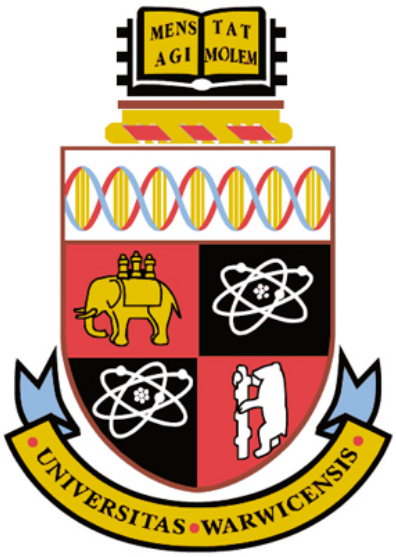


Predicting UK Election Outcome using Political News Sentiment and Constituency Socioeconomic Conditions

by

Seth Odia

Supervised by

Dr. Gihan Mudalige

for the degree of

MSc Computer Science

Department of Computer Science

University of Warwick

September 2020

Abstract

In this project I aim to be able to accurately predict the outcome of UK elections by predicting what party wins in the various Westminster Parliamentary constituencies in England. This will be done using data from news sites and blogs as well as socioeconomic conditions in each constituency around England and derive relationships between these data sources and election results with the use of machine learning. To do this, I am going to make use of various classification models such as AdaBoost, Random Forest, Decision Tree and Support Vector Machine.

Keywords - Classification, Machine Learning, Natural Language Processing, Sentiment Analysis, Correlation, Elections, Politics, Data Mining.

Acknowledgements

I would like to thank and express my deepest gratitude to Dr. Gihan Mudalige for supervising and supporting the development of this project.

Contents

Chapter 1

Introduction

The United Kingdom practices a parliamentary system of government which is a democratic form of government in which the party (or a coalition of parties) with the greatest representation in the parliament (legislature) forms the government, its leader becoming Prime Minister. Executive functions are exercised by members of the parliament appointed by the prime minister to the cabinet. The parties in the minority serve in opposition to the majority and have the duty to challenge it regularly.[3] The UK uses the First Past the Post electoral system to elect members of parliament to Westminster. In this system, during a General Election, 650 constituencies across the UK each hold separate contests. To become an MP, a candidate needs the largest number of votes in their area.[4] Which ever party wins the majority of the 650 constituencies forms the government with their party leader becoming the Prime Minister.

Predicting elections using traditional methods has proven to be problematic as issues arise from its accuracy to the amount of time and effort required to make these predictions. Also, the political views of people across the population have overtime become more diverse and non identical to someone else from a similar background. This can be seen as an effect of the internet age where people consume information from an abundance of diverse sources like social media sites such as Facebook and Twitter and online news and blog sites such as TMZ and Huffington Post.

Such platforms have an ability to influence peoples political views and beliefs and because there are so many of these platforms compared to a number of decades ago where there were only a number of newspaper companies and TV channels, it has led to the public having a more diverse view on political subjects and thus more diverse opinions during elections. This has made making predictions more difficult and inaccurate, and because of this, people such as electorates and candidates end up believing wrong predictions. For the election candidates, wrong predictions can affect their campaigns negatively as for instance, they end up

spending more time campaigning where they believe they are less popular instead of the places where more campaigning will make a difference.

Traditional election prediction methods such as the exit polls which is a survey of thousands of voters just after they have cast their ballot. The exit poll is based on 144 constituencies in England, Scotland and Wales. The constituencies are chosen to be demographically representative of the country, balanced between rural and urban seats, and weighted slightly in favour of marginal areas. Although its accuracy has increased in recent years, it still has a long way to go and this can be seen in the 2015 election where the exit polls although predicting a Conservative win, did not predict a Conservative majority and in 2017, the first take of the exit poll correctly predicted the Conservatives would be the largest party, but stopped short of saying there would be a hung Parliament.[1]

Another traditional method are the opinion polls which provide a snapshot of people's views on a particular issue. During the 2019 election campaign YouGov published a poll based on the views of 100,000 voters, applying national trends to individual constituencies. Over the last few elections, 90% of polls conducted just before the election have been within four percentage points of the vote share for the major parties, but are not even as accurate as the exit polls.[2]

The aim of this project is to obtain correlation between election results and voter sentiment in regards to political news as well as socioeconomic conditions in each constituency around England. From this it it is then possible to ascertain which attributes were the most influential in this election and to predict voting patterns without having to make use of a poll. This project is motivated by the aspiration to further comprehend how media consumption and economic and social attributes influence voting choices.

1.1 Objectives

To achieve the overall aim of the project, it has been broken down in several tasks. These objectives, are reiterated below.

- Collect data of past elections with breakdown of election outcome.
- Scrape online data from news sites involving political parties and candidates.

- Clean data and form datasets with the obtained data
- Make use of Natural Language Processing on news data to derive voter sentiment on candidates and political parties.
- Preprocess dataset attributes to make data more presentable to the model in order to provide better predictions
- Test various models to find which works best for predicting election results in the Westminster Parliamentary constituencies in England.
- Fine tune model parameters to have as low bias as possible while also having as low variance as possible on training data.
- Split datasets into training and test data and train models with data.
- Test models with test data and measure accuracy using metrics such as AUC-ROC, F1 Score, Mean Squared Error.
- Evaluate which features affect election results the most.
- Visualize results using map of England showing output from each constituency

1.2 Changed Objectives

Some of the objectives have been changed as the project have evolved some of these are :

- Predicting UK elections instead of Nigeria elections
- Using news data to get sentiment as an alternative of using twitter data
- Using social and economic constituency factors as attributes along with sentiment rather than just using sentiment alone.

1.3. Resources & Tools

This project relies on using the Python programming language. I chose to use Python because it is easy and flexible to use, versatile and has a robust collection of libraries that make machine learning tools easily available to use. To write this python code I used the Jupiter notebook and Colab applications.

I used various Python libraries to access tools to bring my project to life many of these being machine learning tools such as:

- Sklearn
- Matplotlib
- Pandas
- Numpy

To accumulate online news data, I made use of the BeautifulSoup library a tool which enables one to scrape text from websites. Then to process the text in order to draw out sentiment, the nltk library is used.

Chapter 2

Literature Review

This chapter aims to look at existing literature which is used to form a context for this project. Such literature are based on subjects surrounding the structure of the UK elections , techniques used to make predictions in election results including both traditional methods and more recent methods such as building and training machine learning models and the application of Natural Language Processing to politics.

2.1 Internet Effect on Elections

In recent years, because of the increase in popularity of the internet, studies have been undergone to see how the internet has affected elections. Grofman(2014) talks about how terms such as e-democracy have morphed over the years from speculative and optimistic accounts of a future heightened direct citizen involvement in political decision-making and an increasingly withered state apparatus, to more prosaic investigations of party and governmental website content and micro level analyses of voters' online activities.[5] In this book, he investigates the role of digital media and competitive advantage, campaigns and the effect of social media, online communication as way of fomenting nonviolent revolutions and the undeniable and important role of the internet on democracy around the world. He uses the 2010 national elections in Brazil as a research point due to the fact that an expanding sum total of Brazilian candidates had begun to use websites and social networking applications as an essential part of their overall campaign efforts and findings in his study suggest that certain internet sites can be better suited for political campaign purposes than others and that this most likely varies depending upon the political and electoral systems, as well as culture giving an example of how Twitter was a more suitable media campaign platform compared to Facebook and Orkut. He then ends with stating

that these new media connections between candidates and voters can play a central role in shoring up voter support.

A study by Bimber (2003) shows that the internet facilitates highly selective forms of engagement, we see evidence that through use by campaigns, supporters of candidates more engaged even if the effect is not overwhelming in scale and even if it is primarily online. If at least some politically interested individuals can find a channel for their desire for activism, that would be an attractive development. [7]. This proves that the media has an effect on voter preferences and with the right media coverage during an election a candidate or party has a greater chance of winning an election.

