the optimal feature set (length and F-score).

4.2.4 Final model selection

The validation performance of the three final classifiers is shown in Tab. 7. According to the results, k-NN beats both the Gaussian and logistic classifiers on almost all metrics. In fact, only the Gaussian ML has a slightly higher sensitivity than the k-NN. Following the same rationale as in the model order selection, Tab. 7 indicates that k-NN should be selected as the final election outcome predictor.

Table 7: Final model selection for the election prediction task.

	F-score	**Accuracy**	**FP rate**	**Sensitivity**	**Precision**
Perfect	1	1	0	1	1
Naive Bayes II	0.549	0.929	0.041	0.569	0.564
Logistic discr.	0.593	0.946	0.018	0.525	0.737
k-NN	0.61	0.951	0.013	0.524	0.783

Reality is however seldom as simple as the reasoning of the previous paragraph would suggest. As already hinted in section 3.2, this fact becomes apparent when evaluating the generalization performance of all three final classifiers. Tab. 8 shows that the performance order proposed by the validation in Tab.

7 is actually inverted when evaluating on the test set. Tab. 8 shows that the Gaussian ML with naive Bayesian assumptions has the best F-score and sensitivity of the three, but slightly worse accuracy, false positive rate and precision than the other two. In fact, the Gaussian model has an effectively identical generalization performance as the MOP dummy classifier, where a candidate is predicted to get elected if he or she previously was a member of parliament. Tab. 8 also contains two additional dummy methods for reference. The random dummy simply selects either class by equal chance, whereas the AZ (all zeros) dummy always selects the "not elected" class. The mismatch in the relative frequencies of the two classes (Fig. 5.1) actually leads to the method having the best accuracy of the dummy approaches.

Table 8: Generalization performance on test data.

	F-score	**Accuracy**	**FP rate**	**Sensitivity**	**Precision**
Random	0.097	0.471	0.518	0.349	0.057
AZ dummy	0	0.918	0	0	0
MOP dummy	0.477	0.913	0.05	0.488	0.467
Perfect	1	1	0	1	1
Naive Bayes II	0.477	0.913	0.05	0.488	0.467
Logistic discr.	0.444	0.924	0.027	0.372	0.552
k-NN	0.442	0.918	0.035	0.395	0.5

5 Discussion

5.1 Preliminary discussion on the data sets

It to be noted that the prediction problems to be solved are somewhat different from that of predicting elections from opinion poles. In particular, there is no feature in our data set that expresses a percentage of voting intentions for each candidate. The success or failure of each candidate is therefore not inferred from knowledge of such voting intentions, but on the known election outcome (elected or not elected) of other candidates in the election. In other words, this election problem can be thought as determining combinations of features that promotes election of the candidates. This raises two specific difficulties in the current problem compared to opinion-poll based election predictions.

Firstly there is no possibility of modeling the deterministic part of the election process, i.e. carrying out D'Hondt's method. This methods is indeed used in Finland among other nations to distribute the number of seats in ballot in each constituency among parties and coalitions, according to their scores (i.e. votes) [5]. Since this process remains unmodeled, it introduces additional error compared to classic election prediction. Moreover, this makes it complicated to make outcome predictions for the candidates that are globally consistent with the number of parliamentary seats. Indeed, the outcome of the election is predicted candidate by candidate such that the resulting total number of elected representatives in the prediction can be made to exceed that by the actual number of available seats in parliament. In other words, the potential of each candidate being elected depends on his or her intrinsic value without comparison to their potentially stronger opponent fighting for the same seats. This is problematic when there is a lot of strong candidates but with few sets left in a county.

Secondly, the data set sorted according to the election outcome (elected - not elected) is heavily skewed. For each winning candidate there is an average of about 12 loosing candidates (Fig. 5.2, Tab. 9 and 1). This required special care (e.g. stratification) when training the model, as was seen in section 2.5.

