Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

Political Position Estimation of Politicians Using Social Media Communication

Karishma Rao

August 2022

A Dissertation submitted in partial fulfilmentof

the requirements for the degree of

Master of Science in Computer Science (Data Science)

Supervisor: Dr. Carl Vogel

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, ismy own work.

Karishma Rao

August 19, 2022

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Karishma Rao

August 19, 2022

Acknowledgements

I would like to take this opportunity to thank all the people without whom this dissertation work would not have been possible.

Firstly, I would like to thank my supervisor Dr. Carl Vogel for his support and guidance over the course of this project work. His feedback, suggestions and insights were greatly helpful in the completion of this dissertation.

I would like to thank my parents and my brother for their constant encouragement and support. Their feedback from a third-person perspective helped me refine the work. Their emotional support was indispensable.

Lastly, I would like to thank my friends for being there for me throughout the tumultuous period of this dissertation.

KARISHMA RAO

University of Dublin, Trinity College
August 2022

Political Position Estimation of Politicians using Social Media Communication

Karishma Rao, Master of Science in Computer Science
University of Dublin, Trinity College Dublin, 2022

Supervisor: Dr. Carl Vogel

Politicians make use of social media to communicate with the public and voice their opinions on different subject matters of public interest. Social media communication is an accessible and easy way for the politicians to express their stance and try and gain the public's favour. However, does this communication accurately represent the voting behaviour of the elected representatives? This project aims to use Twitter communications of the members of the Dáil Éireann (TDs or deputies), the lower house of the Oireachtas (the Irish Parliament) to try and predict their voting behaviour in the motions put forward in front of the Dáil Éireann. The information the general public is exposed to is limited to communications by the officials through different media. By trying to align the online text contributed by the elected officials with their behavior during legislative vote, it is possible to try to get a measure on whether there is delivery on promises made. As part of this study, the tweets of the members of the Dáil Éireann were extracted using Twitter API and the voting data of the deputies were extracted from the Oireachtas's official website. The two datasets were aligned and after text pre-processing and feature engineering techniques were employed, the data was evaluated, and predictions were made using three different classifiers. Another dataset was prepared that used only the voting behaviour of the TDs obtained from the Oireachtas's website and the models were evaluated against this dataset. The performance of twitter and voting data combined dataset models were compared against the models that used only the TDs voting data. This was done to analyse if there was any improvement in the predictions of the TD voting behaviour after adding features from the Twitter data. It was found that out of the three classifiers, the performance of the Decision Tree classifier significantly improved after aligning the twitter data with the individual deputy voting data. The Random Forest classifier performed the best in both cases. The results obtained in this study indicate that with a more finely tuned dataset, twitter data can be used to predict the voting behaviour of the TDs in the legislature.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Estimation of the political position of parties and of politicians has long since been a topic of interest in the field of Cognitive InfoCommunications. In the early 2000's work has been done to estimate the political ideology or inclination of parties as left-wing or right-wing by analysing the party manifestos. The whole text was converted to text units by either expert coders or by using computer coding techniques. The method of analysis that used expert coders involved manually reading and interpreting the text according to a prior confirmed coding scheme and then classifying the political stance. Whereas computer coding techniques relied on a more quantitative rather than a qualitative analysis, by classifying the text according to the contents of pre-defined dictionaries. With the advent of social media, there was a rising interest in using the online content put out by politicians and parties to gather predictions on popularity or election voting outcomes. There has been work done on using social media communications to gauge the public's opinion on politics and popularity of candidates. Research has also been done on the effect of social media in influencing people's opinions.

1.2 Motivation

Politicians survive on winning the public's opinion in their favour. Spending millions on campaigns and advertisements used to be the only way to get recognition and gain momentum. This creates an atmosphere where those backed by powerful people eclipse over the newcomers. Social media provides the means to bridge this gap in monetary and power differences. It has become an increasingly relevant platform for political communication. According to Twitters advertising resources statistics, Twitter had 1.35 million users in Ireland at the start of 2022 [1]. That's around 27% of the population. Given that Twitter restricts the age to 13 and above and only 7% of the population lies in the age range 13-17, Twitter provides a great platform for the politicians to directly speak to a huge chunk of their voter's base. Based on a recent study on the effect of social media on political competitions [2], it has been found that social media platforms provide a good launching pad for newcomers to gain support at almost no cost. For both the newcomers and also the well-established politicians, social media platforms lets them connect to their voters at a human level. The popularity metrics of the communication like likes,

comments or retweets would also give them an idea about how their views and opinions are being received and also to learn about the issues that the public is most concerned about.

Looking at it from the public's perspective, social media serves as an easy way for them to connect to the politicians in their district and get to know the policies they support or even to highlight matters to interest. The voting power lies with the public and the information they are exposed to is limited to communications by the officials through different media.

Following this train of thought, it would be interesting to see if the opinions voiced by these politicians on social media aligns with how they vote in the legislative assemblies, which is where all the actual decisions affecting the lives of the general public are made. The common man has no way of knowing how the politician they voted for, votes on the policies that influences the public's regular life. Although, the voting details of the motions that were debated and voted on are published on the official websites of these legislatures, a regular person would not regularly go and check these websites to see how their elected representative is voting.

It can be assumed that an elected official would use social media platforms to voice their stance on matters of public interest and that is how they would correspondingly vote for in the legislatures. This leads to the question of whether these social media communications can be used to predict how a politician would vote in the future. Of course, not all the social media content put out by the politicians are connected to their views on political matters. Since it's a platform they can use to make them seem more relatable to the public, it could also include titbits about their personal life or hobbies and such. This proves to be an interesting subject matter of study. Can the Twitter data, which would have a huge range of content, be used to predict the voting behaviour of the politician?

1.3 Research Question

Can Twitter data of the elected members (TDs or deputies) of the Dáil Éireann (lower house of the Irish Parliament or Oireachtas), be used to predict their voting behaviour in the legislature?

1.4 Research objective

As discussed previously, social media is an important arena in the political sphere. With the number of social media users on the rise every year, it has become a powerful tool that can be employed by politicians to gain favour. There have been numerous studies on the effect of social media in the political domain. This research aims to link the publicly available opinions and views openly stated on Twitter with the rather obscure voting behaviour of the individual elected representatives.

This project aims to get an estimate of the political position of the elected representatives of the Dáil Éireann, that is, the TDs/deputies from their tweets. The objective is to determine if this estimate of their political position can be used to predict the legislative behaviour of the deputies. Legislative behaviour in this context means predicting how the deputy would vote when a motion is being debated on during a Parliamentary session.

To achieve this objective, the approach is to use text data analytics techniques on the Twitter data and TD voting data and simultaneously do a comparison of the relative informativity of the linguistic and non-linguistic features used. The method designed was to get the twitter data and align it with the legislative voting data obtained from the official parliamentary website. Appropriate feature engineering techniques would then be employed on this aligned dataset that would have been cleaned

using text pre-processing techniques. In order to get predictions on future voting behaviour of the deputies, the data would be fed into machine learning models. This necessitates the conversion of the text data into appropriate numeric forms through vectorization to be used as input for machine learning classifiers. Since the objective was to see if twitter data provides adequate insight into the political position of the elected representative, two different datasets would be prepared for comparison of the accuracy of the predictions. This would give insight on whether adding twitter data does indeed improve the accuracy of the predictions on how the elected deputy would vote in the future motions.

1.5 Dissertation Structure

The following is a description of the layout of the rest of this dissertation. Chapter 2 gives an overview of the related research that has been done on political position estimation, sentiment analysis and the area of cognitive infocommunications. Chapter 3 details the data collection and processing stage. Chapter 4 describes the methodology followed in this study, explaining the experiments designed, models and evaluation methods used. Chapter 5 presents the results of the project work. Chapter 6 is the conclusion that summarizes the work and challenges encountered. It also highlights possible areas of future work based on the results of this study.

Chapter 2

Literature Review

This chapter focusses on the prior research work that has been carried out in this area. It explores the approach used previously for political position estimation and discusses related works on the techniques used in this study.

Research papers on the discipline of cognitive infocommunications contributed towards the understanding of this area.

'Cognitive infocommunications (CogInfoCom) is an interdisciplinary field that targets engineering applications based on emergent synergies between ICT and the cognitive sciences' [3]. The analysis of social media networks has a two-fold application in CogInfoCom. It can be used for getting an insight for social phenomenon such as understanding political analysis or the much recent case of analysis of the spread of a pandemic like Covid-19. A lot of research has been done in the field of metadata and content analysis of social media such as flagging offensive content, predicting popularity, mining to get crowd opinion. The other application in CogInfoCom is providing user with information that enables them to make choices such as through review systems or aggregation services. Social networks have been proven to have increasing cognitive capabilities. [4] argues that providing solutions to both private (that is, thought) and public problems (communication) for humans is a pre-requisite for the assimilation of a CogInfoCom technology as a success.

The perceptions and actions of humans are fundamentally driven by their emotions. The emotional information is coded into both verbal and non-verbal cues in a daily context. The current research on effectively decoding these social cues is diversified and fragmented. [5] suggests a more collaborative approach to implement a robust context-aware human computer interaction system. These include efforts to build a shared digital data repository, new methods for processing data and better computational models that can interpret context. Two main research pipelines are proposed: one dealing with cognitive communication and the second focussing on the technological implementation of the insights gained from the behavioural analysis from the first area. This technological implementation includes development of efficient algorithms for social learning.

There has been exploratory work on emotion detection from multimodal signals such as the detection of spontaneous emotions from a TV political debate speech [6]. Spontaneous emotions are harder to detect than acted emotions due to their task dependent nature. Both the categorical and dimensional models were considered to label the emotions of the audio chunks through crowdsourcing. SVM and FNN methods were used but the fixed-length constraint for the FNN model inhibited the accuracy score.

Accomplishing a task merely does not signify the existence of collaborative effort from a group.

[7] attempted to construct linguistic measures for behaviour of small groups working in collaboration. The linguistic behaviour pattern was observed by also including psychological factors like personality traits of each individual and perception of dominance.

Another work relating to linguistic and behavioural interaction was an automatic scoring system of oral delivery skills [8]. This envelopes the detection and rating of verbal and non-verbal elements. The video classification dataset was generated from unsupervised learning. Such an approach of active data representation is faster than the conventional approaches that involve the design of event detectors through supervised learning. The results were largely in line with human scores.

The research on social media analytics has been extensive and varied. [9] provides a starting point for research into multimodal conformity. The aim was to analyse the contents of Tumblr blogs and check if it conforms to the blog name. This has an extended application in filtering content in websites or portals requiring moderation through the titles or URLs.

The semantics used in a text may influence the impact of the text on the reader's thoughts and emotions. Research has been done on use of text data analytics for emotional analysis. The work [10] examined the linguistics choices of online news covering incidents of terrorist attacks and compared it with those used in the case of natural disasters. The purpose was to identify whether the news articles employed linguistic choices that would comparatively elicit more emotions of fear and anxiety among readers when the topic was terrorist attacks. This was implemented by analysing nominal and verbal categories and the articles on terrorist attacks had more verb usage indicating the hypothesis to be true.