With influential people in politics having some sort of control over the media, and being able to use the media to influence voters, there has been a rise in propaganda used to win support, MacLeod (2019) studies how propaganda is used to influence the public and lifting the veil over how the mass media operates to show how effective it has been with a focus on past events such as fake news, Cambridge Analytica, the Syrian Civil War and RussiaGate.[8]

2.2 Web Scraping

Web Scraping is a technique employed to extract large amounts of data from websites whereby the data is extracted and saved to a local file in your computer or to a database in table format.[12] The internet contains the largest pool of data that can be harnessed and with the use of web scraping tools it is possible to obtain this.Unstructured data is gathered from the web and then converted into more structured datasets perhaps in the from of a table. There are two forms of web scraping which will be discussed below

2.2.1 Application Programming Interfaces

This form of web scraping entails using application programming interfaces, commonly referred to as APIs. This is where a website provides a set of structured HTTP requests that return JSON or XML files. Some of the most popular APIs for this are as follows[14]:

- Scraper Box API - This is a simple API that allows you to scrape data from online resources without experiencing blockades and returns the scraped data in HTML format.
- Scraper Crawler Extract API - This allows you to specify the URL of a web page and retrieve its contents fast and returns data as a JSON array.
- ScrapingBee API - This allows you to handle the various web scraping challenges so that you can harvest data without any worries and returns HTML formatted responses.

Dewi (2019) makes use of the Facebook and Twitter API to scrape data from social media in a way that the user is able to search information, combine and present it in a better way according to user preferences.[15]

2.2.2 Screen Scraping

In this method, you extract data from source code of a website, with an HTML parser or regular expression matching. A prior knowledge of HTML and CSS is helpful while making use of this method to extract data. There are various HTML parser libraries that are compatible with specific programming languages.

- The Java Programming language has parser libraries such as 'Largato and Jerry' and Jsoup.
- Python has parser libraries such as BeautifulSoup, HTML Parser of The Standard Library and Html5lib.
- JavaScript has parser libraries such as DOMParser, Cheerio and Parse5

2.3 Election Prediction

A lot of research has gone into election prediction. There are traditional methods that have been around for a while such as opinion polls and exit polls and also newer methods such as using social media data along machine learning algorithms to make predictions. Research has been done using various elections as a sample and different social media sources such as the 2011 dutch senate election results with twitter[11] and the recent Irish General Election as a case study for

investigating the potential to model political sentiment through mining of social media.

Fabio (2012) measures the predicting power of social media regressing one variable at a time using an ARIMA model which stands for 'Auto Regressive Integrated Moving Average', also known as Box–Jenkins model[9]. This model is fitted to time series data either to better understand the data or to predict future points in the series.[10]. Using data from social media sites such as Twitter and Facebook, he predicts the 2010 UK election results accurately with a 0.19338 percentage point error with an average Conservative prediction equal to 34.298788 and 34.265892, Labour prediction equal to 29.75584 and 28.46665 and Liberal Democrats prediction equal to 23.18733 and 26.28944. Overall, the ARIMA model seems to bypass unrepresentative sample flaws and leads to forecasts that surpass in accuracy those of the more traditional and expensive polls.

2.4 Machine Learning for Election Prediction

Machine learning is a tool that has been applied in various fields ranging from science and medical research for example using machine learning models to predict if someone has cancer, to financial research like producing stock market trends and of course politics is not left out when it comes to ways in which machine learning can be applied as new approaches for predicting various types of elections are being researched and implemented. Various types of datasets and machine learning algorithms are used to achieve these goals on different kinds of elections around the world. Having researched some of them I have discussed them below examining the datasets used, algorithms applied and results achieved.

León-Borges(2017) compares three different machine learning algorithms namely, grouping method K-means, expectation a convergence criteria, EM and methodology for classification LAMDA. To do this he makes use of the Weka and SALSA software to carry out this classification and he applies the algorithms on the local governor elections in 1998, 2004 and 2010 of the state of Quintana Roo, Mexico[16]. He then concluded that the LAMBDA algorithm used along with the SALSA software worked best as it gave the most accurate results.

Tsakalidis (2015) uses data extracted from Twitter and opinion polls to predict the 2014 European Union elections in three countries namely Germany, The Netherlands and Greece with the use of machine learning models such as linear regression, Gaussian process, and sequential minimal optimisation for regression, all implemented using Weka[17]. Gaussian process achieved the lowest Mean Absolute Error (1.31), followed by sequential minimal optimisation (1.35) and linear regression (1.42).

Ramteke (2016) tries to predict the winner of the 2016 American presidential elections between Donald Trump and Hilary Clinton using Twitter data and making use of two machine leaning algorithms named Multinomial Naive Bayes and Support Vector machines[18]. To predict the winner he uses machine learning to know if a tweet is positive or negative and then the candidate with the overall highest positive out of total tweet percentage is predicted as the winner.

Sharma (2016) utilised Dictionary Based, Naive Bayes and SVM algorithm to build their classifier and classified Twitter test data as positive, negative and neutral in order to predict general state elections of India in 2016. Tweets with the names of Indian political parties such as #BJP, #Congress, #NCP, #AAP were collected, and a total of 23,998 tweets relevant to these hashtags were collected. The results of the analysis for Naive Bayes and the SVM was the Bhartiya Janta Party as the winner and for the Dictionary Approach it was the Indian National Congress. SVM predicted a 78.4% chance that the BJP would win more elections in the general election due to the positive sentiment they received in tweets. As it turned out, BJP won 60 out of 126 constituencies in the 2016 general election, far more than any other political party as the next party (the Indian National Congress) only won 26 out of 126 constituencies[19].

2.5 NLP for Election Prediction

NLP which stands for Natural Language Processing deals with the interaction between computers and humans using the natural language. The ultimate objective of NLP is to read, decipher, understand, and make sense of the human languages in a manner that is valuable[20]. One of the uses of NLP is Sentiment Analysis, this is the interpretation and classification of emotions (positive, negative and neutral) within text data using text analysis techniques. Sentiment Analysis can be used to classify emotion of people in a country regarding a political party or candidate which can then be used to predict voting patterns in an election and therefore predict the winner of the election.

Oyebode (2019) uses Sentiment Analysis to predict election outcome in the 2019 Nigeria Presidential elections. To achieve this, he uses VADER a widely used lexicon-based technique which is used to extract sentiment from a piece of text on a range of -1 to +1. The VADER lexicon consists of 7,517 opinion words and text emoticons that express either negative or positive sentiments. This lexicon is then applied on text data gotten from we scrapping Nairaland an indigenous social media platform targeted at Nigerians[21].

Tsakalidis (2015) also uses sentiment analysis for its European Union countries election prediction by using a sentiment lexicon to train a model and use it to differentiate between positive and negative tweets from Twitter datasets. This resulted in *14,060/19,357* German, *13,838/18,993* Dutch, and *13,582/18,356* Greek positive/negative terms. To detect a tweet's sentiment, they used a naive sum-of-weights method on its keywords according to the respective lexicon, and assigned the majority class label (positive/negative) to it[17].

Sharma (2016) made use of SentiWordnet which contained synonyms and antonym of the respective words along with the score of that particular word. This was then used on the tweet data and keywords were compared and assigned polarity using a scoring scheme of +0.1 to +1.0 as positive, +0.1 to -1.0 as neutral and -0.1 to -1.0 as negative[19].

Ramteke (2016) makes use of manual labelling using Hash Tag Clustering and VADER the lexicon and rule-based sentiment analysis tool. First tweets are

labelled based on their hashtag as positive or negative and then VADER is used to label the rest of the data. This data is then used to train a model with the ability to label more tweet data and measure its accuracy using the manual labelled and VADER labeled data. Multinomial Naive Bayes and Support Vector machines were used as machine learning algorithms to make models using the nltk and Scikit learn packages and the Scikit learn SVM model came out with the highest accuracy at 0.99 percent.

Chapter 3

Project Management

In this chapter, I aim to discuss and describe how the project has been managed and evolved since its inception from its original form at the start of this year in the form of a research proposal. The project was originally planned to apply machine learning techniques and algorithms to predict election outcome in Nigeria using social media. As the literature was reviewed and data was collected, the project's topic and focus changed to UK election predictions due to the lack of data on previous Nigerian elections.

