Inferring Real Citizen Behavior from Google Search patterns.

Anirudha Kumar Akela

2018MT10740

Department of Mathematics

akelaanirudha@gmail.com

+919370984612

Students I4 Challenge: Ideation at Isolation; Implementation at Institution

Challenge VI : Inferring Real Citizen Behavior from Google Search patterns

Solution

I have used Machine Learning and Deep learning to predict the election results of Lok sabha and Rajya sabha elections using data from the search patterns taken from google trends.

Assumptions and declarations

Data available on Google Trends:

1. Available weekly from 2004 to present day
2. Available state-wise and for all major political parties.
3. The google trends data has already been normalized over the number of internet users and thus is a true measure of "popularity".[1]
4. Google trends automatically clubs together related searches into a single entity[2] and all such related search activities are taken into account. (for example, "bjp", "bhartiya junta party", "bhajpa",etc. will be clubbed to "Bhartiya Janata Party : Indian Political Party" by Google Trends)

Assumptions:

1. We only look at the vote share and not the number of seats won as vote share is closer to the measure of true "popularity" of a candidate or party and better reflects the success of the party.
2. We only try to predict for parties with more than 10 percent vote share because the data is skewed for smaller and more regional parties and even a small-lived surge in popularity could highly skew the google trends data and affect the model's prediction.

Issues with Google Trends data:

1. Google says that data from before 2009 may not be accurate as they had no filtering methods, and we see this evidently as the data from before 2009 is highly erratic and irregular. Unfortunately, this renders data from before 2009 useless.
2. Google improved its data collection policy in 2011 and thus there was a significant change in the amount of data collected from 1 Jan 2011[3]. Thus, we cannot consider data for any time frame that has data from both before and after 1 Jan 2011.
3. Not enough search result data is available for states with poor internet penetration and/or less population to reliably and accurately use in the model as even a short sudden spike in interest for candidate or a party in the five year period could heavily skew the data.[4] (some such states are: Arunachal pradesh, Goa, Manipur, etc.)
4. These restrictions and issues with the data mean that we can only analyze the results for elections from 2016 onwards.

Machine learning algorithm

Assumptions and declarations:

1. The model has been trained to predict the vote share of any major political party for both the Lok Sabha and assembly elections in any major state by looking at the "popularity" of the party by looking at their search results from Google trends.
    a. This means that we can safely ignore the activities within a state's politics as this will already be acounted for in the Google Trends data.
    b. This enables us to train the model over Party-State-Election tuples.
2. We have 106 available Party-State-Election tuples and thier corresponding data.[5]
3. The features used are:
    a. Weekly trends data
    b. Vote shares of party in previous LS ans assembly elections in the same state.
    c. Whether the party is in power at the State or the Centre level.

Features

The following features are used by the Deep Learning algorithm to model the results[6]:

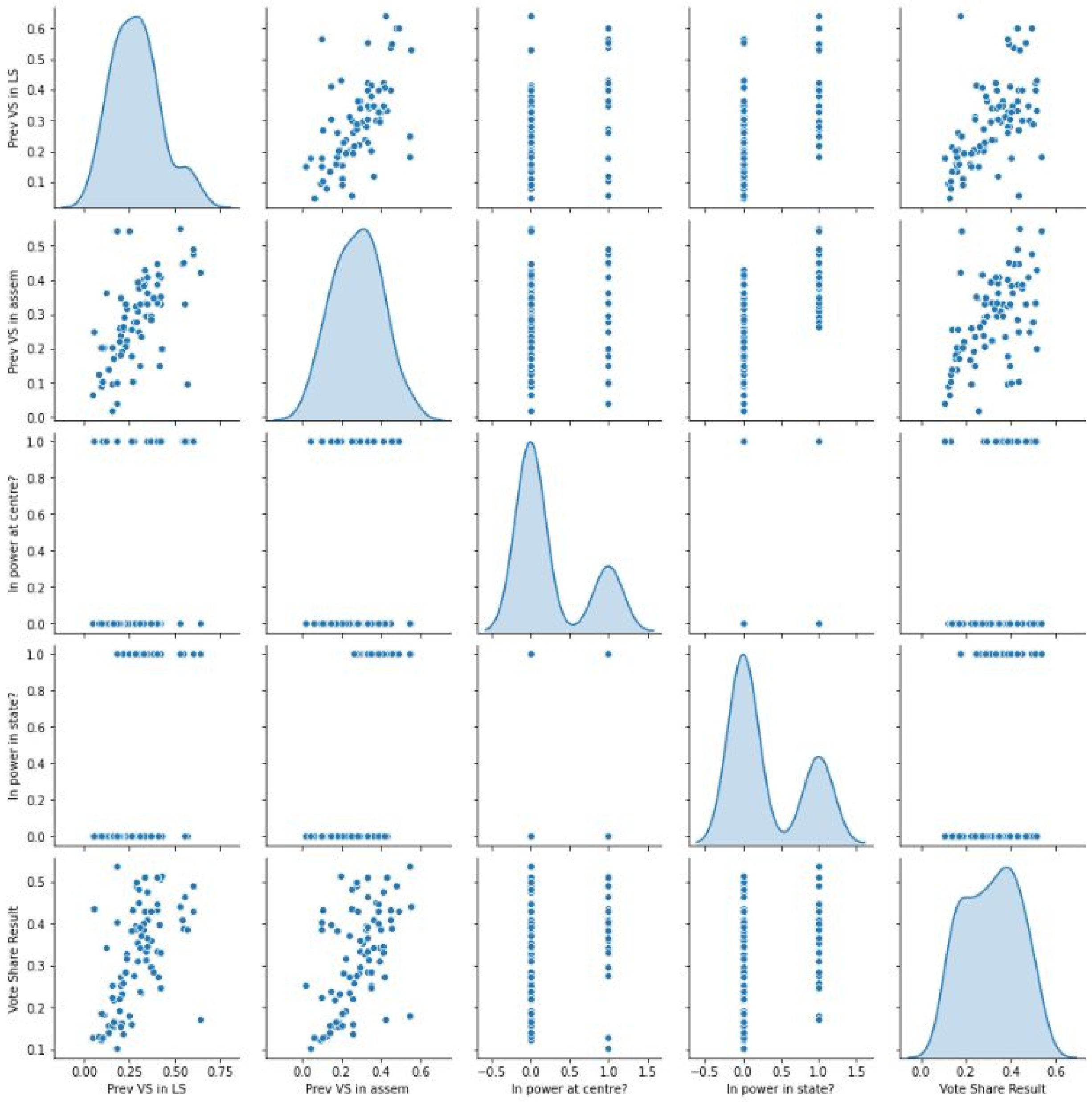
Feature	Reason and how it affects the results
Google trend data for past 262 weeks	As a proxy for "popularity"
`Prev VS in LS`	To take into account the past performance of the party in that state in Lok sabha and Assembly elections. (as past popularity is also a contributing factor in the performance of the party)
`Prev VS in assem`	
`In power at centre?`	We have only the total searches for a party in a state and this measure would obviously be affected whether the party is in power at the cenre or the state level. These two paramaters attempt to take this into account.
`In power in state?`	

Training and Testing data split

We use 80-20 split for train and test set. Thus we train the model on 85 examples and use 21 examples to evaluate the model.[7]

Inspecting the Data and its dependencies:

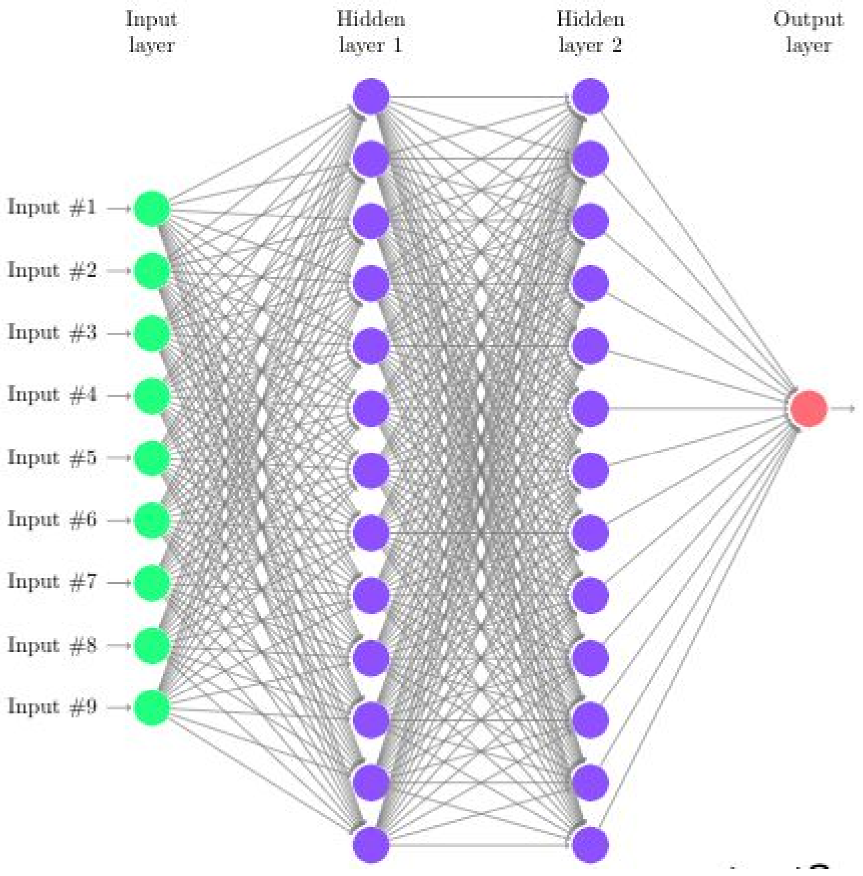
```
<seaborn.axisgrid.PairGrid at 0x7fefe4612cf8>
```



Feature normalization

All the features have been normalized to help better optimise the model and take care of any potential outliers.

Neural Network Model Architecture



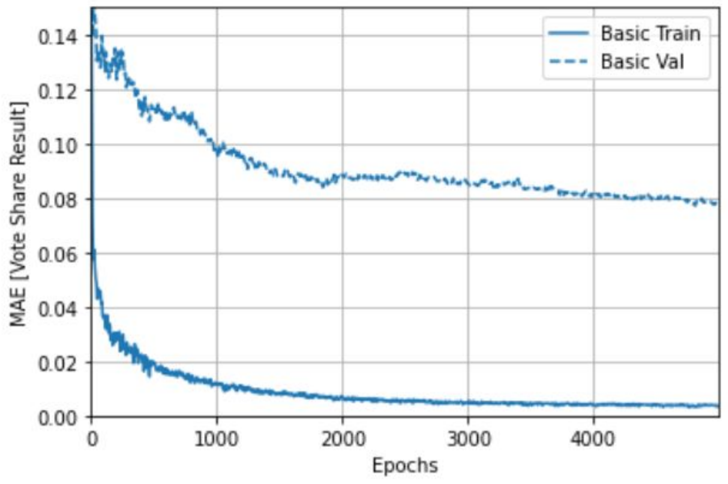
The neural network used has the following architecture[8]:

1. 266 input features (262 weeks, 2 for VS in LS and assembly, 2 for whether in power at centre or state).
2. Two hidden layers with 64 neurons in each
    a. The first hidden layer is a fully connected layer with the sigmoid activation function.
    b. The second hidden layer is a fully connected layer using the relu activation function.
    c. The final output is calculated using a fully connected network from the outputs of the second conneceted layer.
3. There are a total of 23,313 trainable parameters in this neural network architecture.

Training the model

1. The model has been constructed, compiled and trained using Google's Tensorlow 2.0 and Keras high level- neural network architectures.
2. The RMSprop algorithm has been used to optimize the model.
3. Mean absolute error of the predicted vote share has been used as the evaluation metric for the performance of the model.
4. Using these hyper-parameters, the model has been trained for a total of 5000 epochs using the keras.fit() function.

```
Text(0, 0.5, 'MAE [Vote Share Result]')
```