The left-right positions of political parties have been theorised to affect other arguments such as the voting behaviour, policy making, competition during elections and such. This ideological position estimation is a crucial steppingstone to a whole body of subsequent research that deals with various aspects of politics and democracy. The MRG (Manifestos Research Group) has played a key role in initial research on party position estimation by coding the party manifestos from a range of countries and estimated their left-right position. [11] attempted to find the best way to use this MRG data (which at the time, was the only data available) to estimate political position and quantify the quality of this data from the accuracy obtained from their method. The MRG data was found to be quite accurate for implementation of political position.
[12] proposed the use of computer coded content analysis techniques to estimate the political position of politicians instead of the conventional method of using expert coders. These computer coding techniques followed a dictionary-based approach. Although the paper introduced a new explicitly positional expert coding scheme, it was noted that the huge amount of hand-coding associated outweighs the benefits. This avenue of using computer coding techniques opened the possibility to estimate the political position of individual politicians, as research up till then had been limited to political parties. [13] presented a novel method of treating text as data in the form of words and not something that needs to be read and interpreted. This was a breakaway from the traditional textual content analysis techniques using human or computer coding. As the method treated words as data, it meant the method could be extended to other languages without requiring knowledge of the language by the analyst. This method was applied to estimate political position from party manifestos and then extended to estimate political position from speeches made in legislature. They also provided a measure of uncertainty of the political position estimate, enabling the categorisation to be statistically significant or merely an error. [14] compared the linguistically quantifiable features of the party manifestos of small right-wing European parties and found that fully computerized analysis can give insights into right-wing party policies.

With the rise of social media there was now a new source of data available for political analysis apart from political texts from manifestos. The use of social media to study and predict the offline behaviour of the subjects is the new area of study. There has been work done on the influence of social media on stock markets, economics, public sentiment, etc. Much work has been done on the influence of social media on people and the possibility of its effect on their behaviour in other spheres. A sub-area of research in this field focusses on the effect of social media on the political scenario. The work suggests

that there is increasing evidence that social media communication can be used as an indicator for offline behaviour. There have been studies to support that social media popularity is indicative of election outcomes. [15] found that the Twitter mentions of a candidate were a statically significant indicator of how the candidate performed in the elections. This study was an improvement over the previous studies as it took into account an array of variables such as the district's gender and race composition, established media coverage and incumbency. Furthermore, the study found that social media provided a more accurate estimate of the political behaviour than televised media. [16] did an analysis on the performance of Twitter data in predicting election outcomes when location data is added. They note that user data on social media platforms has transitioned from only text to multimedia content and there needs to be more research in the area of multimodal sentiment analysis. The effort so far has been on NLP techniques that focus on text based data. [21] gives a survey of sentiment analysis techniques. Feature selection is the first step in sentiment analysis and the approaches for this have been lexicon based or automated statistical methods. Point-wise mutual information, latent semantic indexing and chi-square method are the most popular feature selection methods.

[17] shows the feasibility of tri-modal sentiment analysis that combines audio, visual and textual data. They produced a new dataset sourced from YouTube videos and through experiments showed that the integration of all 3 is an improvement over models that use one modality at a time. They identified polarized words, smile, gaze, pauses and pitch as the most useful differentiators.

The current state-of -the-art in task based sentiment analysis mostly ignores the complexity of human emotions, failing to capture the dimensionality and granularity in its entirety. The current research is mostly based on document, sentence and word level sentiment analysis.

[18] traced the development of statistical methods that are used in the analysis of literary styles. The approach developed from using word length and frequencies for authorship identification to using statistical methods that used function words like conjunctions to assign probabilities to authorship beliefs. Different statistical techniques were employed over time. In recent times stylometry has shifted because of the advent of machine learning and AI and the huge volume of easily accessible digital data. These modern techniques treat stylometry as a pattern detection problem. [19] presented a technique of using letter bigrams for authorship attribution. The work justified the versatility of the technique by testing it on a diverse range of texts and previously published datasets.

[20] aimed to use Twitter data of Dublin City Council members along with the Councillor and motion data to predict the voting behaviour of the Councillors in the city council. The results obtained were marginally adequate to support this idea. This project work uses this paper as the main guideline for the approach followed and extends the dataset to analyse the behaviour of elected representatives at a national level.

Chapter 3

Data Collection and Processing

3.1 Data Overview and Feature Engineering:

A major part of this project work went into the data collection and feature engineering aspect. The methods employed for this have been described below.

3.1.1 Data Collection:

There were two different datasets that had to be collected: Twitter data and deputy voting data.

Twitter Data:

The twitter data of the TDs were collected using the Twitter API. The extraction of tweets of a specific user requires Elevated Developer access. This had to be requested through the Twitter developer portal. Twitter API restricts the number of tweets that can be extracted from a specific user to the most recent 3200 tweets. But this was not a problem, as the scope of this project restricted the Twitter timeline to the period between January 1, 2022 to June 30, 2022 and none of the TDs twitter exceeded the limit for this time frame.
The list of usernames of the TDs were acquired online from a website [22]. Of the 160 total deputies of the parliament, 7 do not use Twitter. These were excluded from the study. There was a problem extracting the Twitter data of 8 deputies through the API from their username. In the end, a total of 146 TDs were considered for this experiment and their tweets were extracted. Python's tweepy library was used to get data from the API and the user_timeline function was used to get the tweets of specific users from their usernames. Initially, Twitter data was obtained for the period February 2019 to June 2022 resulting in 298,881 rows of data. But this dataset had to be reduced to only include the tweets from January 1, 2022 to June 30, 2022 due to problems arising while data alignment with the voting data as described later in this section. The tweets were stored in a csv file to be used.

User	Tweet	Created At
QuinlivanTD	Discussing the problems with the Passport Service and the need for the service to be resourced	2022-07-14 18:00:06+00:00
QuinlivanTD	@OireachtasNews @dfatirl @ClareFM @ClareChampion @ClareEcho @Limerick_Leader @Tipp	2022-07-14 17:30:35+00:00
QuinlivanTD	Lack of a statement from the Taoiseach is truly appalling. https://t.co/KiO6rGGKLv	2022-07-14 11:45:31+00:00
QuinlivanTD	RT @johnfinucane: Speaking to BBC this evening on the abhorrent displays on numerous bonfire	2022-07-14 11:41:09+00:00
QuinlivanTD	Irish times / Ipsos MRBI	2022-07-14 07:16:40+00:00
QuinlivanTD	@jamiemathis01 @orourke_darren @MaryLouMcDonald @ReadaCronin @JohnnyGuirke Green	2022-07-13 21:36:56+00:00
QuinlivanTD	@CllrPioSmith @davidtobin100 One like This would be kinda normal https://t.co/2FJdMaiHnO	2022-07-13 18:59:30+00:00
QuinlivanTD	@CllrPioSmith @davidtobin100 Yes I saw that and it,Äôs very welcome- but what have the lead	2022-07-13 18:44:02+00:00
QuinlivanTD	Simply appalling and totally unacceptable. Critical failure of leadership in the Unionist/ Loyalist	2022-07-13 18:26:58+00:00
QuinlivanTD	RT @MCCaher: Social dancing in the Millennium Centre on Saturday 23rd¬†July. Music by Rober	2022-07-13 16:05:31+00:00
QuinlivanTD	RT @orourke_darren: ?üîå Sinn FV©in,Äôs Green Hydrogen Bill will be debated at 2nd Stage	2022-07-13 14:53:18+00:00
QuinlivanTD	As this is why I,Äôm a proud member of @sinnfeinireland and a part of a national effort to buil	2022-07-13 14:39:29+00:00
QuinlivanTD	RT @IrelandsFuture: ?üö®TICKETS NOW AVAILABLE	2022-07-13 10:21:00+00:00
QuinlivanTD	@angevf Yes I spoke on it!	2022-07-12 21:14:26+00:00

Figure 3. 1: Sample tweet data extracted from Twitter API

Voting Data:

A crucial part of this project was to acquire the voting data of the TDs. The Oireachtas' website publishes data on all the debates and measures that have been discussed and voted on in the Parliament for both the Dáil Éireann and Seanad (upper house of the Parliament). It has the motion title, the motion text or the debate transcript and the details of how each member voted for that motion. All the motions for the time period January 1, 2022 to June 30, 2022 were considered for this project. The data had to be acquired directly from the website using web scraping techniques. This was done using Python's Beautiful Soup package. This package allows to easily parse html data and extract only the data that's required for the analysis. This extracted motion data was again saved in a csv file for further use. The columns scraped from the website were 'deputy, 'motion_date', 'motion_title' and 'vote'. The username column was added to the voting dataset from the file containing the deputy names and corresponding Twitter usernames using excel lookup function, resulting in 5 columns in the final dataset.

Title	Date	Outcome	Yes	No	Abstain
Offences against the State (Amendment) Ac	29-Jun-22	Lost	['Bacik, Ivana.', 'Barry, Mick.'	['Berry, Cathal.', 'Brophy, Colm.',	['Andrews, Chris.', 'Brad
Offences against the State (Amendment) Ac	29-Jun-22	Carried	['Bacik, Ivana.', 'Berry, Cathal	['Barry, Mick.', 'Boyd Barrett, Ric	['Andrews, Chris.', 'Brad
Criminal Justice (Amendment) Act 2009: Mo	29-Jun-22	Carried	['Bacik, Ivana.', 'Berry, Cathal	['Barry, Mick.', 'Boyd Barrett, Ric	['Andrews, Chris.', 'Brad
European Council Decision: Motion (Resume	29-Jun-22	Carried	['Andrews, Chris.', 'Bacik, Iva	['Barry, Mick.', 'Boyd Barrett, Ric	[]
An tOrd Gnv≥ - Order of Business	28-Jun-22	Carried	['Brophy, Colm.', 'Browne, Ja	['Andrews, Chris.', 'Bacik, Ivana.	[]
Our Lady's Hospital Navan Emergency Servic	22-Jun-22	Lost	['Andrews, Chris.', 'Bacik, Iva	['Brophy, Colm.', 'Browne, Jame	[]
Our Lady's Hospital Navan Emergency Servic	22-Jun-22	Carried	['Brophy, Colm.', 'Browne, Ja	['Andrews, Chris.', 'Bacik, Ivana.	[]
Our Lady's Hospital Navan Emergency Servic	22-Jun-22	Carried	['Brophy, Colm.', 'Browne, Ja	['Andrews, Chris.', 'Bacik, Ivana.	[]
An tOrd Gnv≥ - Order of Business	28-Jun-22	Carried	['Berry, Cathal.', 'Brophy, Col	['Andrews, Chris.', 'Bacik, Ivana.	[]