3.1 Timeline

The project was split into multiple parts with each of the parts having its own time duration and deadline in order to ensure progression was ongoing. While these deadlines existed, they were not completely strict therefore allowing room for flexibility and therefore changes could be made to these deadlines to better suit my progress at that point and the sort of results being achieved at that moment. The project duration lasted for about 36 weeks which began in January 2020 and lasted up until 24 September 2020. My timeline was built around the four submissions spread out throughout the project and these were the Research Proposal due on the 20th of February 2020, Project Presentation due on the 24th of April 2020, Interim Report due on the 24th of June 2020 and finally the Dissertation Report due on the 10th of September 2020. These submissions helped keep me on track and be able to access my progress at each stage. Feedback from my supervisor for these submissions also was helpful in assessing my progress and knowing what was being doing well, what needed to improved upon, what features were not necessary and therefore could be removed and new features which would improve my project.

Due to the flexibility of the timeline, the initial timeline had multiple differences compared to the final timeline as objectives changed slightly and deadlines shifted.

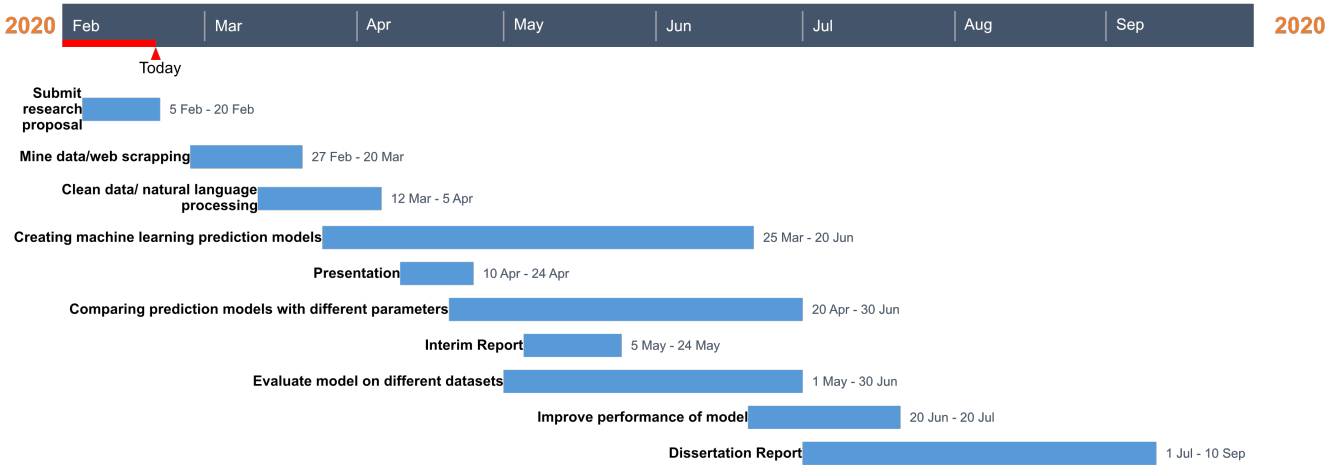


Figure 3.1 Initial Project Timeline as Submitted in Research Proposal

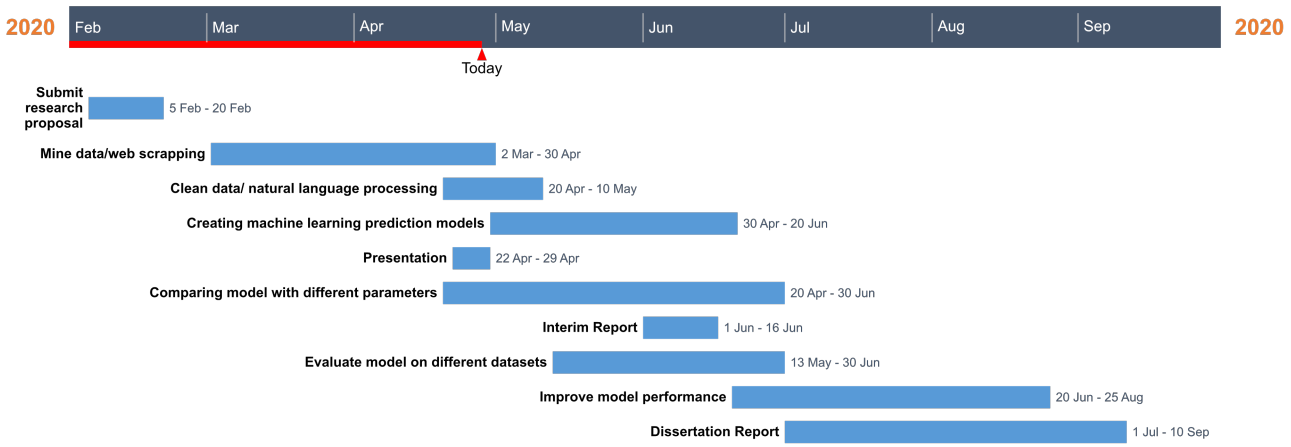


Figure 3.2 Project Timeline as at Presentation date

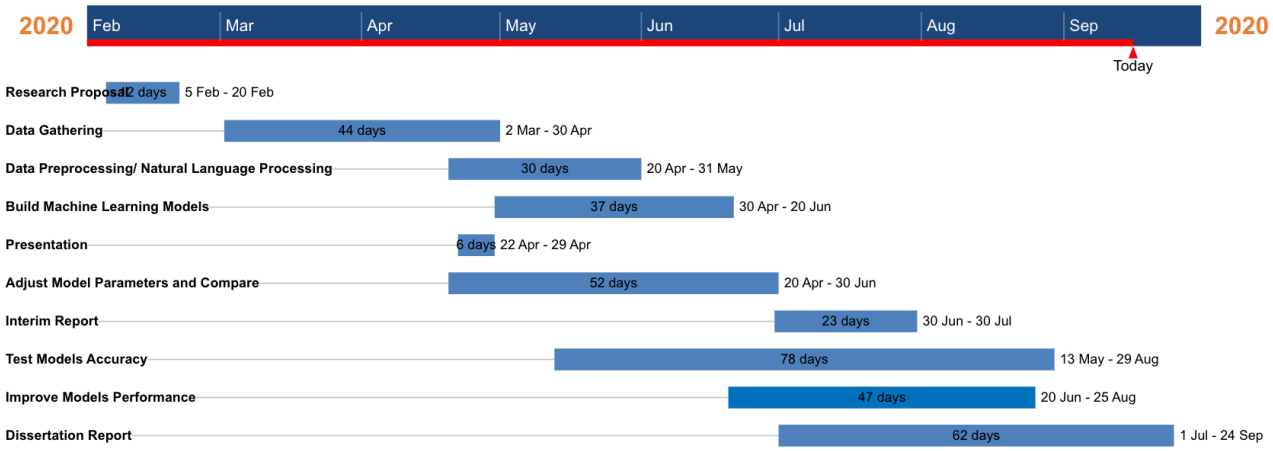


Figure 3.3 Project Timeline as at Interim Report Submission

Through the period working on my project the timeline plan changed relatively as shown in the difference between Figure 3.1 all the way to Figure 3.3, this was due to the pandemic affecting my workflow as I was unable to make use of the university computers which had faster processing powers compared to my personal computer. Also, due to the fact I had coursework and exams while working on the project it was necessary for me at these intervals to prioritise less on the project to concentrate on them. My health also played a role in adjusting my timeline as I unfortunately became ill twice and it became necessary to take a short break. While I started most stages on time, I did not necessarily end at eh expected time as different stages blended with each other along the way and I ended up working on multiple stages together for example, the preprocessing and building machine learning models stage. With all that I still ensured I met all submission deadlines on time roadblocks The first stage of my project after the initial research proposal which kicked off the project was to gather data. This took longer than I expected as finding available online text to provide sentiment was harder than I had initially though it would be. The solution to this was me scraping data of google news headlines to get the relevant data required for this project.

3.2 Risk Assessment

There were three potential risks notified before embarking on my my project. These were:

Underestimating Time to Complete Tasks

Severity: Medium

Probability: Medium

Reason: A task may be more complicated to complete the initially expected. If the difference is sever this can greatly affect the project progress but a small difference will not greatly affect the project.

Strategy: This has been taken care of as agile methodology allows flexibility in the project to readjust. Also enough time is given in the end during dissertation writing stage that any task that is still not done can be done within that period.

Illness Causing Delays

Severity: High

Probability: Low

Reason: As the project is done individually, illness has the chance of stopping progression for extended periods of time.

