Machine Learning and Political Ideology in the U.S.: An Experimental Case Study using SMOTE

Héctor Bahamonde *[1] and

Cristobal Quininao †[2]

[1]Assistant Professor, Instituto de Ciencias Sociales, O′Higgins University

[2]Assistant Professor, Instituto de Ciencias de la Ingeniería O′Higgins University

March 30, 2021

*hector.bahamonde@uoh.cl; www.HectorBahamonde.com.

†cristobal.quininao@uoh.cl; https://cquininao.wordpress.com.

Authors are listed in alphabetical order. This project was funded by the Center for at .

Abstract

Abstract here.

Abstract length: 2 words.

Please consider downloading the last version of the paper here.

Keywords— conjoint designs; vector support machines; support for democracy; United States.

I. Materials and methods

The overall model training and validation process consists in a three phases. First we clean the database by eliminating repeated observations, and we normalize using a MinMax scaler. We perform two different experiments, first we use machine learning algorithms to classify participants as vary liberals or not, and later we do the same for very conservative. Since our resulting training pairs datasets are strongly imbalanced, prior to the construction of learning models, we use the SMOTE (Chawla et al. 2002) oversampling procedure with five nearest neighbors for each vote-selling sample (Artetxe et al. 2020). The resulting training data pairs is then used for validation in a repeated cross-validation (RCV) process. This process consists in construct several machine learning models by using the same dataset but under the idea of generate folds and repetition. In particular, each cross-validation repetition consists in: partitioning the dataset in a particular number of folds, using each fold as a test dataset while using all remaining data for model training. Finally, we report average performance measures across all repetitions. In this manuscript, the reported results are the average of the 30 repetitions of the CV models.

I. Classification Methods

Several machine learning approaches can be used for analyze the present data, in a predictive model framework (Vapnik 2013; Witten and Frank 2002; Maimon and Rokach 2005). Due to the nature of the experiment and the data, and since we do not have a large dataset, we discard the application of deep learning algorithms (Balas et al. 2019). Therefore, we focus in the following classical well-known methods: *Support Vector Machines, Multilayer Perceptron and Naïve Bayes method.*

I.1 Support Vector Machines (SVM)

As stated in (Maimon and Rokach 2005, Chapter 12), support vector machines (SVMs) are a set of related methods for supervised learning, which has applications to classification (the output data is a sequence of tags) and regression (the output data is a continuous variable) problems. When using SVM for the classifying problem, we look for linear functions that can separate the data based on the support vectors or boundaries of the classes. The idea of the algorithm is to find the optimal hyperplane, i.e., the hyperplane that maximizes the distance to the boundaries of each class. This can be done by stating a quadratic programming problem (Vapnik 2013). When the classes are not linearly separable, then it is possible place the data into a space of superior dimensionality using a kernel trick (Vapnik 2013; Maimon and Rokach 2005), so that the transformed dataset might become linearly separable. We used the SVM module from Python Sklearn library for training and estimation. We try linear and Radial Bases Function (RBF).

I.2 Multilayer Perceptron

Multilayer perceptron (MLP) is the classical feed-forward artificial neural networks (ANN) composed of multiple densely interconnected layers of computational units, aka artificial neurons (Wolff et al. 2019). It corresponds to a supervised learning algorithm that finds a function that maps our multidimensional samples (socio demographic data) into the vote-selling response of the experimental participants. Any MLP is constructed as follows: (i) we fix a number of hidden layers, (ii) the architecture starts with m artificial neurons corresponding to the features of the input data corresponding to the input layer, (iii) any artificial neuron in the first hidden layer integrates the inputs from the input layer and combines the input values with a weighted linear summation (the weights become part of the parameters to be tuned in the learning phase), (iv) the result of this summation is nonlinearly transformed through an activation function (for instance an hyperbolic tanh function). The procedure is repeated for each hidden layer, until the output layer is reached. The connection weights can be learned from data applying the back-propagation algorithm (Haykin and Network 2004). Concerning the implementation of the MLP, we used the MLPClassifier module from Python Sklearn library for training and estimation. The MLPClassifier implements a MLP algorithm that trains using back-propagation through a stochastic gradient descent. We used the rectified linear unit function for the activation of the hidden layers. The MLP model constructed has two hidden layer, each of 50 neurons.