Figure 3. 2: Motion data sample as extracted from Oireachtas website

councillor	username	motion_date	motion	vote
McGuinness, John	JMcGuinnessTD	15-Jun-22	Electoral Reform Bill 2022: Instruction to Committee (Resumed)	yes
McGuinness, John	JMcGuinnessTD	15-Jun-22	Electoral Reform Bill 2022: Report and Final Stages	yes
McGuinness, John	JMcGuinnessTD	15-Jun-22	Special Educational Needs School Places: Motion (Resumed) [Pri	yes
McGuinness, John	JMcGuinnessTD	15-Jun-22	Special Educational Needs School Places: Motion (Resumed) [Pri	yes
McGuinness, John	JMcGuinnessTD	15-Jun-22	Energy Security: Motion (Resumed) [Private Members]	yes
McGuinness, John	JMcGuinnessTD	01-Jun-22	Circular Economy, Waste Management (Amendment) and Miner	no
McGuinness, John	JMcGuinnessTD	01-Jun-22	Property Services (Land Price Register) Bill 2021: Second Stage (F	no
McGuinness, John	JMcGuinnessTD	27-Apr-22	Birth Information and Tracing Bill 2022: Report Stage (Resumed)	no
McGuinness, John	JMcGuinnessTD	27-Apr-22	Birth Information and Tracing Bill 2022: Report Stage (Resumed)	yes
McGuinness, John	JMcGuinnessTD	27-Apr-22	Financial Resolution No. 2: Mineral Oils Tax	no
McGuinness, John	JMcGuinnessTD	27-Apr-22	Financial Resolution No. 2: Mineral Oils Tax	no

Figure 3. 3: Motion data sample after transformation data to right form

The figures 3.2 and 3.3 show the sample of the data extracted from the Oireachtas website. Figure 3.2 shows the crude data obtained from html parsing. The 'Yes', 'No' and 'Abstain' columns had the names of all the deputies that had voted 'Yes', 'No' or 'Abstain' respectively for every motion. This data had to be transformed in a form that had each deputy name as a separate row. This was done using data transformation techniques in Python.

3.1.2 Dataset Description:

The Twitter dataset consists of 392,862 rows of tweet data. The columns considered for this study were the 'User', 'Tweet' and 'Created At'. A description of the columns has been given in the table below:

Column	**Description**
User	Twitter screen name (username) of the user.
Tweet	Full text of the tweet.
Created At	Tweet creation date.

Table 3. 1: Description of columns in Twitter dataset

The motion voting dataset consists of 9263 rows of data. the table below (Table 3.5) gives the description of the columns extracted and considered for this experiment:

Column	Description
deputy	Name of the TD.
username	Username of the TD.
motion_date	Date of the motion vote.
motion_title	The title of the motion.
vote	The TDs vote for the motion.

Table 3. 2: Description of columns in motion dataset

3.1.3 Data pre-processing:

The data obtained from these websites is very noisy. Before this data can be used to train the machine learning models it needs to be cleaned and processed. This involves applying various data pre-processing techniques. This is an important step in order to achieve results with good accuracy. The model needs to be trained on only the important features of the data to improve efficiency and obtain meaningful results. The steps taken for data pre-processing on the two datasets have been explained below. The pre-processing techniques were employed on the 'Tweet' column of Twitter data and the 'motion' column of the voting data.

Punctuation:

Punctuation marks don't contain essential information. This is redundant data. Punctuations were removed from both the columns. This was done using regex in Python. The 're' module has functionalities for regular expression matching. The expression '[^\w\s]+' removes any character that is not a word character or whitespace.

Tokenization:

Tokenization is a common step in text analytics. It refers to splitting a document, sentence, phrase or words into smaller units (tokens) as needed by the application. In this case, the tweets and motion title needed to be split into words. Python's ntlk library was used for this purpose. The tokenize() module of ntlk has functionalities to tokenize the data. It can either tokenise a sentence into words using word_tokenize() or a paragraph into sentences using sent_tokenize(). The word_tokenize() function was used to get a list of words for each tweet and each motion title. This data can now be used as an input for further data processing like removal of stop words.

Stop words:

Stop words are words that are frequently used in the English language. These include pronouns, conjunctions, articles, etc. such as a, the. These words are part of the grammatical sentence structure but do not add much information to the sentence in the context of processing for ML models. Removing these words helps in getting rid of unwanted text features that don't add meaning to the data. It is just noise that increases the data size and processing time and may end up skewing the model training towards unwanted results. The ntlk library of Python has a pre-defined list of stop words in 16 different languages. This list of stop words defined in the corpus module was used to remove all the stop words.

ntlk requires a list of words as input. This requires tokenisation before stop words removal. The list of words is matched against the list of words in the stopwords module in corpus and removed if it is present in the corpus list. This list of pre-defined stop words can be accessed using ntlk.corpus.stopwords.

Conversion to lowercase:

Python is a case sensitive language. This means that 'Hello' would be treated differently from 'hello'. The data is hence converted to all lowercase to enable accurate comparison in the later stages of the implementation. The lower() function was used on the list of words obtained from the previous step to convert all the words into lowercase.

Tweet specific pre-processing:

Since the tweet text has some special data that adds noise to the data and does not add any meaning, some additional pre-processing had to be performed on the 'Tweet' column in the Twitter dataset. In case the tweet was a retweet, the data includes the retweet tag 'RT' to identify it as a retweet. The 're' module of Python was used to remove all the 'RT's from the text. For the purpose of this study, equal importance was given to both the tweets posted by the deputy and the tweets that were retweeted by the deputy. This was done based on the assumption that if a politician retweets something it would mean that he supports the text it contains and is a representation of his position on the matter. It should be noted though that Twitter allows the addition of your own comments to the post before retweeting. This means that retweeting something does not always mean support for the text in the retweet as the politician might have added his own comments that are against the opinion voiced in the retweet. Removal of the RT tags does not address these specific cases.

The tweet texts also contained URLs, emojis, hashtags and twitter handles of the mentions in the tweet. These do not add much meaning to the data. To reduce the size and complexity of the data these were removed using preprocessor module of tweet-preprocessor. preprocessor.set_options was used to specify the data that needed to be removed. Preprocessor runs through all options by default unless certain options have been specified. The preprocessor.clean() function is then used to remove the data that has been specified in the set_options. In this case the options specified were OPT.URL, OPT.EMOJI, OPT.MENTION and OPT.HASHTAG.

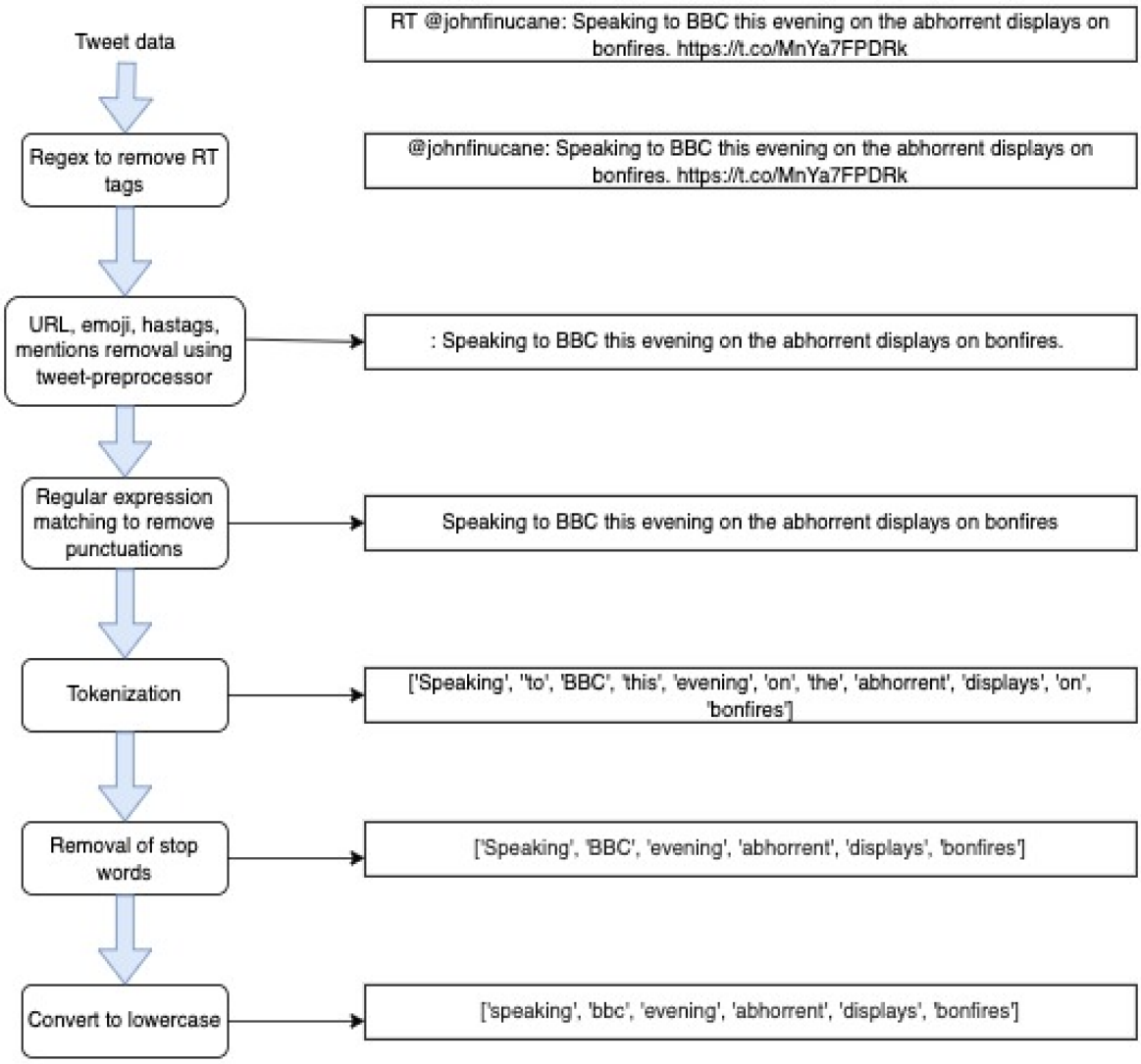


Figure 3. 4: Twitter data pre-processing flow diagram

3.1.4 Data alignment:

To use the Twitter data to predict the voting behaviour of the deputy in future motions, the Twitter and motion voting datasets had to be combined first. The data was brought together such that for every motion and deputy from the voting dataset should align with all the tweets that the deputy had tweeted before the date of the motion. The resultant dataset should have all the tweets that the TD had tweeted before the date of the motion aligned with each motion, for every TD under consideration. Every row in the aligned dataset would correspond to every tweet that the deputy has tweeted before the date of the motion, for every single motion that they voted on.