Strategy: Illness that is not so serious can be handled if there are breaks in progress as project is agile and therefore flexible enough to accommodate a break caused by an illness.

Delay due to unforeseen circumstances

Severity: Medium

Probability: High

Reason: Due to project been done over a long period of time it is unable to predict everything happening to detail and know of any random event causing delay. The pandemic false under this as it was unforeseen and caused delay in executing my project

Strategy: This has been taken care of as agile methodology allows flexibility in the project to readjust. Also enough time is given in the end during dissertation writing stage that any task that is still not done can be done within that period.

3.3 Coordination with Supervisor

Regular meetings were held with my supervisor throughout the duration of this project. These meetings were held mostly fortnightly with additional meetings being organised if there was an issue that needed to be discussed before the next official meeting date. Initial meetings were used to set a structure for the project knowing what will be feasible and the necessary tools I would need to successfully complete my project. In later these meetings, updates were given to my supervisor on my updates so far and what goals or objectives were in line to be achieved next, My supervisor also use this as an opportunity to give me feedback on what he thinks about my progress so far, letting me know if I am on the right path and makes recommendations on what I can include or remove from my project to make it better.

3.4 Technical Decisions

For this project, the kanban methodology has been adopted as well as parts of the agile methodology. Kanban works by making a visual system for managing work as it moves through a process. Kanban visualizes both the process (the workflow) and the actual work passing through that process. The goal of Kanban is to identify potential bottlenecks in your process and fix them so work can flow through it cost- effectively at an optimal speed or throughput[22]. I have also adopt some elements of agile which will allow my project to be less rigid and be able to evolve over time as it emphasises flexibility, continuous improvement, and high quality results[23], by always accepting changes to the requirements in order to be able to achieve the aim of the project.

Chapter 4

Methodology

In this chapter, the methodology which will be used to achieve building accurate models to predict UK elections by predicting the winner in each constituency in England and then overall in the UK using the party with the highest sum of constituencies won. There are several components of the methodology used to achieve a functional model and these will be looked into in detail.. The following sections are required to achieve the creation of an efficient model for this project:

- Political News Data
- Sentiment Analysed Data
- Constituency Data
- Election Result Data
- Train model
- Test model
- Evaluation

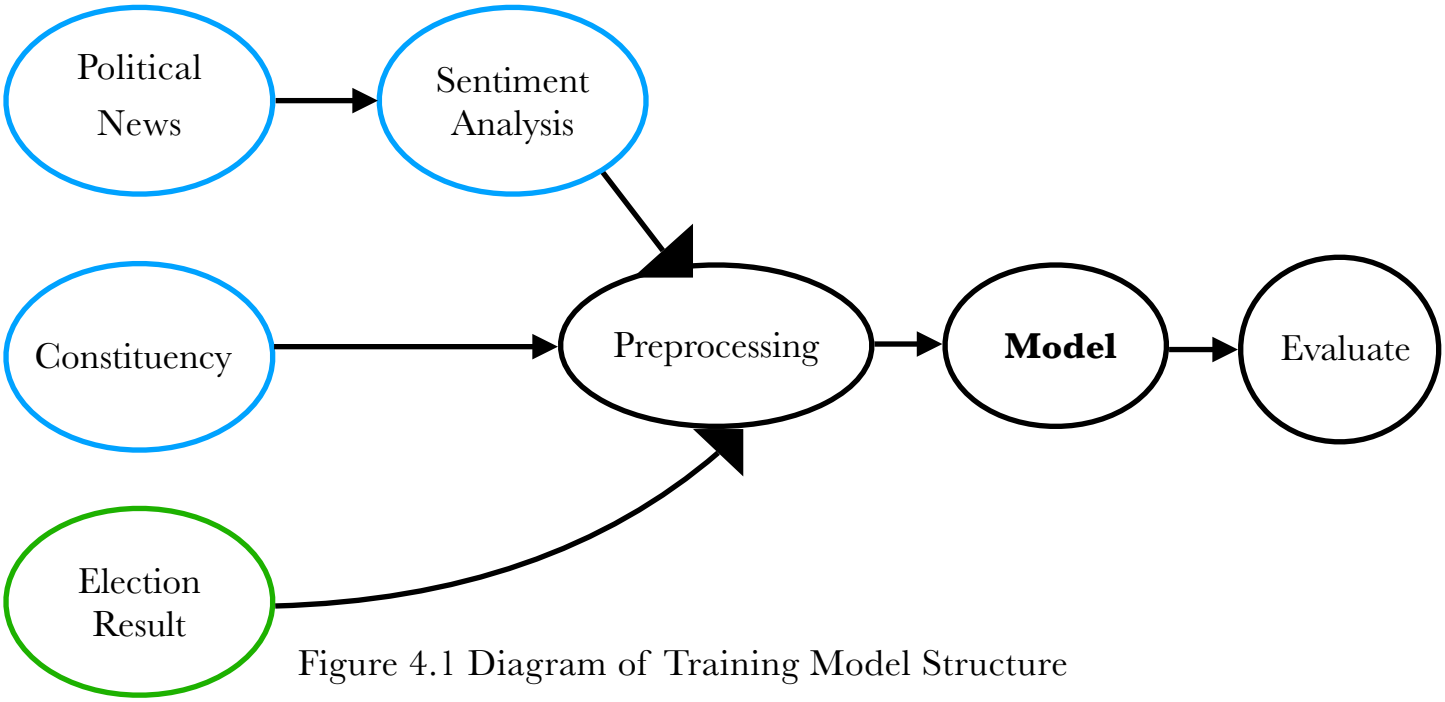


Figure 4.1 Diagram of Training Model Structure

Figure 4.1 showcases the structure of the model in its training phase. The training dataset consists of political news, sentiment analysis, constituency data and election results which are the inputs inserted to the model which its behaviour will depend on a machine learning algorithm. Constituency and political news data fall under the input vector while election results is the corresponding output vector which is the target(label) of the input.

4.1 Political News Dataset

For this dataset a compilation of news headlines and its summaries were collected. For each of the past elections being analysed (2019, 2017, 2015, 2010, 2005) top news data from google.com was collected in the time periods leading up to the elections, these time periods were divided into 6 parts. The first being 5 months before the elections and the subsequent ones each a month closer to the election with the last one being the month of the election. In each month, the top 10 news stories were scraped. This was done using the requests library which is a simple HTTP library for Python which allows you to send HTTP requests extremely easily. without the need to manually add query strings to your URLs; and the BeautifulSoup library which is a Python library for pulling data out of HTML and XML files. A sample of the script used is shown below:

```
def __init__(self, term,start,end):
    self.term = term
    self.start = start
    self.end = end
    self.sentiment = 0
    self.subjectivity = 0
    self.url = 'https://www.google.com/search?
tbs=cdr%3A1%2Ccd_min%{1}%2Ccd_max%{2}&q={0}
&source=lnms&tbm=nws'.format(self.term,self.start,self.end)


def run(self):
     headers = {
```

```
    "User-Agent":
        "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/44.0.2403.157 Safari/537.36"
        }
    response = requests.get(self.url, headers=headers)
    soup = BeautifulSoup(response.text, 'html.parser')
    headline_results1 = soup.find_all('div', class_='JheGif nDgy9d')
    headline_results = soup.find_all('div', class_='Y3v8qd')
    #open the spreadsheet we will write to
    with open('%s%s.csv' % (self.term,self.start[-4:]), 'a') as file:
    w = csv.writer(file)
    for text1,text in zip(headline_results1,headline_results):
        blob1 = TextBlob(text1.get_text())
        blob = TextBlob(text.get_text())
        w.writerow([blob1+'. '+blob,'%s' % (self.start)])
```

What this script does is take an object with a term, start date and end date and then inserting these keywords in the google search url where it will search for the top stories involving the term between the start date and end date. It then uses the requests library tool to go to the google url page and then uses beautifulSoup to save the html data of the request. When this is done each headline and summary on the google page are saved under the headline_results and headline_results1 as a list. A new csv file is then created and the data in the lists is then stored in the csv file by iterating through the list of news stories.

For each election, four different datasets were created with each focusing on a keyword in which top stories revolving around the keyword were scraped.

Below is a sample of one of the csv datasets which was scraped from google news data. The left column contains the news data while the right column contains the time period.