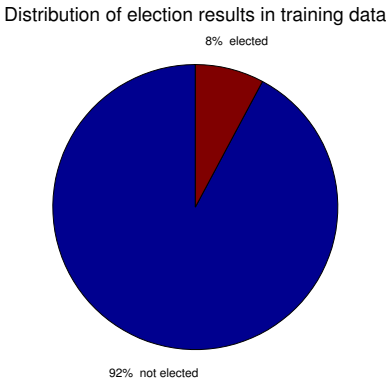

Figure 8: Training data class distribution in the election outcome prediction task.

Table 9: Number of candidate entries in the training data set: overall count and elected candidates (i.e. seats) in absolute value and percentage of the official count presented in Tab. 1.

Constituency	# of Cand.	%	# of Seats	%
South Savo	43	46	5	83
Helsinki	135	51	9	43
Häme	58	44	8	57
Central Finland	63	45	6	60
Kymi	59	50	5	42
Lapland	46	40	1	14
Oulu	74	47	7	39
Pirkanmaa	85	41	6	33
North Karelia	42	39	2	33
North Savo	48	41	2	22
Satakunta	50	52	2	22
Uusimaa	192	47	15	43
Vaasa	59	37	5	29
Varsinais-Suomi	83	43	8	47
Åland	0	0	0	0

Table 10: Number of candidate entries in the test data set: overall count and elected candidates (i.e. seats) in absolute value and percentage of the official count presented in table of table 1.

Constituency	# of Cand.	%	# of Seats	%
South Savo	22	23	0	0
Helsinki	56	21	3	14
Häme	35	27	2	14
Central Finland	33	23	1	10
Kymi	29	25	1	8
Lapland	20	18	2	29
Oulu	35	22	5	28
Pirkanmaa	42	20	6	33
North Karelia	25	23	1	17
North Savo	25	21	0	0
Satakunta	19	20	5	56
Uusimaa	103	25	7	20
Vaasa	38	24	4	24
Varsinais-Suomi	44	23	6	35
Åland	0	0	0	0

5.2 Party prediction

Due to the low observation count of the data set and other problems discussed in section 5.1, it makes sense that the final model complexity is rather simple. This is especially true remembering that that the model order selection was done using all covariates, as explained in section 4.1. When taking into account

the relative sizes of the different parties (see Fig. 5.2), it seems (Tab. 4) that the selected Gaussian model works better for larger than smaller parties. This is not entirely true when comparing e.g. the "PS" and "KOK", where the former has better a better generalization performance (F-score: 0.85 vs. 0.72), although it has a relatively smaller presence in the training data. On the other hand, the smallest parties ("KA","KTP","KTP","SEN" and "STP") clearly have the worst performance according to Fig. 8. This behavior is to be expected, as classes with very few members are always challenging to model. The selected Gaussian model also fails here since the small parties are assigned small class priors according to Fig. 5.2. Consequently, for a member to be assigned to one of these parties is unlikely. Naturally, other possible explanations exist, which remain only speculative without further analysis of the data.

The dummy approaches in Tab. 5 show that random guessing is clearly a bad idea when you have 17 alternatives, although it does yield a better F-score than always selecting the largest party (LP dummy). Then again, the LP dummy outperforms random guessing when precision is not important. More importantly however, the final Gaussian ML model clearly outperforms both dummy approaches. The model achieves a high accuracy and a low false positive rate, but obviously still has a lot of room for improvement with respect to sensitivity and precision.