To do this, first the ‘motion_date’ and ‘Created At’ columns from the voting data and Twitter data were brought into the same datetime format. The datasets were imported as dataframes in Python. This resulted in two dataframes: one containing the tweets of all the TDs and the other containing the motion voting data of all the TDs. To align these two dataframes a row by row comparison of the tweet date and username with the motion date and username was carried out. For every row in the twitter dataframe, the usernames in the two dataframes were first compared. If this was found to be the same then the tweet date and the motion date was compared. If the tweet date was before the motion date then the tweet data was appended to the motion data row and this was added as a new row in the resulting

dataframe. For instance, if the motion date was suppose January 26, 2022. Then all the tweets that a particular deputy had posted before this date would get added as a new row along with the motion data in the resulting dataframe. If the deputy had 100 tweets before this date then there would be 100 rows of data, one row for every tweet. The resulting row would have the columns from both dataframes, that is, 'User', 'Tweet', 'Created At' from Twitter dataset and 'deputy', 'motion_date', 'motion_title' and 'vote' from the motion dataset. This resulted in a fully aligned dataset. This operation was done using itertuples function of Python, which proved to be very time consuming.

The initial datasets had twitter data from February 2019 to June 2022 with 298,881 rows and motion data from February 2020 (when the current 33rd Dáil Eireann members were elected to power) to June 2022 with 40,514 rows, resulting in approximately 12,108,864,834 row by row comparisons as every row in each dataset had to be compared to each other. This would have required computational resources beyond the scope of what was currently available. The computational complexity was reduced by comparing the time taken with different subsets of data and finally, only the data from January 2022 was considered for the scope of this project due to time and resource constraints.

The final fully aligned dataset had 1,983,938 rows of data.

3.1.5 Feature Engineering:

The quality of the output of the ML models depends on the quality of the input that was fed into it. Garbage-in, garbage-out is a popular notion in machine learning. These inputs to the models are called features. These features are usually in the form of columns of data. As part of any text analytics, it is essential to identify and use the most informative features to get the best results. This is done by the process of feature engineering. The goal of feature engineering is to produce new features or enhance exiting features to ensure proper inputs to the model and enhance accuracy.

The following section describes the feature engineering techniques used in this project.

Text similarity:

The twitter data would be useful in predicting the motion voting behaviour of a deputy only if the two texts are similar in context. A politician's views on climate change would not influence their opinion on marriage laws for example. Thus, adding a feature that gave a measure of the similarity between the tweet text and the motion text would be add valuable information to the dataset. If the two texts have very low similarity, the tweet text would likely not influence the voting pattern of the deputy for that motion.
The Jaccard Index similarity was used to compute this similarity between the two texts. The Jaccard index is essentially an intersection by union ratio of two sets. For two texts it gives the ratio of the number of common words to the total number of words. A higher Jaccard score would indicate greater similarity between the texts.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

The tweet text and motion title texts were in the form of list of strings after the pre-processing. To compute the Jaccard index between the two lists of strings, they were first converted to sets using the set() function. This allowed the use of the intersection() and union() set functions to be used. The Jaccard index score was computed between the tweet text and motion title text for every row in the combined dataset. This new feature was named 'similarity' and added to the dataset as a new column.

Negation count:

The negative words used in a sentence gives an idea about the person's point of view. It is helpful when analysing if the person is for or against the text being considered. A negative word count in the texts would thus be helpful in making predictions. The idea behind this is that when comparing the tweet texts and motion title texts, if the texts are highly similar with good textual overlap, in that case if the number of negation words in the tweet text is higher than the negation words in the motion texts, it can be safe to conclude that the deputy is against the said motion. As there was no pre-defined library that contained a list of negation words, a list was manually created. This list contained negation words from the English language such as not, never, hardly, seldom, exclude, neither, etc. This curated list of words was used to get the count of negated words in the tokenized tweet text and the tokenized motion title text. The negation count feature was added to the dataset as two new columns: tweet_neg_count and motion_neg_count.

Baseline model features:

The motion dataset containing just the data from the motions and how the deputies voted was going to be used as a baseline for comparison against the combined dataset model. Some feature engineering techniques were applied to this voting dataset to improve its efficiency in making predictions. A new column most_common_vote was added to the motion dataset. This column contained the most common vote of the deputy from all the past motions, for every deputy. The possible options in a Parliamentary motion vote are 'Yes', 'No' or 'Abstain'. For every deputy the most_common_vote column held the most frequently occuring vote by that deputy in the past. If the deputy had voted 'Yes' in most of the motions, then this could indicate that they are likely to vote 'Yes' in the future motions too. This was achieved in python by grouping the rows by the deputy names and then using Pandas series mode operation on the vote column to determine the most common vote for that deputy.

The other column that was added to the motion dataset was the last_vote column. This column was populated from the voting behaviour of the deputy in the last motion. The basic principle here was that if there is some pattern of behaviour in the deputy's voting behaviour where they had become more agreeable or disagreeable during a period of time, then they might vote the same way in the next vote. The data was sorted first based on the deputy username and then the motion date. The new column then contained the vote from the previous motion. As an example, if the motion under consideration was dated 21 February 2022 and the deputy voted 'No' for the motion dated immediately before this date say on 31 January 2022, then the last vote column would be populated with 'No' for the motion dated 21 February. This was implemented in Python by using the dataframe shift() function on the vote column.

3.1.6 Data Imbalance:

In order to get insight into the kind of data we are dealing with a histogram of the voting data was plotted. The graph shows how each of the deputies voted in all the motions. It can be seen that the data is fairly skewed toward 'Yes' and 'No'. The number of 'Abstain' votes is significantly less as compared to the other two. This was noted to be taken into consideration when tuning the hyperparameters for the classifier models.

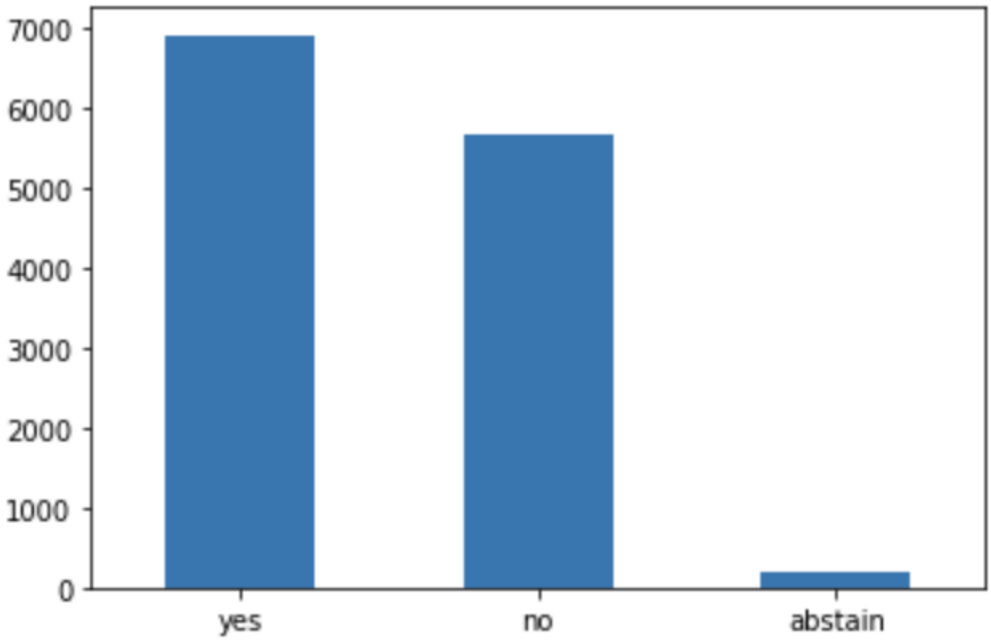


Figure 3. 5: Data distribution of voting output of TDs in the motion dataset

3.1.7 Data splitting:

The splitting of the dataset into train-test datasets is an essential step in machine learning. This allows for evaluation of the model performance. Train-test split involves splitting the data into two subsets of data. The training data that would be used to train the model and the test data which is used to evaluate the performance of the model on new data that it has not seen during training. This evaluation is done by comparing the output prediction of the model for the test data with the actual target variable value.

For this project the baseline and combined datasets were both split into train-validate-test data. This is done to avoid over fitting of the model. This mechanism gives a measure of the generalization of the model. A good model should be able to generalize well for any new input that is fed into it and still get good accuracy in making predictions. The model training is done using the training dataset. The validation data is used to gauge the performance of the model. It is useful when tuning the hyperparameters as the accuracy for validation data gives an unbiased view of the model performance. It is also useful to decide the input features of the model or tune other model configurations. For every epoch, the model is trained using the training data and then evaluated using the validation data. All these go towards tuning the model to attain the best performance possible. The test data is completely unseen data that hasn't been used anywhere before during model training or tuning. This new unseen data is fed into the final model to get a completely unbiased performance measure of the model. It is an indicator of how well the model is generalizing.

For this project, the Twitter data from January 1, 2022 to June 1, 2022 was used as training data and the data for the month of June was used as test data. For the motion voting dataset the training data was from January 1, 2022 to June 15, 2022. The test data was all the data between June 15, 2022 to June 30, 2022. This is because for the motion data, there were 5062 rows of voting data just between June 15 to June 30 as compared to 3671 rows of data between January 1 to June 15. This training data and test data were stored as separate csv files. Python's sklearn's train_test_split () was used to split the data into training and validation data. It is first imported from sklearn.model_selection. This function takes the X and y data as input along with the test size. Here X is the input data containing the features and y is the target variable that we are trying to predict. In this case the target variable y is the vote column

taking the possible values ‘Yes’, ‘No’ or ‘Abstain’. The test size was specified as 0.2. This splits the data into 80% as training data and 20% for validation data. The training data file was split in this manner. The test data was imported as a separate dataframe from the csv file and left as is.

Chapter 4

Methodology

After data collection, pre-processing and splitting into train test validation, the next step is to prepare the data to be fed into the ML models. This is a classification problem and the data needed to be formatted as per the requirement. In order to get the data into the right format to be fed into the ML model, appropriate NLP techniques had to be applied. The following section describes this.

4.1 Target labelling:

The algorithms for machine learning classifiers deal with numeric data. The target variable in this case was the vote column that contains text data as 'Yes', 'No' or 'Abstain'. A target_label function was defined that mapped the text labels to a number and this was added as the new 'target' column in the training and test datasets for both the combined dataset and the motion dataset. The target variable held the value 0 for 'Abstain', 1 for 'No' and 2 for 'Yes'.

4.2 Vectorization:

Vectorization is an important concept in NLP. It is a method of mapping text into a vector of numbers. This is important as the ML algorithms support numeric inputs and not text inputs. There are various ways in Python to convert words into numbers. For this project, CountVectorizer() was selected to perform this vectorization. CountVectorizer function is imported from the sklearn.feature_extraction.text module. The fit_transform() function of CountVectorizer performs the actual data transformation based on the options set using the CountVectorizer function. CountVectorizer takes text input and maps it into a vector where each word is represented by its frequency of occurrence in the text. For example, given a text as input:

text = ['hello this is sample data sample', 'this is the second statement'].