News Stories	Dates
Jeremy Corbyn increased Labour's vote share more than any of the party's leaders since 1945. Jeremy Corbyn increased Labour's vote share more than any of the party's leaders since 1945. Socialist achieves bigger swing than Tony Blair. Harriet Agerholm …	3A6%2F07%2F2017
Labour won social media election, digital strategists say. Party's use of Facebook and Twitter to motivate voters, rather than attack Conservatives, regarded as key to success. Robert Booth and Alex Hern. Fri 9 Jun 2017 …	3A6%2F07%2F2017
Students inspired by Corbyn played big role in Labour surge. The surprise surge of Jeremy Corbyn's Labour party in the general election has been explained by a significant rise in youth turnout, with experts saying 18 to …	3A6%2F07%2F2017
Celebrate the transformation of the Labour Party, not the …. Lea Ypi argues that the Labour Party is now a reinvigorated force. It is the largest social democratic party in Europe; its members are now at the centre of its …	3A6%2F07%2F2017
Gordon Brown tried to broker DUP deal after 2010 hung parliament. They've effectively just replaced the Ulster Unionists (UUP) - but the two parties are anything but the same. While the UUP is a relatively harmless centre-right party …	3A6%2F07%2F2017
Thanks to Jeremy Corbyn, populism is no longer a dirty word. More than just revive a moribund Labour Party and social democratic agenda – he is fundamentally transforming who are the "British people". Corbyn and his …	3A6%2F07%2F2017
Jeremy Corbyn, life and soul of the Labour Party. But although the personal devotion that Mr Corbyn inspires among tented grime fans has given the party a big boost, Labour remains divided on how it might …	3A6%2F07%2F2017
Why the Rise of Corbyn's Labour Party Should Worry the West. In the days since British Prime Minister Theresa May's disastrous snap election, the Labour Party and its leader, Jeremy Corbyn, have been taking in the sheer …	3A6%2F07%2F2017
Corbyn critics win election to Labour parliamentary committee. Ian Lavery, the Labour party chairman, attempted to calm concerns among MPs on Friday, telling the Mirror: "I don't see deselection as the way forward.".	3A6%2F07%2F2017
The roar of the Glastonbury crowd is not enough – Labour …. If there were an extended metaphor for the state of the Labour Party since the snap election, it would be the Glastonbury Festival. I was there this year and heard …	3A6%2F07%2F2017

Figure 4.2. Subset of Labour Party 2017 News Data

4.1.1 Conservative Leader News

This dataset consisted of top news stories of the conservative party candidate running for the position of Prime Minister of the UK. As the person running for this position may be different in each election, the candidates name is used as the keyword for retrieving the top news stories concerning them within the 6 month period pre elections. For each of the elections used in this projects these are the party leaders for the Conservative Party:

- 2005 - Michael Howard
- 2010 - David Cameron
- 2015 - David Cameron

- 2017 - Theresa May
- 2019 - Boris Johnson

4.1.2 Conservative Party News

This dataset consisted of top news stories of the conservative party as a whole. 'Conservative Party' is used as the google search keyword for retrieving the top news stories concerning the party within the 6 month period pre elections.

4.1.3 Labour Leader News

This dataset consisted of top news stories of the labour party candidate running for the position of Prime Minister of the UK. As the person running for this position may be different in each election, the candidates name is used as the keyword for retrieving the top news stories concerning them within the 6 month period pre elections. For each of the elections used in this projects these are the party leaders for the Labour Party:

- 2005 - Tony Blair
- 2010 - Gordon Brown
- 2015 - Ed Miliband
- 2017 - Jeremy Corbyn
- 2019 - Jeremy Corbyn

4.1.4 Labour Party News

This dataset consisted of top news stories of the labour party as a whole. 'Labour Party' is used as the google search keyword for retrieving the top news stories concerning the party within the 6 month period pre elections.

4.2 Sentiment Analysis

Sentiment analysis measures the tone and intent of online comments. It involves natural language processing technologies to help comprehend entities and relationships to reveal positive, negative, neutral or uncertain attributes. An alternative approach to sentiment analysis includes more granular sentiment analysis which gives more precision in the level of polarity analysis which aims to identify emotions in expressions (e.g. happiness, sadness, frustration, surprise).

To perform sentiment analysis, the text has to be processed by performing the following[24]:

- Tokenization- Where text is split into sentences and individual words.
- Stemming- A method for classifying related words.
- Lemmatization- Words are reduced to their root form.
- Normalization- Transforming a list of words to a more uniform order.
- Vectorization- NLP methodology to convert words into numbers.

For my project I made use of the VADER(Valence Aware Dictionary for Sentiment Reasoning) which is a model used for text sentiment analysis that is sensitive to both polarity (positive/negative) and intensity (strength) of emotion. It is available in the NLTK package and can be applied directly to unlabelled text data. VADER sentimental analysis relies on a dictionary that maps lexical features to emotion intensities known as sentiment scores. The sentiment score of a text can be obtained by summing up the intensity of each word in the text[25].

VADER was used to draw out the polarity in the political news datasets. The polarity lied between -1 to +1 with -1 being completely negative news and +1 for completely positive news. Each news story polarity was derived using the SentimentIntensityAnalyzer function in the nltk VADER library afterwards after deriving the polarity score of all the news stories and adding them to a list, the mean of all the polarity scores is calculated and this value is used to represent the total polarity score for the candidate or party for that particular election year.

```
sia = SentimentIntensityAnalyzer()
results = []

for text1,text in zip(headline_results1,headline_results):
    blob1 = TextBlob(text1.get_text())
    blob = TextBlob(text.get_text())
    blob_all = blob1+'. '+blob
    pol_score = sia.polarity_scores(str(blob_all))
    pol_score['headline'] = str(blob_all)
    results.append(pol_score)

    dfr = pd.DataFrame.from_records(results)
    self.sentiment = dfr["compound"].mean()
```

This script uses the SentimentIntensityAnalyzer function to calculate the polarity of the news data. It iterates through the list of news stories, calculates the polarity score of each one of them and then adds it to a new list. The mean of all the values are then calculated and saved as the sentiment polarity score of the object.

4.3 Constituency Socioeconomic Conditions Dataset

The data gotten for this was gotten from the house of commons website which has made available variable collections of datasets for public usage. It uses 8 different socioeconomic categories to classify the 533 Westminster Parliamentary constituencies in England.

Below is a subset of all the data collated from the website

Constituency	House median	Wage Median	Proportion British	Prop owning own house	Cons. A*-C %	0-9	10-19
Kensington	1450000	840	0.476823452	0.337770936	0.776	0.110977166	0.095173567
Cities of London and Westminster	1200000	780	0.480445455	0.315429136	0.741	0.091860223	0.087179448
Chelsea and Fulham	980000	830	0.566316925	0.403049714	0.764	0.119782414	0.089360611
Hampstead and Kilburn	755000	850	0.540614975	0.381704119	0.611	0.128909332	0.09497056
Holborn and St Pancras	751750	680	0.597108565	0.274977336	0.693	0.10442188	0.114619942
Westminster North	750000	830	0.464396126	0.304122651	0.735	0.129777765	0.104050337
Hammersmith	730000	800	0.553520078	0.307479118	0.641	0.118639502	0.093273278
Battersea	715000	830	0.654836987	0.419969195	0.801	0.117859055	0.071551819
Islington South and Finsbury	696250	870	0.637498052	0.265584498	0.717	0.097156457	0.089193218

Figure 4.3 Subset of Constituency Socioeconomic Conditions Dataset part 1

20-29	30-39%	40-49%	50-59%	60-69	70-79%	80%
0.140270139	0.175599454	0.151542083	0.129816534	0.091953012	0.068256413	0.036411633
0.183004473	0.196330826	0.14117004	0.120822523	0.087749147	0.057077749	0.034805571
0.157063316	0.179224894	0.149682229	0.12295097	0.083746955	0.062079892	0.03610872
0.156663317	0.20150809	0.147112979	0.106780335	0.074352351	0.054975146	0.03472789
0.224696255	0.175601215	0.121259665	0.106411286	0.074398785	0.04974144	0.028849533
0.146134176	0.205051552	0.14769974	0.110864266	0.073991516	0.049330162	0.033100487
0.185721502	0.197945782	0.145470618	0.114834147	0.071299882	0.046152551	0.026662738
0.210567026	0.254468801	0.134739777	0.094297444	0.05760355	0.037067888	0.02184464
0.273014749	0.209811172	0.115067568	0.096407072	0.058937851	0.037625687	0.022786228

Figure 4.4 Subset of Constituency Socioeconomic Conditions Dataset part 2

4.3.1 Proportion Owning House

This was a count of the percentage of the population who had ownership of their own residence. This has been included as a social and economic indicator. This data ranges from 97% in Sefton Central to 20% in Hackney South and Shoreditch and was assembled in the 2011 Census

4.3.2 Age Distribution

This category splits the demographic of constituency residents into 9 age groups, (0-9, 10-19 and so on up until 80+) and has been included as a social indicator. This data was collected in the 2011 census.