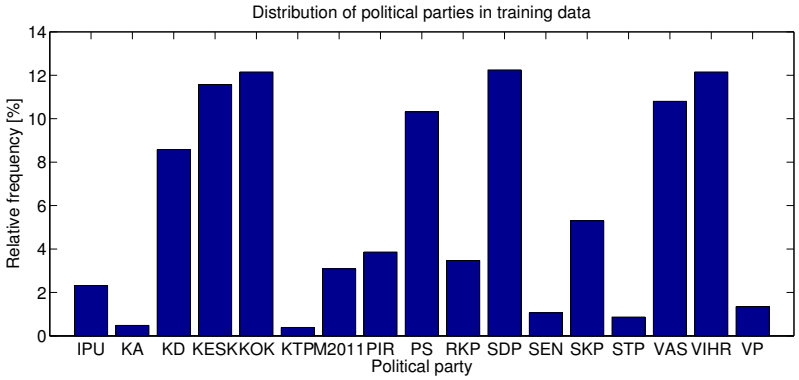

Figure 9: Distribution of political parties in the training data.

5.3 Election results

The central unanswered questions remaining from section 4.2.4 are

- why does not the final model selected through validation (k-NN) give the best generalization performance?
- why does the MOP dummy perform so well?

The questions are certainly not easily answered, but there are a few main points to be made here. First of all, the generalization performance of the k-NN classifier is partly explained by the fact that non-parametric methods tend to work better on large data sets. In other words parametric methods (Gaussian

ML and logistic discriminant) are better suited for small and noisy data sets [3], which obviously is the case here.

Why the validation results of Tab. 7 seem to be in conflict with the generalization results of Tab. 8 is a harder question to answer. The most obvious reason is the possibility of an error in the validation procedure, which would be rather catastrophic for the validity of all results. This is however unlikely, as the validation procedure seems to produce otherwise sensible results. Another possibility is that the training set biases the results, despite the use of K-fold cross-validation. Experimenting with other values for K than $K = 10$ could also yield different results. Additionally, high model orders aren't separately penalized (e.g. through regularization) in the current implementation, which could lead to overfitting in some cases.

The most simple answer to the second question posed in the beginning of the section is that for the given amount of observations, whether or not a candidate is a former member of parliament seems to be the only feature distinctly correlated with the current election outcome. This is supported by the fact that it is also the first feature added to the feature set of the Gaussian ML, which equals the generalization performance of the MOP dummy in Tab. 8. Unfortunately, based on the performance on the test data, one seems to come to the conclusion that of the implemented methods, the MOP dummy remains the best and simplest predictor of the election outcome. With some reservations however, one could also recommend the logistic discriminant if accuracy and precision were the most important metrics of the classifier.

5.4 Future Work

Perhaps somewhat unexpectedly, the binary classification of the election outcome proved to be more challenging than the multi-class political party prediction task. This is why more effort was put into testing different approaches on the former. Consequently, a lot of room for experimenting is left for future work in both cases. In particular, it would be interesting to compare a discriminative model such as the logistic (gradient ascent) classifier, with the implemented generative Gaussian ML model in the party prediction task.

Many sensible models still remain to be examined for the election outcome task. For example one could start by implementing a simple classification tree which combines the "previous MOP" feature with other features to see if any performance gains are possible over the MOP dummy. Understanding the true reasons for the validation/ generalization results mismatch discussed in 5.3 is also a theme requiring further investigation. One partial solution would be testing different regularization models and cross-validation partition sizes. Once progress is made in the aforementioned fields, it could be reasonable to try some mixture of experts to boost classification performance, as proposed in [3], instead of relying on a single classifier as is done currently. Furthermore, the experts could for example be distributed among voting constituencies or clusters of constituencies. Another idea, which actually showed some promising preliminary results is to use the party predictions as an additional input feature for the election outcome predictor. Unfortunately, this did not quite make it to the final results of this paper.

References

[1] Nate Silver, "Response bias and the shy tory factor," August 2008.

[2] Official Statistics of Finland (OSF), "Parliamentary elections 2011," November 2013.

[3] Ethem Alpaydin, *Introduction to machine learning*, MIT press, 2004.

[4] Marina Sokolova and Guy Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427 – 437, 2009.

[5] Wikipedia, "Finnish parliamentary election, 2011," 2013.

[6] Wikipedia, "D'hondt method," 2013.