In this case the CountVectorizer will have two text inputs. Each input is automatically pre-processed and tokenized and then represented as a matrix of numbers. It converts everything to lowercase and tokenizes at word level by default. The output in this example is shown below:

hello	this	is	sample	data	the	second	statement
1	1	1	2	1	0	0	0
0	1	1	0	0	1	1	1

Figure 4. 1: Sample output of CountVectorizer()

This matrix is created by taking each unique word in the whole input text as an individual column. For the list of strings, each string is one row in the matrix. The value of a cell is the count of the number of times that word occurs in the string.

For the motion dataset, CountVectorizer was run on the motion title column. For the combined dataset, CountVectorizer was implemented on the motion title and tweet text column. For the combined dataset the max_features option of CountVectorizer had to be used on the tweet text and motion title text columns, to restrict the number of features due to excessive usage of RAM otherwise. This is because the matrix generated from vectorization of the tweets and the motion title was very big. For the logistic regression classifier the max_features was set to 500 for both columns. For decision tree and random forest the max_features was set to 300 for both columns.

4.3 Combining features with DataFrameMapper:

In this work, two different datasets are being considered. Each dataset will be trained and evaluated by ML models. One dataset contains only the councillor username, the motion title and date and their vote. This dataset is referred to as the motion dataset and is used as a baseline for comparison. The other dataset is the combined dataset which contains the Twitter data aligned with the motion data as discussed previously. For the combined dataset, all the features are used as input to the model and predictions are obtained. But for the baseline motion dataset, the features were fed individually into the model to obtain the performance with a single feature as input and also, all features together were used as input to get the model performance.

When only one feature is being used as input to the model, it can be used directly if it is in the right format. After vectorization of a text column, the transformed text is now a matrix. If only one feature is being considered, then this vectorized data can be used as the input to the model. But to combine this feature with other text or non-text features and feed all the features as an input to the model requires special processing. The DataFrameMapper is a module in python's sklearn-pandas library. With DataFrameMapper pre-processing techniques can be applied to multiple column features and these features are then grouped together as a single object. This object is then used as the input to the classifier models. The DataFrameMapper takes a list of tuples as its input. Each tuple represents a column in the dataset that we want to use as a feature input to the model. Column name is used as the first element. The second element specifies the pre-processing technique that needs to be applied to that column to get it into the right format for the classifier input.

The transformations that were applied to each column for the two datasets is given below:

motion_title: CountVectorizer()
tweet: CountVectorizer()

most_common_vote: CategoricalImputer(strategy='constant', fill_value='unknown') followed by LabelBinarizer
last_vote CategoricalImputer(strategy='constant', fill_value='unknown') followed by LabelBinarizer()
username CategoricalImputer(strategy='constant', fill_value='unknown') followed by LabelBinarizer()
similarity: None
tweet_neg_count: None
motion_neg_count: None

The CategoricalImputer function used here replaces missing values in the data with the text specified in the fill_value. The strategy used is constant which means that all missing values will be replaced by the constant text specified. The other option would have been to replace it with the most frequent value. Once the missing values are filled, the LabelBinarizer then maps each label in the categorical data to a unique number.

The None keyword is used in DataFrameMapper when no data transformation needs to be performed for that particular column.

Thus, for the motion dataset the motion_title, most_common_vote, last_vote and username columns were passed as columns to the DataFrameMapper. The model was evaluated one feature at a time and also for all the features. For the combined dataset the motion_title, tweet, similarity, tweet_neg_count and motion_neg_count were passed as inputs to the mapper.

4.4 Modelling:

3 ML algorithms were implemented on both the baseline motion dataset and the combined dataset. The algorithms selected for this experiment were Multinomial Logistic Regression, Decision Tree Classifier and Random Forest Classifier.

4.4.1 Multinomial Logistic Regression:

Logistic regression is normally used for binary classification where the target variable can take only two values. By default, logistic regression cannot be used for multi-class classification, that is, problems that have more than 2 values for the target variable. But in our case the target variable had 3 possible values: Yes, No or Abstain. One strategy to implement logistic regression on a multi-class problem is one-vs-all. In this strategy the multi-class problem is converted to multiple binary classification problems and logistic regression is applied on these subsets. However an easier alternative is the multinomial logistic regression which natively provides support for multi-class problems. Multinomial logistic regression is an extension of logistic regression that allows for multi-class classification. It changes the loss function to cross-entropy loss and the output is a probability for each class variable instead of a single probability prediction.

Logistic regression is implemented using sklearn's library. The model LogisticRegression is imported from sklearn.linear_model module. This standard logistic regression model can be configured as a multinomial logistic regression model by setting the multi_class argument to multinomial.

model = LogisticRegression (multi_class='multinomial', penalty='l2', solver='lbfgs', max_iter=50000)

The penalty term penalizes high values of the cross-entropy loss function. It is a form of regularization to account for overfitting problems.

The L2 penalty is given by:

$$L2\ Penalty = \theta^T\theta = \sum_{j=1}^{n} \theta_j^2$$, where θ is the cost function.

max_iter specifies the maximum number of iterations for the algorithm to converge.

The hyperparameter value C is selected using k-fold cross validation method which is explained further in this report. This hyperparameter value has an inverse relationship with the penalty term. As C increases, the penalty becomes less significant, increasing the value of parameters.

4.4.2 Decision Tree Classifier:

A decision tree algorithm uses a set of rules to reach a final decision. It is a supervised learning technique. The working of the algorithm uses the logic of splitting of the dataset using yes/no questions and keep splitting until the data points for each target class is separate. This results in the formation of a tree structure. The last nodes in this tree are the leaf node and the first node, which is the whole training dataset, is the root node. A label is then assigned to each leaf node. This is done using the most common label for the data points in that node. The evaluation of the split is done by computing the loss functions that evaluate node purity. Gini index or entropy can be used to measure the node purity. The gini index gives a measure of the variance. The entropy is a measure of disorder in the node. It is a measure of randomness.

$$Gini = 1 - \sum_{i=1}^{C} p_i^2$$

$$Entropy\ E(S) = \sum_{i=1}^{C} -p_i log_2 p_i$$

Decision tree algorithm is implemented in Python using DecisionTreeClassifier that needs to be imported from the sklearn.tree module. The parameters ‘criterion’, ‘max_depth’, ‘max_features’ and ‘splitter’ were then computed by hyperparameter tuning through GridSearchCV. The default values were assigned for the rest of the parameters.

criterion: Default is gini. This specifies the criteria for measuring the quality of the split.
splitter: specifies the strategy to choose the split. The split can be ‘best’ to choose best split or ‘random’ to choose a random split from the possible options.
max_depth: to specify the maximum allowed depth of the tree
max_features: specifies the number of features to consider when deciding the split.

4.4.3 Random Forest Classifier:

The random forest classifier is essentially an ensemble of decision trees. Each decision tree in the forest predicts an output class. The class with the most votes is then given as the output of the model as the final prediction.

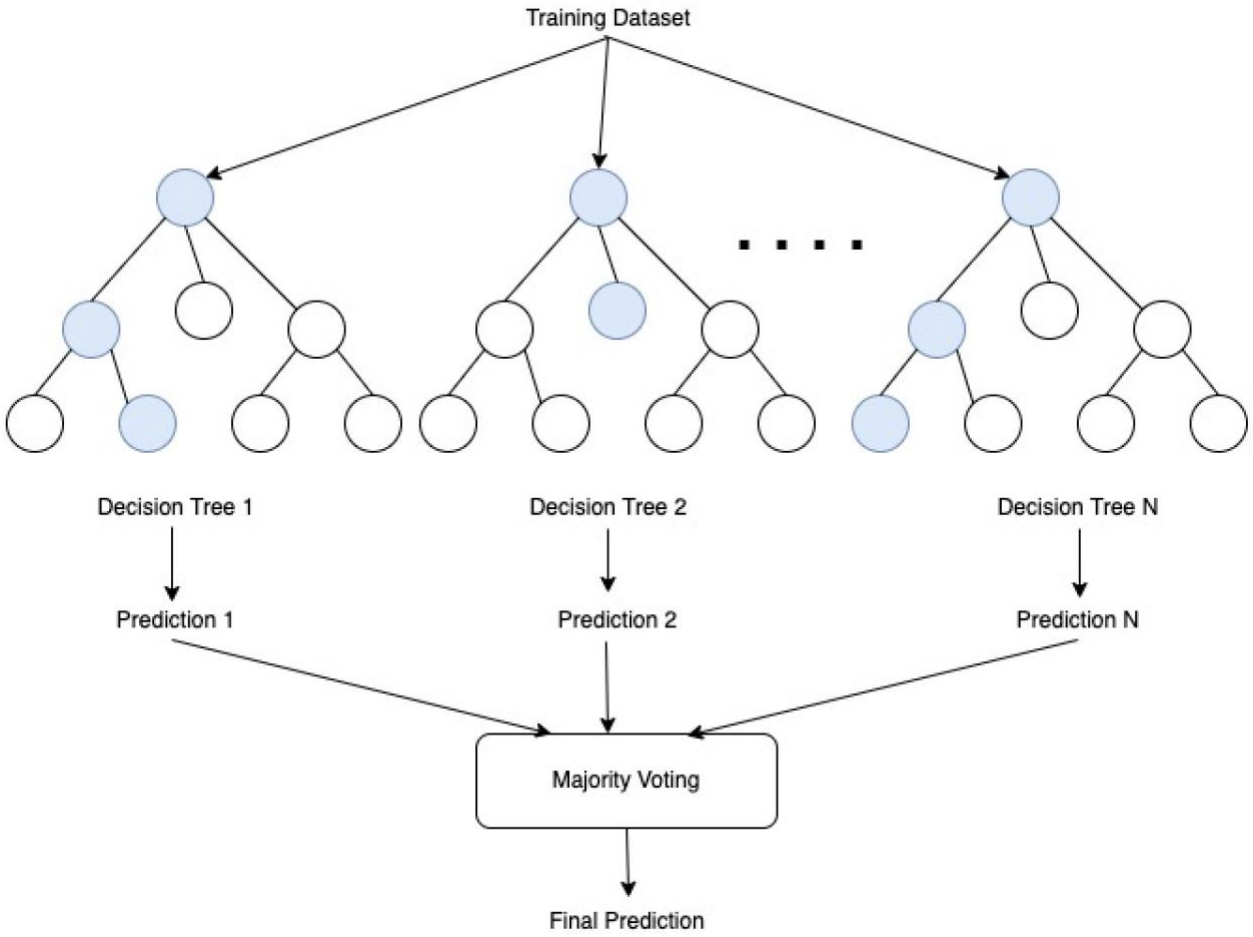


Figure 4. 2: Random Forest algorithm

In order to get good results from the model, the individual trees should have low correlations with each other. In order to ensure this low correlation, random forest employs bagging. Bagging involves random sampling of the dataset with replacement. Suppose the training data is [a, b, c, d] then with bagging in random forest for each tree a random sample of size N (4) is taken from this set, with replacement. [c, d, c, b] could be a resultant dataset for a tree. This results in different trees and ensures that no two trees are too correlated. Additionally, when considering which features to use for a split, random forest can only pick from a random subset of the available features and not every available feature. This also leads to more variance in the trees. Random forests make up for over-fitting that occurs with decision trees and generally results in more accuracy. However it is slower than decision tree algorithm as it needs to compute multiple decision trees.