4.3.3 Proportion Born in the UK

The percentage of residents in a constituency born in the UK has been included as a social indicator of a region. This data was collected in the 2011 Census and ranges from 41% in Brent North to 98% in Houghton and Sunderland South.

4.3.4 Percentage of Students Achieving 5A*-C Grades at GCSE

The percentage of students achieving good school grades has been included as a social indicator. This data was collected in 2018 and ranges from 38% in Knowsley to 85% in Altrincham and Sale West.

4.3.5 Median House Price

Median house price is a measure of the cost of living in each constituency and has been included as an economic indicator. This data was recorded in March 2019 and ranges between £72,500 in Liverpool, Walton to £1,450,000 in Kensington.

4.3.6 Brexit Vote

The Brexit category is an estimation of the percentage of residents in each constituency who voted to leave the European Union. This data is extrapolated from the 2016 Brexit referendum results and ranges from 20% in Hackney North and Stoke Newington to 76% in Boston and Skegness.

4.3.7 Median Wage

The weekly median salary of the population living in the constituency has been included as an economic indicator. This data was recorded in April 2019 and ranges from £890 in Wimbledon to £420 in Leicester East.

4.4 Election Result Dataset

This dat was gotten from the house of commons website which has made available variable collections of datasets for public usage and wikipedia when it was unavailable on the house of commons website. Figure 4.5 below shows a subset of the election result data gotten from the website.

Constituency	Party
Aberavon	Lab
Aberconwy	Con
Aberdeen North	SNP
Aberdeen South	SNP
Airdrie and Shotts	SNP
Aldershot	Con
Aldridge-Brownhills	Con
Altrincham and Sale West	Con
Alyn and Deeside	Lab
Amber Valley	Con

Figure 4.5 Subset of 2019 Election Results

4.5 Preprocessing

Data preprocessing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we preprocess our data before feeding it into our model[26]. Data preprocessing in Machine Learning refers to the technique of preparing, cleaning and organising the raw data to make it suitable for a building and training Machine Learning models. In simple words, data preprocessing in Machine Learning is a data mining technique that transforms raw data into an understandable and readable format.

4.5.1 Data Quality Assessment

Because my data was taken from multiple sources which sometimes were in different formats, a lot of time was consumed in dealing with data quality issues when working on this project. It was simply unfeasible to expect that the data will be perfect.

To fix the issue with missing values in the datasets, the affected rows in the tables are eliminated. This method works when there are not too many missing values in the data such that eliminating them will not lead to a huge change in results if the data was available. If a feature has mostly missing values, then that feature itself

can also be eliminated. This issue arose with the 2005 election data because at that time there was a difference in the constituency structure compared to its present day arrangement. Because of that Constituency results that were expected in the dataset were missing as they were under different names or simply non-existing.

4.5.2 Inconsistent values

It is known that data can contain inconsistent values. This issue is likely to occur at some point. It may be due to human error or maybe the information was misread while being scanned from a handwritten form or just different datasets having different formats.

For my election result data the party in different elections were spelt differently. For example one dataset would refereed to the Liberal democrats party as 'Libs' while another referred to them as 'LD' while it was obvious for a human to know they referred to the same party, the computer would not be aware and see them as separate entities. Therefore all entries of the Liberal Democrats party was changed to 'LD'.

4.5.3 Feature Encoding

Feature encoding is basically performing transformations on the data such that it can be easily accepted as input for machine learning algorithms while still retaining its original meaning. For nominal data a one to one mapping can be done. In the election result data, the winning party is nominal data and it has been preprocessed using the label encoder to become a numerical value. The political parties and their encoded values are:

- 'Con' - 0
- 'Green' -1
- 'KHHC' - 2
- 'LD' - 3
- 'Lab' - 4

- ‘Resp' - 5
- ‘Spkr’ - 6
- ‘UKIP' - 7

4.6 Model

A machine learning model is a file that has been trained to recognise certain types of patterns and the model can either be supervised or unsupervised. If the model is supervised, the model is trained over a set of data, providing it an algorithm that it can use to reason over and learn from those data. As seen in figure 4.6 it involves learning a function that maps an input to an output based on example input-output pairs. Supervised learning models are sub-categorised as either a regression or classification model[26].

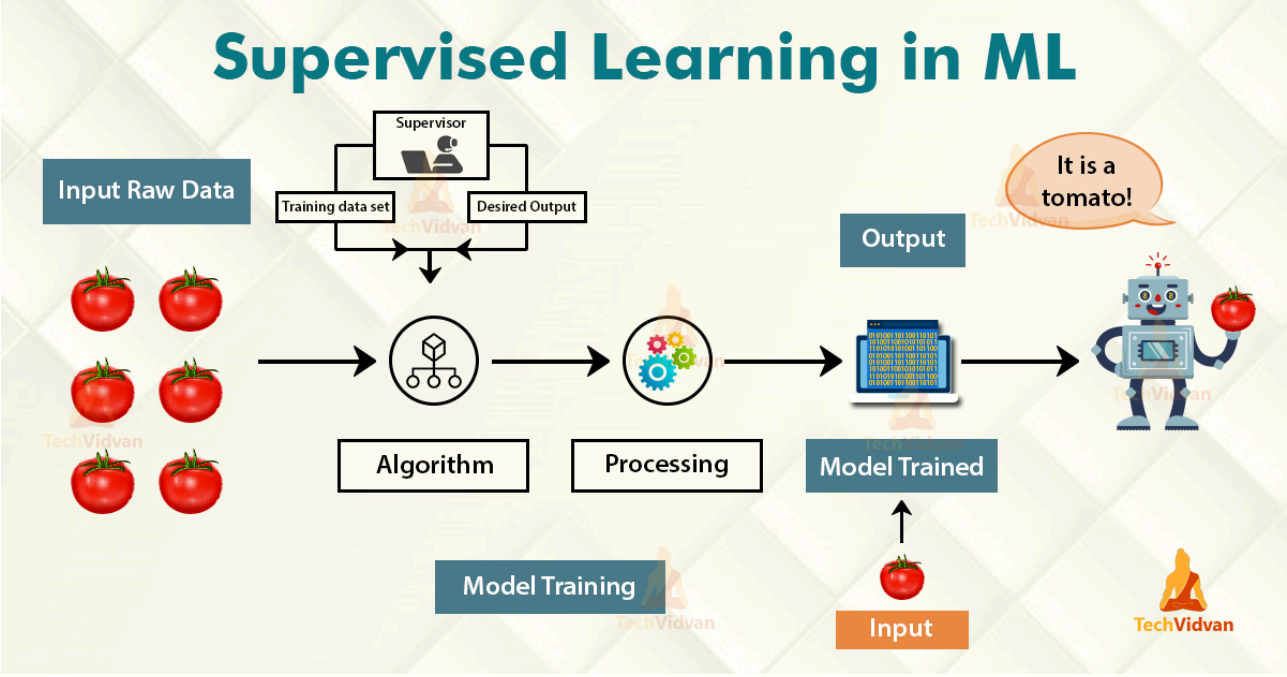


Figure 4.6 Supervised Learning Model Diagram

For my project the classification model is being used which is a subcategory of the supervised learning model. A classification model tries to draw some conclusion from the input values given for training. It will predict the class labels/categories for the new data as seen in Figure 4.7. Of the various types of classification models, here are the four types that were utilised in my project.