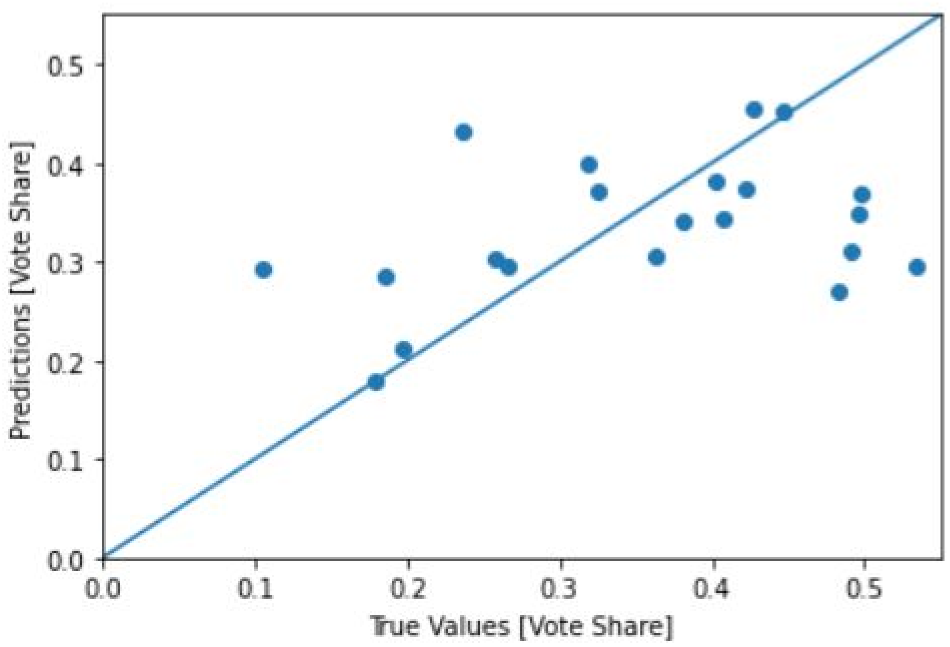


Training and test (validation) errors vs iterations of the optimisation algortihm.

This is the mean absolute error on the training examples over the iterations of the optimization algortihm. As we can see, the error flattens out and we do not gain any performance by training the model further.

The average absolute error over the test set is around 0.08 after about 5000 epochs.

Making predictions on the test examples



Predicted vote shares vs the actual vote shares over the test examples.

Analysis predicted results

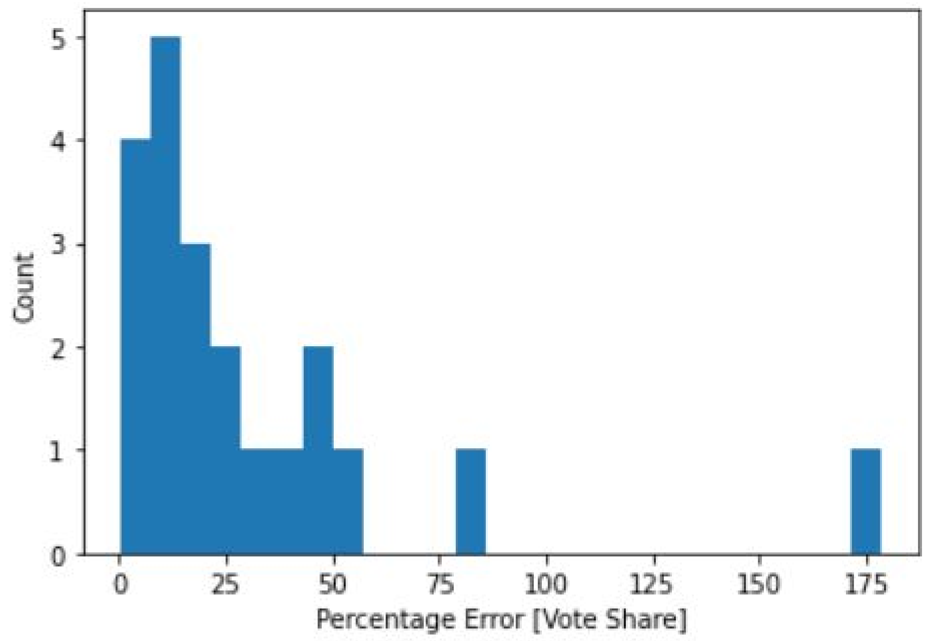
The algorithm does well on Party-State-Election sets with intermdiate vote shares. (15-40 percent) and does particularly poorly on examples where the vote share is either very low or very high.