Random forest algorithm is implemented using sklearn's RandomForestClassifier. This class is imported from sklearn.ensemble. For this project, the n_estimators parameter was configured through hyperparameter tuning. n_estimators parameter is used to specify the number of decision trees to be used in the algorithm. The other parameter values such as criterion, max_depth etc were kept at the default values.

4.5 Hyperparameter tuning:

A crucial part of any machine learning problem is to configure the various available parameters. The configuration settings could either boost or reduce the performance. It is not possible to manually set different parameter values and then evaluate and compare the performance for those. There are many hyperparameter tuning methods that have been developed to make this easier. It is a mechanism to test the model performance for a range of possible values and then select the best ones. These tuning mechanisms can be used to select any model configuration question such as choosing the best feature, the number of trees in a random forest or the maximum depth of the tree, etc. The hyperparameter techniques that have been used in this project work have been discussed below.

4.5.1 k-fold Cross Validation:

k-fold cross validation method can be used to evaluate the model performance for a range of different parameter values. The dataset is shuffled and then split into 'k' groups as specified. For each group, that group is held-out as test data and the rest of the data is used to train the model. It is then evaluated against this held-out test data. The accuracy score is then saved and the model is discarded. This is repeated for all 'k' splits of the data. For instance, if k=5, the data would be split into 5 subsets and the output of this cross validation would be 5 evaluation scores.

In this project k=5 was used. This method of k-fold validation was used to choose the hyperparameter value C in the logistic regression classifier and the parameter n_estimators in random forest classifier. The best parameter values obtained as a result were then used to configure the final model. To implement this validation in Python, cross_val_score() was used which is available in the sklearn.model_selection module.

The parameters that were specified to use this function were:
estimator: specifies the model that is to be used to fit the data.
X: the data to be fit.
y: the target variable.
scoring: the metric to be used to evaluate the model performance.
cv: the splitting strategy. If int then it specifies the number of folds in k-fold.

4.5.2 GridSearchCV:

The GridSearchCV is scikit-learn's hyperparameter tuning tool. It is imported from the sklearn.model_selection module. The different parameters with their ranges are specified in a parameter grid. GridSearchCV then runs through all the parameters and gives the best combination of these parameters as the output. The criteria for selection of the best parameter is based on a scoring metric specified. However it should be noted that the 'best parameters' that the GridSearchCV arrives at is only the best amongst the range that was specified in the grid.

GridSearchCV was used in this project to arrive at the best parameters for the decision tree classifier.

The parameters that were specified in the grid are given below:

'criterion': ['gini', 'entropy']

'max_depth': [None, 2, 4, 6, 8, 10]
'max_features': [None, 'sqrt', 'log2', 0.2, 0.4, 0.6, 0.8]
'splitter': ['best', 'random']

The GridSearchCV inputs specified were:

'estimator': specifies the model.
'param_grid': specifies the parameter grid to be used.
'cv': determines the cross-validation strategy.
'n_jobs': specifies the number of jobs to be run in parallel.

4.6 Evaluation metrics:

Evaluation metrics are important when analysing the performance of the model. These metrics are used when selecting the best set of parameters during hyperparameter tuning and also to compare the performance of the 3 models used in this experiment. The following section describes the evaluation metrics that were used in this project for hyperparameter selection and model performance comparison.

4.6.1 Accuracy score:

The accuracy score gives a measure of the number of correct predictions against the total number of predictions.

$$Accuracy = \frac{Correct\ Predictions}{Total\ Predictions}$$

For binary classification accuracy can be explained in terms of positives and negatives. True Positives (TP) are values that were predicted as True when the actual class label was also True. True Negatives (TN) are values that were predicted as False when the actual class label was also False. False Positive (FP) is when the prediction was True but the actual class label was False. False Negative (FN) is predicting False when the actual label is True.

$$Accuracy\ score = \frac{TP + TN}{TN + TP + FN + FP}$$

In python the accuracy score of a model can be computed using sklearn.metrics.accuracy_score(). This function compares the true value and the predicted value for the target variable, for all predictions.

4.6.2 F1 score:

The F1 score is a useful metric for classification models. It also works on imbalanced datasets where the target variable are skewed towards one value. It is a combination of both the precision and the recall of a model. F1 score is computed as the harmonic mean of the precision and recall.

$$F1\ score = \frac{2TP}{2TP + FN + FP}$$

It gives equal weight to both. A high F1 score means high precision and recall.

The F1 score of the model was used as an evaluation metric during cross validation process. The set of hyperparameter values that produced the highest F1 score was selected. F1 score can be computed for cross validation using sklearn's cross_val_score. The scoring metric parameter for this function was set to 'f1_macro'. In multi-class problems, the F1 score is computed for every class using one-vs-rest approach. To get the average of the score for the model across all classes, different averaging techniques are available: macro, micro and weighted. Macro averaging treats all classes as equally important and computes the average of all scores across all classes. Micro average is computed from the sums of the TP, FN and FP. Weighted averaging method gives different weights to each class when computing the average score. The weight of the class depends on its proportion of occurrence in the dataset. Macro averaging is good to be used in cases of imbalanced dataset and hence was the option of choice in our case as each target variable class was to be treated as equally important.

Chapter 5

Results

This chapter discusses the results obtained after applying the 3 ML classifier models to the motion dataset and the combined dataset. The 3 ML models were applied to the motion dataset with all 4 features as input. The 3 models were also evaluated for the motion dataset taking one feature at a time, for all 4 features. For the combined dataset, all 6 features were given as input to the 3 ML models.

5.1 Dummy classifier:

A baseline dummy classifier was used to evaluate the performance of both the baseline model using motion data and the model trained using combined Twitter and motion data. This dummy classifier was implemented using sklearn's DummyClassifier with strategy = most_frequent. This gives the most common vote as the prediction every single time. This dummy baseline model achieved an accuracy of 0.52 on training data and 0.51 on test data.

5.2 Baseline Model:

The dataset obtained only from extracting the features from the motion data website was used as a baseline. Each feature in the motion dataset was used to train all 3 models individually and then all together. Hyperparameter tuning was done to configure each model properly.

5.3 Multinomial Logistic Classifier:

The hyperparameter value C for the classifier was selected using k-fold cross validation and F1 scoring with macro averaging applied.

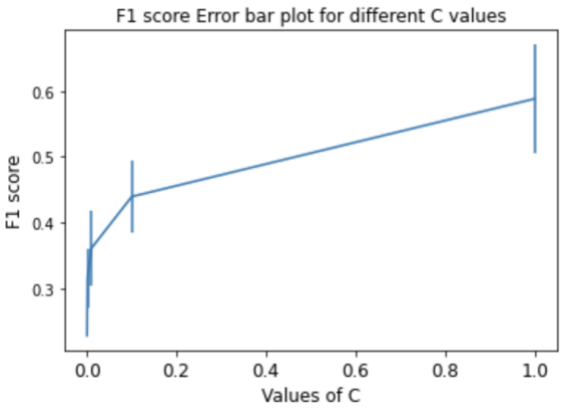
The range of values considered for C were [0.0001, 0.001, 0.01, 0.1, 1]. The hyperparameter tuning was done taking one feature at a time.

The results obtained for the F1 score for different values of C for each feature is given below:

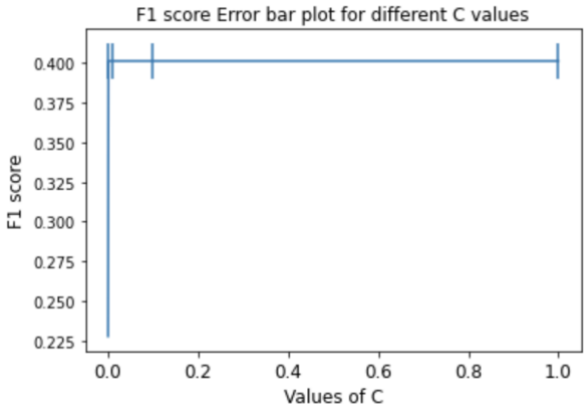
Feature Considered	C = 0.0001	C = 0.001	C = 0.01	C = 0.1	C = 1
motion_title	0.23	0.32	0.36	0.43	0.59
most_common_vote	0.23	0.40	0.40	0.40	0.40
last_vote	0.23	0.46	0.46	0.46	0.46
username	0.23	0.23	0.23	0.22	0.22
All features	0.23	0.46	0.45	0.79	0.76

Table 5. 1: F1 scores of features when trained using Multinomial Logistic Regression for different C values

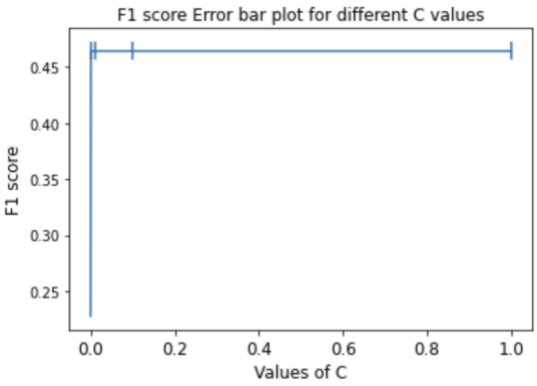
The C value with the highest F1 score was selected for each feature and the model was trained using that C value. The error bar plots were plotted for these F1 scores to select the C value with the highest F1 score and low standard deviation. C = 0.1 was selected to train all models as it had the highest value for all features.