I.3 Naïve Bayes Method

Naïve Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption that all features of an individual are independent. Bayes' theorem provides a rule to calculate the conditional probability of an event given some knowledge, in terms of the conditional probability of the knowledge given the event. This idea along with the naive independent assumption allows the model to estimate the joint probability distribution of a feature vector as the product of the unidimensional distribution probabilities of each feature (Wolff et al. 2019). In our study we use the GaussianNB module from Sklearn library, which implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian.

II. Classification Performance Metrics

We illustrate the chosen metrics with only liberal classification machines, the conservative situation is straightforward. At the end of each cross-validation fold we have two different vectors: y_{test}

corresponding to the responses of the participants to the very liberal question restricted to the fold used for testing, and y_{pred} corresponding to the predicted answers given by the model for the input features. With these two vectors at hand we compute the

- True positive counts (tp): number of participants such that the model predicts as very liberal and they actually belong to the very liberal class
- True negative counts (tn): number of participants such that the model predicts as not very liberal and they do not belong to the very liberal class
- False positive counts (fp): number of participants such that the model predicts as very liberal but they do not belong to the very liberal class
- False negative counts (fn): number of participants such that the model predicts as not very liberal but they actually belong to the very liberal class

with this numbers, we compute the Recall or sensitivity R[1], the positive predictive value PPV or precision[2] and the f-score[3] with the relationships

$$R = \frac{tp}{tp + fn}, \quad PPV = \frac{tp}{tp + fp}, \quad F = \frac{2}{1/R + 1/PPV},$$

and finally report the average and the standard deviation across all cross-validation folds. Notice one could use the accuracy metric defined as

$$A = \frac{tp + tn}{tp + tn + fp + fn} = \frac{tp + tn}{\text{size of the fold}},$$

[1]The fraction of examples classified as very liberal, among the total number of very liberal examples.
[2]The fraction of true very liberal examples among the examples that the model classified as very liberal.
[3]A perfect model has an f-score of 1.

but considering that the dataset is strongly imbalanced, previous measures are more informative. Another metric widely used to compare binary classifiers (Wolff et al. 2019; Artetxe et al. 2020) is the Receiver Operating Curve or ROC, which corresponds to a plot of sensitivity R versus the false positive rate $FPR = \frac{fp}{fp+tn}$. ROC is computed as a probability curve and, and the area under ROC curve of AUC represents the degree of separability. In other words, it quantifies in a number how much the model is capable of distinguishing between classes. In our setting, the higher the AUC, the better the model is at predicting liberals as liberals and non liberals as non liberals. Remark that (i) a perfect model has AUC equals to 1, (ii) a model with AUC approximately 0, means that the model is actually reciprocating the classes, therefore a simple relabelling is enough to get good results; and (iii) the worst case scenario is when the AUC equals 0.5, meaning that the model has no class separation capacity.

The methodology we use to compute previously explained metrics is as follows: at each fold test, we use the remaining folds as training dataset. Since the resulting training data set is imbalanced, we use the oversampling technique SMOTE with five neighbors on the minority class. Remark that the test set remains imbalanced which has some consequences. In particular, even one misclassified sample translates into large reductions of the performance measures. Moreover, the number of remaining liberal samples in each fold depends on the number of folds, thus having consequences in the imbalance ratio. We report results with decreasing number of RVC folds in order to test the stability of the results.

II. Results

Tables 1 and 2 show the results for metrics mentioned above of the machine learning techniques after 40 repetitions (3 fold), 10 repetitions (4 fold), 8 repetitions (5 fold) and 4 repetitions (10 fold) of the RCV experiments with SMOTE class imbalance correction. There is no statistical evidence to argument that the number of folds has an effect on the performance of each method. An F- test over the number of folds shows that there is no statistically significant difference $p \gg 0.1$.

If we test the performance of each methods we find some differences, specially on the PPV, f-Score and AUC metrics. SVM with RBF kernel is above SVM Linear, MLP, and NB independently of the number of folds. On the other hand, the recall metric (R), shown in Table 1 (left), shows no sufficient evidence to affirm that methods perform differently. Indeed, applying an F-test over the different type of machines for the each folds number we find no strong statistical evidence, and it would depend on the confidence level whether or not we reject the null hypothesis (for instance, the p-value es approx 0.026 for 10 folds). However, the result changes dramatically, if we discard the MLP model which performs below any other model ($p \approx 0.77$ for 10 folds). We conclude that according to the recall metric, all methods except for MLP have similar performance. Figure 2 shows the ROC curves for all approaches in the case of RCV with 5 folders.