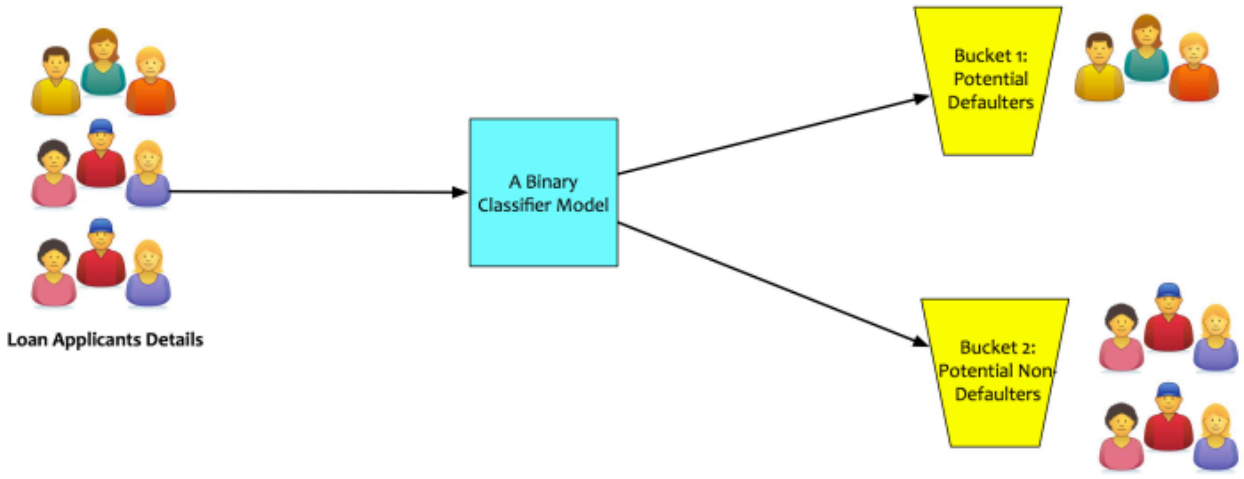


Figure 4.7 Classification Model Diagram

4.6.1 Decision Tree Classification

Decision Tree Classifier is a Supervised Machine Learning where the data is continuously split according to a certain parameter.

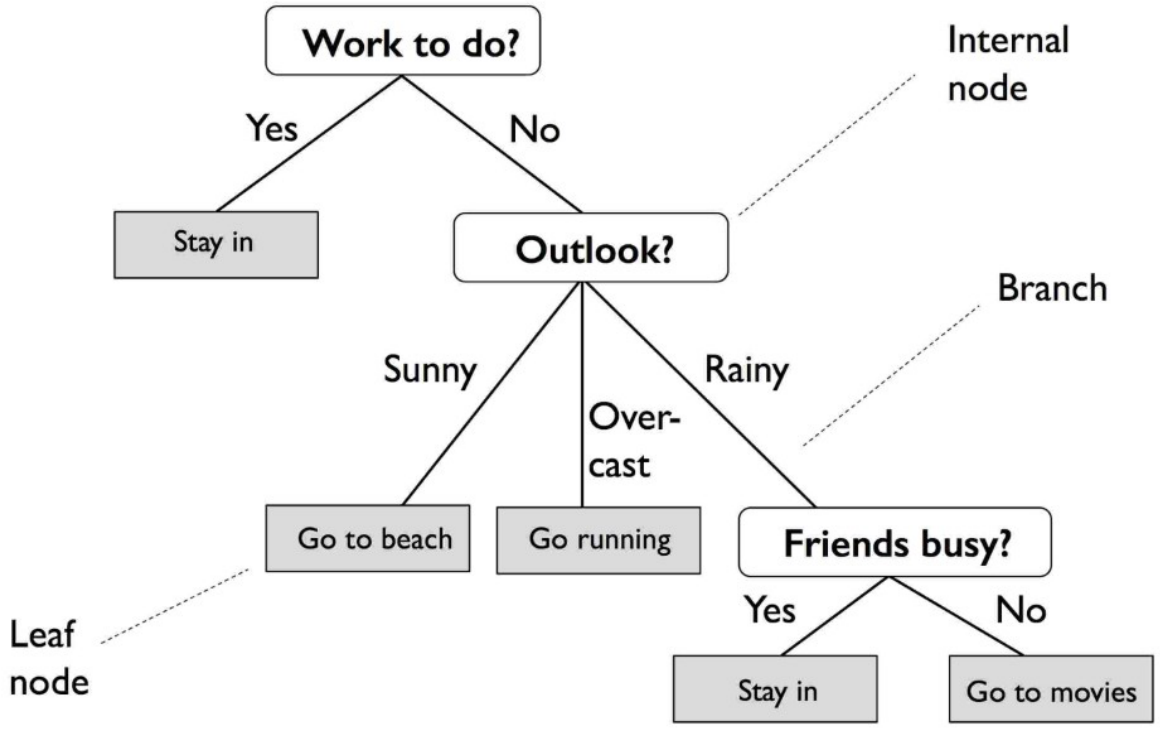


Figure 4.8 Decision Tree Classifier

As seen in Figure 4.8, decision trees are made up of:

- Nodes - Test for the value of a certain attribute.
- Branches - Correspond to the outcome of a test and connect to the next node or leaf.
- Leaves - Terminal nodes that predict the outcome.

For classification using decision trees, the decision variable is categorical. The tree is constructed through a procedure known as binary recursive partitioning. This process involves splitting the data into partitions iteratively, and then further dividing it up on each of the branches.

Advantages:

- Keeps out insignificant features.
- Very fast at classifying unknown records.
- Easy to interpret for small-sized trees.

Disadvantages:

- Easy to overfit.
- Decision tree models are often biased toward splits on features having a large number of levels.
- Small changes in the training data can result in large changes to decision logic.

4.6.2 Random Forest Classification

Random forest, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.

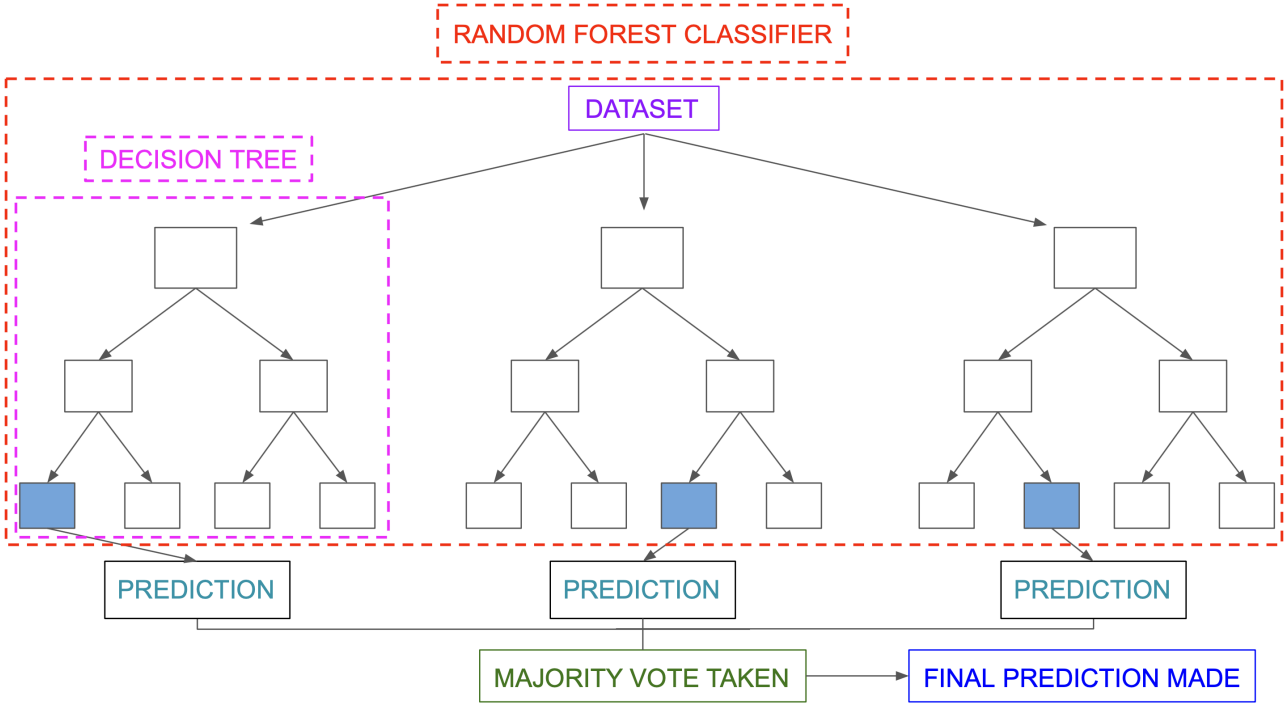


Figure 4.9 Random Forest Classifier

The key for the random forest to work is having low correlation between trees as this will ensure ensemble prediction are greater than individual predictions. This is so because the trees end up protecting themselves from individual inaccuracy so if certain trees end up being wrong, others will be right and as a group they will make the right decision.