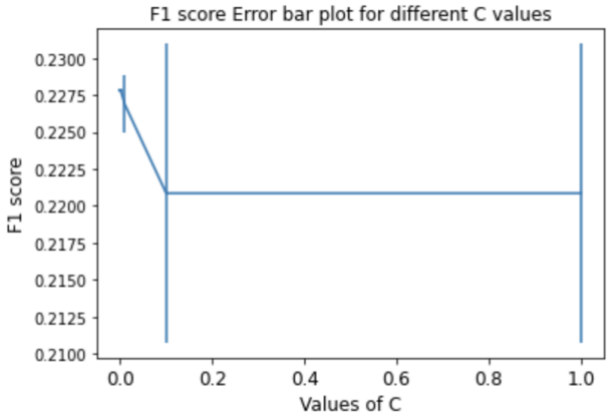
a. With ‘motion_title’ as feature



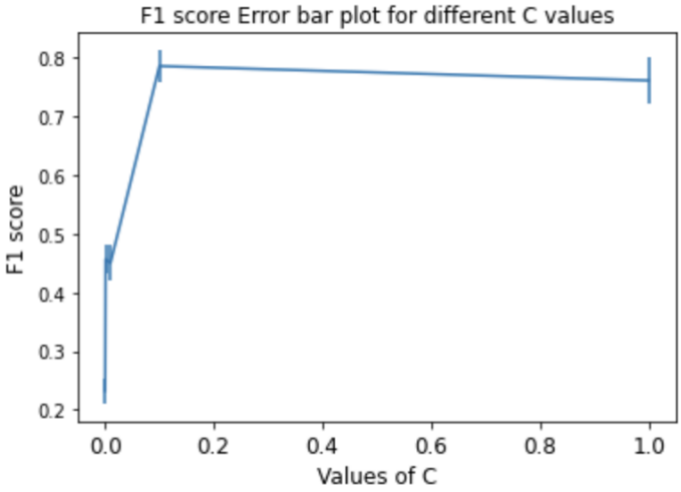
b. With ‘most_common_vote’ as feature



c. With last_vote as feature



d. With username as feature



e. With all features

Figure 5. 1: Error bar plots for F1 scores for hyperparamter tuning of C

The multinomial logistic regression model was then trained using this C value by taking each feature individually and then all features together. The trained models were then tested on unseen test data. The accuracy scores of these models have been given in the Table 5.2.

Feature considered	Training accuracy	Test accuracy
All	0.74	0.70
motion_title	0.55	0.58
most_common_vote	0.60	0.58
last_vote	0.69	0.58
username	0.60	0.59

Table 5. 2: Accuracy scores of features when trained using Multinomial Logistic Regression

It can be seen that the model that used all 4 features together as input performed the best out of all. The accuracy of the models using only one feature each were similar to each other, with no individual feature standing out as giving significantly higher performance. The model obtained an accuracy of about 0.70

on unseen test data when all features were used together. When only a single feature was used to train the model, the accuracy dropped to around 0.58 on test data for all the features.

5.4 Decision Tree Classifier:

For decision tree classifier the parameter tuning was done using GridSearchCV. The parameters were tuned using 5 fold cross validation. The model was trained taking one feature at a time and all features together. The best parameter values obtained for each feature is shown below.

Feature considered	criterion	max_depth	max_features	splitter
All	entropy	None	None	best
motion_title	entropy	2	0.4	best
most_common_vote	gini	None	None	best
last_vote	gini	None	None	best
username	gini	None	log2	Best

Table 5. 3: Best parameter values for features when trained using Decision Tree classifier

These best parameter values obtained were used to train the decision tree classifier model taking one feature at a time and then all features together. The performance of this model is given in Table 5.4.

Feature considered	Training accuracy	Test accuracy
All	0.53	0.50
motion_title	0.56	0.58
most_common_vote	0.59	0.58
last_vote	0.68	0.58
username	0.60	0.59

Table 5. 4: Accuracy scores of features when trained using Decision Tree classifier

In this case, the model that trained using all the features performed the worst with only 0.50 accuracy on test data which isn't even better than the dummy classifier baseline model. The models trained using only one feature at a time performed better with accuracy of 0.58. The model trained using only username of deputy as feature performed the best with 0.59 accuracy. Overall, the decision tree model performed poorly as compared to the multinomial logistic regression model.

5.5 Random Forest Classifier:

The n_estimators parameter was tuned using k-fold validation method. The model was trained taking one feature at a time and all features together. F1 scoring with macro averaging was used as an evaluation metric.

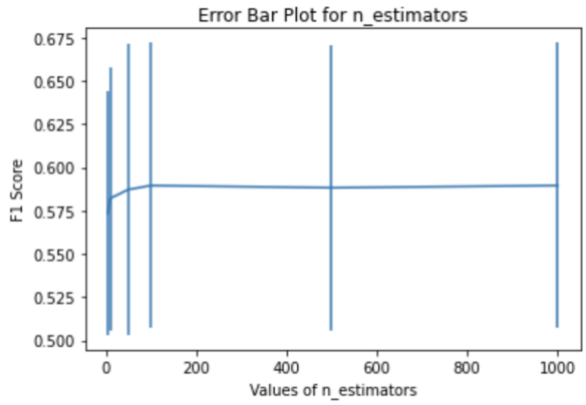
The range of values considered for n were [5, 10, 50, 100, 500, 1000]. The hyperparameter tuning was done taking one feature at a time.

The results obtained for the F1 score for different values of C for each feature is given below:

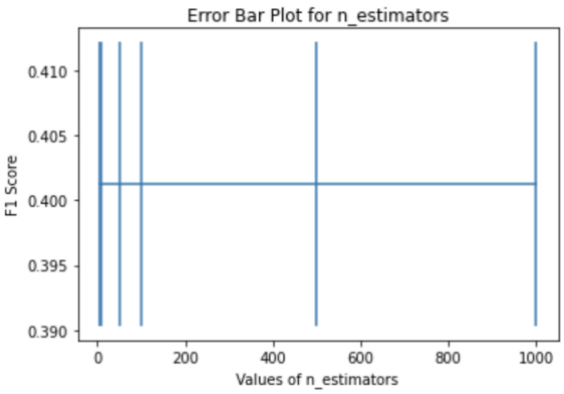
Feature Considered	n = 5	n = 10	n = 50	n = 100	n = 500	n = 1000
motion_title	0.57	0.58	0.59	0.59	0.59	0.59
most_common_vote	0.40	0.40	0.40	0.40	0.40	0.40
last_vote	0.46	0.46	0.46	0.46	0.46	0.46
username	0.21	0.22	0.21	0.21	0.21	0.22
All features	0.95	0.94	0.95	0.95	0.95	0.95

Table 5. 5: F1 scores of features for different values of n_estimators for Random Forest classifier

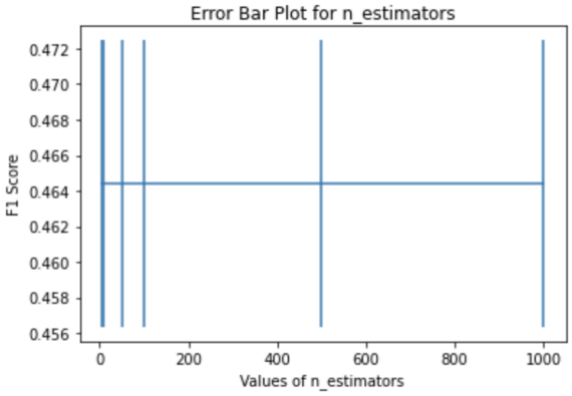
The n value with the highest F1 score was selected for each feature and the model was trained using that n_estimator value. The error bar plots were plotted for these F1 scores to select the n value with the highest F1 score and low standard deviation. There wasn't any improvement in the F1 score for values of n greater than 50. So n_estimators = 50 was selected to train all the models.

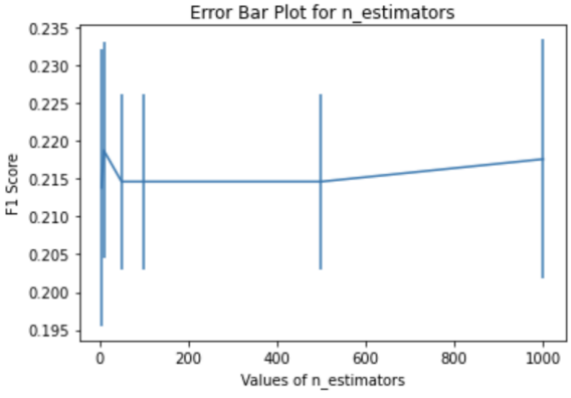


a. With 'motion_title' as feature



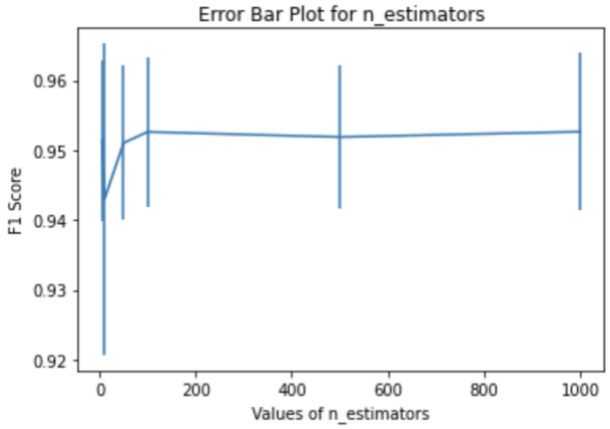
b. With 'most_common_vote' as feature





c. With last_vote as feature

d. With username as feature



e. With all features

Figure 5. 2: Error bar plots for F1 scores for hyperparameter tuning of n_estimators

The random forest model was then trained using this n value by taking each feature individually and then all features together. The trained models were then tested on unseen test data. The accuracy scores of these models have been given in the Table 5.6.

Feature considered	Training accuracy	Test accuracy
All	0.97	0.92
motion_title	0.55	0.58
most_common_vote	0.60	0.58
last_vote	0.68	0.58
username	0.60	0.57

Table 5. 6: Accuracy scores of features when trained using random forest classifier

The random forest model performed the best out of all 3 ML models achieving an accuracy of up to 0.92 on test data. This is significantly higher than the accuracy of the decision tree and logistic regression models. However, when only one feature at a time was considered to train the random forest model then the accuracy fell considerably to 0.58 which is similar to what was achieved using the decision tree and logistic regression models.

The performance of all 3 models trained using the baseline motion dataset have been summarised in Table 5.7.

Algorithm	Feature	Training Accuracy	Test Accuracy
Logistic Regression	All	0.74	0.70
	Motion data	0.55	0.58
	Most common vote	0.60	0.58
	Last vote	0.69	0.58
	username	0.60	0.59
Decision Tree	All	0.53	0.50
	Motion data	0.56	0.58
	Most common vote	0.59	0.58
	Last vote	0.68	0.58
	username	0.60	0.59
Random Forest	All	0.97	0.92
	Motion data	0.55	0.58
	Most common vote	0.60	0.58
	Last vote	0.68	0.58
	username	0.60	0.57

Table 5. 7: Summarised performance results for all 3 models

5.6 Results from Twitter and motion data combined dataset:

Finally, the Twitter and motion data aligned dataset was trained using all 3 ML models. This time the models were trained only using all features together and not individually. This was decided as from the baseline model trained previously it was noted that the models trained using all features together generally performed better.

As before, hyperparameter tuning was done using k-fold cross validation to select C for the logistic regression model. C = 1 was selected for this model. For random forest classifier n_estimators = 50 was selected after k-fold cross validation hyperparameter tuning. GridSearchCV was used to get the best parameter values for the decision tree and criterion = ‘entropy’, max_depth = None, max_features = None and splitter = ‘best’ was used as the configuration.