Evaluation metric

I am using the absolute percentage error in the vote share predicted as an evaluation metric for the algorithm.

$$percentageError = \frac{|predictedVoteShare - actualVoteShare|}{actualVoteShare}$$

The predicted vote share has an **average absolute percentage error of 30.46%**. (over 21 test examples)



Distrbution of the absolute percentage error in the predicted vote share on the test examples.

From the distribution histogram, we can see that only a couple of the test examples make a very large portion of the total error. Some of the reasons for this large error on these two examples is:

1. *Very less vote share for a popular national party but with less footing in that state.*
   (the example with ~175 percent error is for BJP in the Kerala 2016 assembly elections, the popularity of the party would have been higher due to the surge in popularity of BJP with the win in the 2014 LokSabha elections) [9]

2. *Coalitions not capturing the real vote share.*
   (the example with ~87 percent vote share is BJP in the Bihar 2019 Lok sabha elections where BJP made coalitions with the regional parties[10] and BJP only fought on 19 of the 40 Lok sabha seats[11].)

Ignoring these two outliers, we have an **average absolute error of 19.87 percent** in the predicted vote shares.

Thus, the model predicts vote shares within 20 percent of the actual vote shares with in 90 percent of the cases (19/21).

Concluding remarks

Predicting politcal results is hard and without actual public polling like those done in exit polls, its even harder. This problem is specially challenging due to the complex nature of the problem and the influence of many dozens of factors affecting the performance of a party in an election. Some of these may be popularity and public opinion of major leaders of the party, opinion of the work done by the party, public frustation/satisfaction with opposing parties or just that the “junta” wants a change in power. It is very hard to capture all this information quantitatively and calculate how all these factors will affect a party’s performance. Another possible source of error in human-made predictions is the personal bias of the expert. All this reasons make a purely - mathematics based Machine Learning algorithm a very good choice for such a task.That said, it is still not perfect as it is bery hard to quantify all these “emotions” of the public and a machine cannot take these things into account.

Related Files and Data Sheets:

- The accompanying Google Colab Notebook where you can view all my code and make sure I didn’t cheat in the prediction errors: https://colab.research.google.com/drive/1I6rnENGTaiKmrRipvZgls26tOtO-ybKe
    - You will need to upload this file to Google Colab to run the notebook.
    - Running this Colab notebook might give skightly different error because of random initialization.
    - Here is a snapshot of the notebook used for the discussion here.
- Code on Github
- All the data regarding Party-State-Election vote shares can be found in this sheet: https://drive.google.com/file/d/1KkYQJicVJ4aovv31NR8PB_Q6mPEhKPAP/view?usp=sharing
- All the raw data from Google trends can be found here

Related Links:

[1] https://support.google.com/trends/answer/4365533?hl=en
[2] https://1drv.ms/u/s!App3rdkJBBi_iRF7a8WxR8asW7KW?e=YwYty5
[3] https://1drv.ms/u/s!App3rdkJBBi_iRg1WApjV9QYawDC?e=BIHPEx
[4]https://trends.google.com/trends/explore?date=all&geo=IN-MN&q=%2Fm%2F0135cw,%2Fm%2F0135dr
[5]https://colab.research.google.com/drive/1I6rnENGTaiKmrRipvZgls26tOtO-ybKe#scrollTo=WWEKyb4bbBjK&line=1&uniqifier=1
[6]https://colab.research.google.com/drive/1I6rnENGTaiKmrRipvZgls26tOtO-ybKe#scrollTo=CiX2FI4gZtTt&line=1&uniqifier=1

[7]https://colab.research.google.com/drive/1I6rnENGTaiKmrRipvZgls26tOtO-ybKe#scrollTo=qn-IGhUE7_1H&line=2&uniqifier=1
[8]https://1drv.ms/u/s!App3rdkJBBi_iRk_hG0vje2qkE4b?e=0w7Y83
[9]https://www.news18.com/news/opinion/five-reasons-why-bjps-victory-march-was-halted-in-kerala-as-congress-enjoyed-sabarimala-dividend-2159139.html
[10]https://economictimes.indiatimes.com/news/elections/lok-sabha/bihar/bihar-lok-sabha-election-results-live-news/articleshow/69457325.cms
[11]https://www.indiatoday.in/elections/lok-sabha-2019/story/bjp-jdu-seat-share-bihar-lok-sabha-election-2019-1376204-2018-10-26