The f-scores shown in Table 2 confirm that SVM with RBF kernel improves over SVM Linear, MLP and NB regardless of RCV number of folders. An F-test carried out over these results confirms that the performance differences between predictive models are statistically significant ($p \approx 0.002$) . Specific one-sided t-tests comparing each pair of modeling approaches confirms that SVM with RBF kernel perform better than SVM Linear, MLP and NB. However, the superiority of SVM with RBF kernel relative to MLP is less pronounced ($p \approx 0.05$). On the contrary, the effect is more pronounced for the AUC metric. The F-test carried out over these results confirms ($p \ll 0.001$), t-test pairwise tests ($p < 0.001$).

Table 1: *Average ± standard deviation Recall R (left) and positive predictive value PPV (right) performance of SVM (linear), SVM (RBF), MLP and NB for decreasing number of folders in the repeated cross validation process. All results are calculated with SMOTE oversampling correction of class imbalance.*

		SMOTE		
nfolds	SVM (linear)	SVM (RBF)	MLP	NB
10	0.68±0.07	0.67±0.08	0.65±0.07	0.68±0.08
5	0.68±0.05	0.67±0.04	0.65±0.06	0.66±0.06
4	0.67±0.03	0.66±0.04	0.64±0.04	0.67±0.04
3	0.68±0.03	0.67±0.05	0.67±0.05	0.67±0.05

		SMOTE		
nfolds	SVM (linear)	SVM (RBF)	MLP	NB
10	0.45±0.06	0.52±0.08	0.47±0.07	0.44±0.07
5	0.46±0.05	0.52±0.07	0.48±0.05	0.44±0.05
4	0.45±0.04	0.52±0.06	0.47±0.04	0.43±0.05
3	0.45±0.03	0.52±0.04	0.45±0.03	0.44±0.04

Table 2: *Average ± standard deviation f-score (left) and AUC (right) performance of SVM (linear), SVM (RBF), MLP and NB for decreasing number of folders in the repeated cross validation process. All results are calculated with SMOTE oversampling correction of class imbalance.*

		SMOTE		
nfolds	SVM (linear)	SVM (RBF)	MLP	NB
10	0.54±0.06	0.58±0.07	0.54±0.06	0.53±0.06
5	0.55±0.05	0.59±0.05	0.55±0.05	0.53±0.05
4	0.54±0.03	0.58±0.04	0.54±0.03	0.52±0.04
3	0.54±0.02	0.58±0.03	0.54±0.03	0.53±0.04

		SMOTE		
nfolds	SVM (linear)	SVM (RBF)	MLP	NB
10	0.68±0.04	0.72±0.05	0.68±0.04	0.67±0.04
5	0.67±0.03	0.72±0.03	0.68±0.03	0.67±0.03
4	0.67±0.02	0.72±0.04	0.67±0.03	0.67±0.03
3	0.68±0.02	0.71±0.02	0.68±0.02	0.67±0.02

References

Artetxe, Arkaitz et al. (2020). “Balanced training of a hybrid ensemble method for imbalanced datasets: a case of emergency department readmission prediction.” In: *Neural Computing and Applications* 32.10, pp. 5735–5744.

Balas, Valentina Emilia et al. (2019). *Handbook of deep learning applications.* Vol. 136. Springer.

Chawla, Nitesh V et al. (2002). “SMOTE: synthetic minority over-sampling technique.” In: *Journal of artificial intelligence research* 16, pp. 321–357.

Haykin, Simon and N Network (2004). “A comprehensive foundation.” In: *Neural networks* 2.2004, p. 41.

Maimon, Oded and Lior Rokach (2005). “Data mining and knowledge discovery handbook.” In:

Vapnik, Vladimir (2013). *The nature of statistical learning theory.* Springer science & business media.

Witten, Ian H and Eibe Frank (2002). “Data mining: practical machine learning tools and techniques with Java implementations.” In: *Acm Sigmod Record* 31.1, pp. 76–77.

Wolff, Patricio et al. (2019). “Machine learning readmission risk modeling: a pediatric case study.” In: *BioMed research international* 2019.

.... **Word count**: 1,814 ....

III. Appendix

I. Appendix 1