The results obtained from the 3 models have been tabulated in Table 5.8.

Algorithm	Training Accuracy	Test Accuracy
Multinomial Logistic Regression	0.59	0.58
Decision Tree Classifier	0.67	0.65
Random Forest Classifier	0.90	0.80

Table 5. 8: Accuracy scores for all 3 models when trained using the Twitter and motion data combined dataset

The model performances from the combined dataset (Twitter and motion aligned data) are slightly different than the results obtained from the models trained using just the motion dataset earlier.

Multinomial logistic regression performed the worst this time with only 0.58 accuracy on test data as compared to 0.70 accuracy that was obtained from the motion data baseline model. The random forest classifier again had the best performance with 0.90 accuracy on training data and 0.80 accuracy on test data. The performance of the decision tree model improved significantly from the baseline model when twitter data was added and the accuracy increased from 0.50 to 0.65. It should be noted that while all models performed quite well, the decision tree model was the only one whose performance improved after adding twitter data to the dataset while the performance of the other two models reduced as compared to when only the motion data was used to make predictions.

However, all the models including both the baseline motion data models and the combined data models performed better than the baseline dummy classifier model.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

The motivation behind this research was to get an idea of a politician's legislative voting behaviour from their social media communication to enable the general public to make choice judgements on which politicians they may support. The intent was to gauge whether the social media communications of a politician is a reliable indicator of their actual political position on various matters of public interest. The aim of the research was to determine if the social media communication of the politicians on Twitter could be used to predict how they would vote for a motion in legislature.

The approach followed to answer this question was to get the voting data and the Twitter data from the corresponding sources and then align the two. The voting behaviour predictions for each deputy was made using this combined data. To get an accurate picture of whether adding the Twitter data was indeed resulting in better voting behaviour predictions, the performance of these models was compared against a baseline model which used only the motion voting data to make predictions without having the Twitter data in the picture. For this research a lot of work went into the data collection, pre-processing, and feature engineering stage. Web scraping skills had to be employed to get the data from the motion voting website, and a significant amount of effort went into extracting the required features from the html data as the information was spanned over different webpages via hyperlinks. Various NLP feature engineering techniques had to be performed on the data as the main features were the motion text and tweet text and extracting the important data from these and converting them to a suitable form to be used for a ML classification problem was not straightforward. Once the final data had been prepared with appropriate feature engineering, the model training and evaluating part was quite undemanding.

The results obtained from this study show that with a proper cleaner dataset and an improved ML model, Twitter data can be used to predict the voting behaviour of the politician. All 3 models achieved good accuracy ranging from 60% to 80%. Although the models that were trained using only the motion data also obtained similarly good accuracy except for the decision tree model. Thus, it cannot be stated that Twitter data is 'improving' the predictions in all cases. But for the decision tree model, the accuracy increased by almost 30% from 0.50 to 0.65 after adding Twitter data to the dataset. Although an accuracy score of 0.65 by itself is not that great especially compared to random forest that achieved a 0.80 accuracy score, it can be said that with a more finely tuned and possibly larger training dataset and a better classifier the prediction accuracy can be improved further.

6.2 Challenges

One of the major challenges in this project work was the huge volume of data both from Twitter as well as the motion voting website. This data was large and noisy, requiring a substantial amount of work towards data preparation. Since Twitter API was used, there were issues with API timeout errors. Because of this, the data had to be extracted for 20 to 30 deputies at a time and stored into multiple csv files which were then concatenated at the end.

Aligning the Twitter data with the motion data was a challenge due to the sheer volume of the data. as it was a row-by-row comparison of two dataframes with 392,862 and 9263 rows each (~3,639,080,706 comparisons). This was a challenging process due to resource and time constraints.

There were limitations in the proper sentiment analysis of the text such as detection of sarcasm. Nuanced emotions like those are hard to detect and misinterpreting such nuances affects the proper classification of the text.

A major limitation of this research work is the lack of the motion text. The motion voting data was scraped directly from the parliament website. Unfortunately, the way the website is designed was not ideal for scraping the motion text. The webpage had a list of all the motions voted on and the overall outcome of the vote as 'lost' or 'carried'. Each individual motion was set up as a link. The motion link routed to a webpage that had details on the motion including how each deputy had voted. But the motion description part was not a constant for all the motions. For some motions, the detailed description part contained the full text of the motion that was debated upon and for some it contained the debate transcript of the parliamentary session instead of the motion text. For this study only the motion text was required and not the debate transcript. But there was no way to distinguish between the two texts through web scraping. So, for the purposes of this research work only the motion title was used as a feature. The drawback of this is that the subject matter that is being voted on is unclear from just the title of the motion, proving to be not as insightful into what the motion is about. For example, a motion titled 'Home Heating Fuels' is about calling on the Government to cancel carbon tax increase and temporarily remove excise duty on home heating oil. But just the title in this case is giving no information about what exactly is being voted on. This is crucial information that's being lost as the research is trying to estimate the politician's political position from their tweets and try and use that information to see how they would vote on in a similar subject matter but that is not possible if the subject that is being voted on is unclear. The model performance would possibly improve if there was access to the motion subject text.

6.3 Future work

As noted previously, this research only considered the title of the motion as a feature input to make predictions. It would be useful to see how the addition of insightful motion text that succinctly summarises the motion text, if added to the data, affects the accuracy of the predictions. This would probably require the manual summarizing of the motions to get a descriptive column detailing the text of the motion in a line or two.

An ablation study on the significance of each feature would be an interesting line of study as an extension of this research. This could be done by training the classifiers with all features and then excluding one feature at a time to see the difference in the performance of the model and gauge whether the exclusion of any specific feature adversely affects the prediction accuracy. Analysis of the positive or negative impact of a feature on the model would be helpful in increasing its performance.

For this research, due to resource constraints only the tweet data from the Twitter API output was considered as a feature. The other metrics like the retweet count, number of likes or comments was not included in the study. Analysing the effect of these features on the accuracy of predictions could be a part of later study. An interesting question that arises from this is: does the popularity of the tweet have an effect on the voting behaviour of the politician? If a politician posts his support for a policy and the public reaction gauged through the comments or number of likes is against the policy, would he be likely to change his vote? Conversely, if a tweet is found to be popular among the public, would that mean the politician would vote in favour of the motion?

This study was limited to the members of the Dáil Eireann but the same approach can be used for any other legislative body members expanding beyond the borders of Ireland. It could be extended over multiple countries or governing bodies. It would be interesting to see if there is variation in the results obtained across different regions of the world and whether that difference is reflective of the culture of the region.

Bibliography

[1] "Votes." https://www.oireachtas.ie/en/debates/votes/?debateType=dail

[2] M. Petrova *et al.*, "Social Media and Political Contributions: The Impact of New Technology on Political Competition."

[3] P. Baranyi, A. Csapo, and P. Varlaki, "An overview of research trends in CogInfoCom," in *INES 2014 - IEEE 18th International Conference on Intelligent Engineering Systems, Proceedings*, Sep. 2014, pp. 181–186. doi: 10.1109/INES.2014.6909365.

[4] C. Vogel and A. Esposito, "Linguistic and Behaviour Interaction Analysis within Cognitive Infocommunications," 2019.

[5] A. Esposito, A. M. Esposito, and C. Vogel, "Needs and challenges in human computer interaction for processing social emotional information," *Pattern Recognition Letters*, vol. 66, pp. 41–51, Nov. 2015, doi: 10.1016/j.patrec.2015.02.013.

[6] M. deVelasco V´azquez, R. Justo, and A. L. Zorrilla, "Can Spontaneous Emotions be Detected from Speech on TV Political Debates?," 2019.

[7] M. Koutsombogera and C. Vogel, "Observing collaboration in small-group interaction," *Multimodal Technologies and Interaction*, vol. 3, no. 3, Sep. 2019, doi: 10.3390/mti3030045.

[8] F. Haider, M. Koutsombogera, O. Conlan, C. Vogel, N. Campbell, and S. Luz, "An Active Data Representation of Videos for Automatic Scoring of Oral Presentation Delivery Skills and Feedback Generation," *Frontiers in Computer Science*, vol. 2, Jan. 2020, doi: 10.3389/fcomp.2020.00001.

[9] C. Vogel, "Multimodal conformity of expression between blog names and content," in *4th IEEE International Conference on Cognitive Infocommunications, CogInfoCom 2013 - Proceedings*, 2013, pp. 23–28. doi: 10.1109/CogInfoCom.2013.6719247.

[10] A. Karampela and C. Vogel, "Nouns and Verbs in Professional Reporting of Extreme Events," 2019.

[11] M. J. Gabel and J. D. Huber, "Putting Parties in Their Place: Inferring Party Left-Right Ideological Positions from Party Manifestos Data," 2000.

[12] M. Laver and J. Garry, "Estimating Policy Positions from Political Texts," 2000.

[13] M. Laver, K. Benoit, and J. Garry, "Extracting Policy Positions from Political Texts Using Words as Data," 2003.

[14] S. van Gijsel and C. Vogel, "Inducing a Cline from Corpora of Political Manifestos."

[15] J. DiGrazia, K. McKelvey, J. Bollen, and F. Rojas, "More tweets, more votes: Social media as a quantitative indicator of political behavior," *PLoS ONE*, vol. 8, no. 11, Nov. 2013, doi: 10.1371/journal.pone.0079449.

[16] L. Yue, W. Chen, X. Li, W. Zuo, and M. Yin, "A survey of sentiment analysis in social media," *Knowledge and Information Systems*, vol. 60, no. 2, pp. 617–663, Aug. 2019, doi: 10.1007/s10115-018-1236-4.

[17] L.-P. Morency, R. Mihalcea, and P. Doshi, “Towards multimodal sentiment analysis: Harvesting opinions from the web,” in *Proceedings of the 13th international conference on multimodal interfaces*, 2011, pp. 169–176.
[18] D. I. Holmes, “The Evolution of Stylometry in Humanities Scholarship,” 1998. [Online]. Available: https://academic.oup.com/dsh/article/13/3/111/933269
[19] D. v Khmelev and F. J. Tweedie, “Using Markov Chains for Identification of Writers,” 2001. [Online]. Available: https://academic.oup.com/dsh/article/16/3/299/930917
[20] A. Mitchell, A. Esposito, and C. Vogel, “Tweet Analytics for Political Position Estimation.”
[21] W. Medhat, A. Hassan, and H. Korashy, “Sentiment analysis algorithms and applications: A survey,” *Ain Shams Engineering Journal*, vol. 5, no. 4, pp. 1093–1113, Dec. 2014, doi: 10.1016/j.asej.2014.04.011.
[22] 909originals, “Here are the Twitter accounts of all the TDs in Leinster House.” https://909originals.com/2021/07/20/twitter-tds-leinster-house/