Advantages:

- Great with high dimensional data.
- It has methods for balancing error in class population unbalanced data sets.
- It has low bias and average variance.

Disadvantages:

- Random forest models are not all that interpretable.
- There is a tendency for it to overfit, so tuning hyperparameters is advised.

4.6.3 AdaBoost Classification

Boosting is an ensemble technique that attempts to create a strong classifier from a number of weak classifiers. Boosting is an ensemble technique that attempts to create a strong classifier from a number of weak classifiers. AdaBoost works in the following steps:

- Initially, AdaBoost selects a training subset randomly.
- It iteratively trains the AdaBoost machine learning model by selecting the training set based on the accurate prediction of the last training.
- It assigns the higher weight to wrong classified observations so that in the next iteration these observations will get the high probability for classification.
- Also, It assigns the weight to the trained classifier in each iteration according to the accuracy of the classifier. The more accurate classifier will get high weight.
- This process iterate until the complete training data fits without any error or until reached to the specified maximum number of estimators.
- To classify, perform a "vote" across all of the learning algorithms you built.

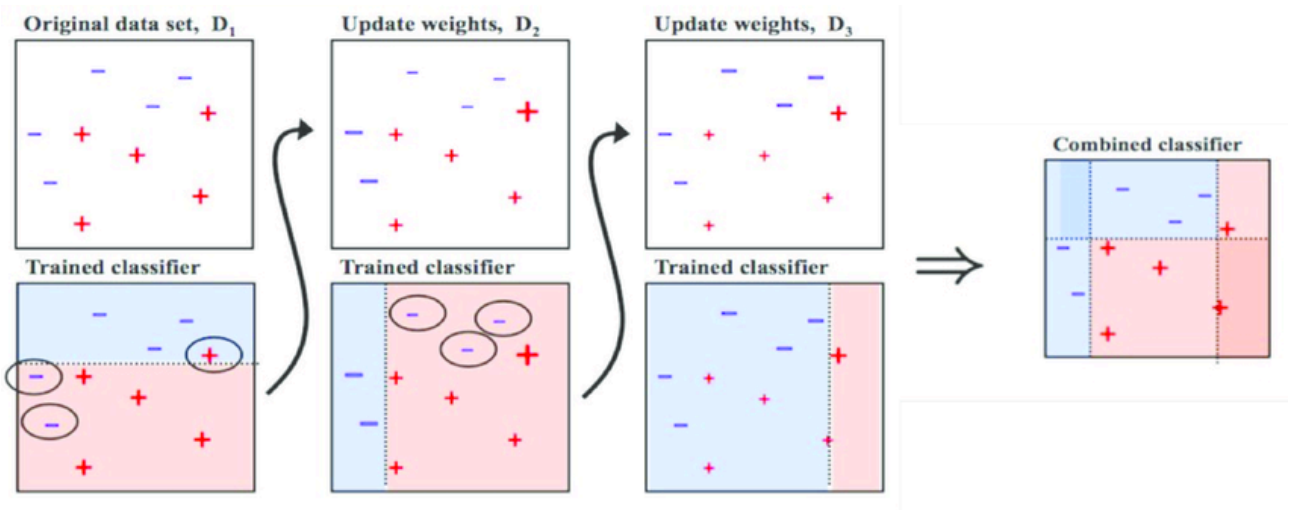


Figure 4.10 AdaBoost Classifier

Advantages:

- It iteratively corrects the mistakes of the weak classifier and improves accuracy by combining weak learners.

- AdaBoost is not prone to overfitting.
- Many base classifiers can be used with AdaBoost.

Disadvantages:

- AdaBoost is sensitive to noise data.
- It is highly affected by outliers because it tries to fit each point perfectly.

4.6.4 Support Vector Machine Classification

In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate[29]. Then, we perform classification by finding a hyperplane or a boundary between the two classes of data that maximises the margin between the two classes. There are many planes that can separate the two classes, but only one plane can maximise the margin or distance between the classes.

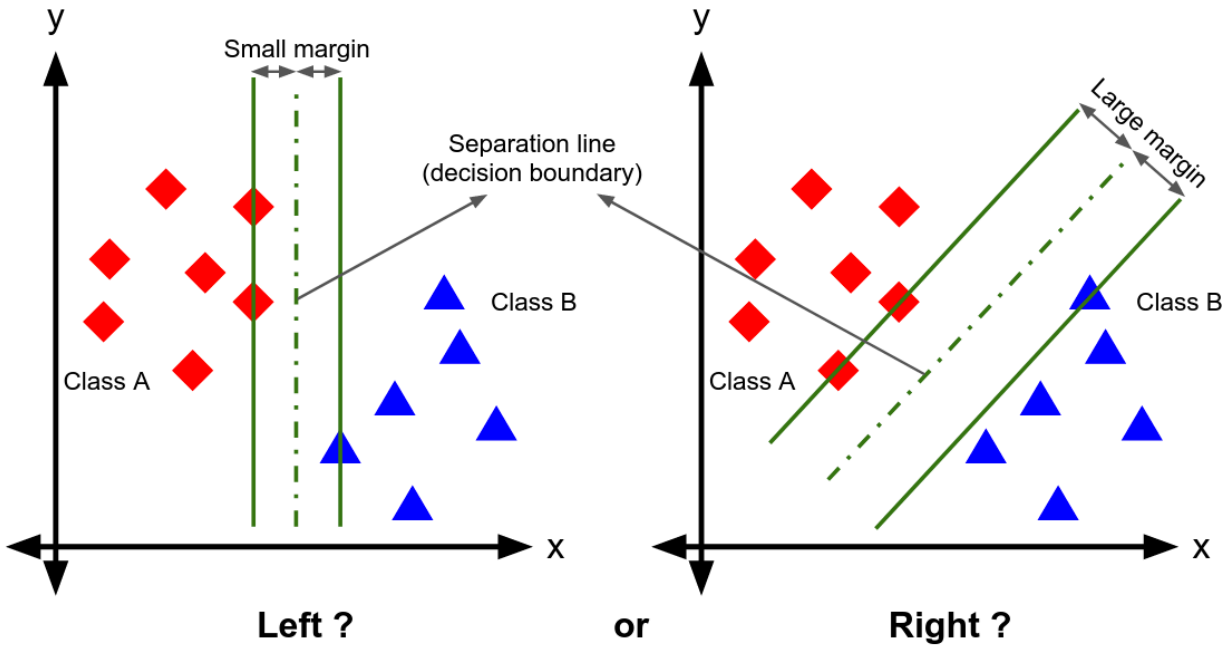


Figure 4.11 SVM Classifier

As seen in Figure 4.11 SVM classifier chooses the right hyperplane as its aim is to maximise the margin.

Advantages:

- It is effective in high dimensional spaces.
- It works really well with a clear margin of separation
- It is memory efficient as it uses subset of training points in the decision function

Disadvantages:

- Large datasets require long training times therefore reducing performance
- It doesn't perform very well, when the data set has a lot of noise

4.7 Training

The process of training involves providing the machine learning model with data to learn from. This will be done by dividing the data into two parts which are the training set and test set. The training set will be used to fit and tune the model while the test set which will be unseen by the model till after the training is done will be used to evaluate the model.

4.7.1 K-Fold Cross Validation

K-Fold Cross Validation consists in splitting the data into K partitions of equal size. For each partition i, the model is trained with the remaining K-1 partitions and it is evaluated on partition i. The final score is the average of the K scored obtained. Figure 4.12 shows a diagram of how this works and how the data is first split into folds and in each split one of the folds becomes the test data while the remaining remain training data.

For this project I will be using 90% of my dataset for training and 10% for testing while I would use the K-Fold Cross Validation with 10 folds.

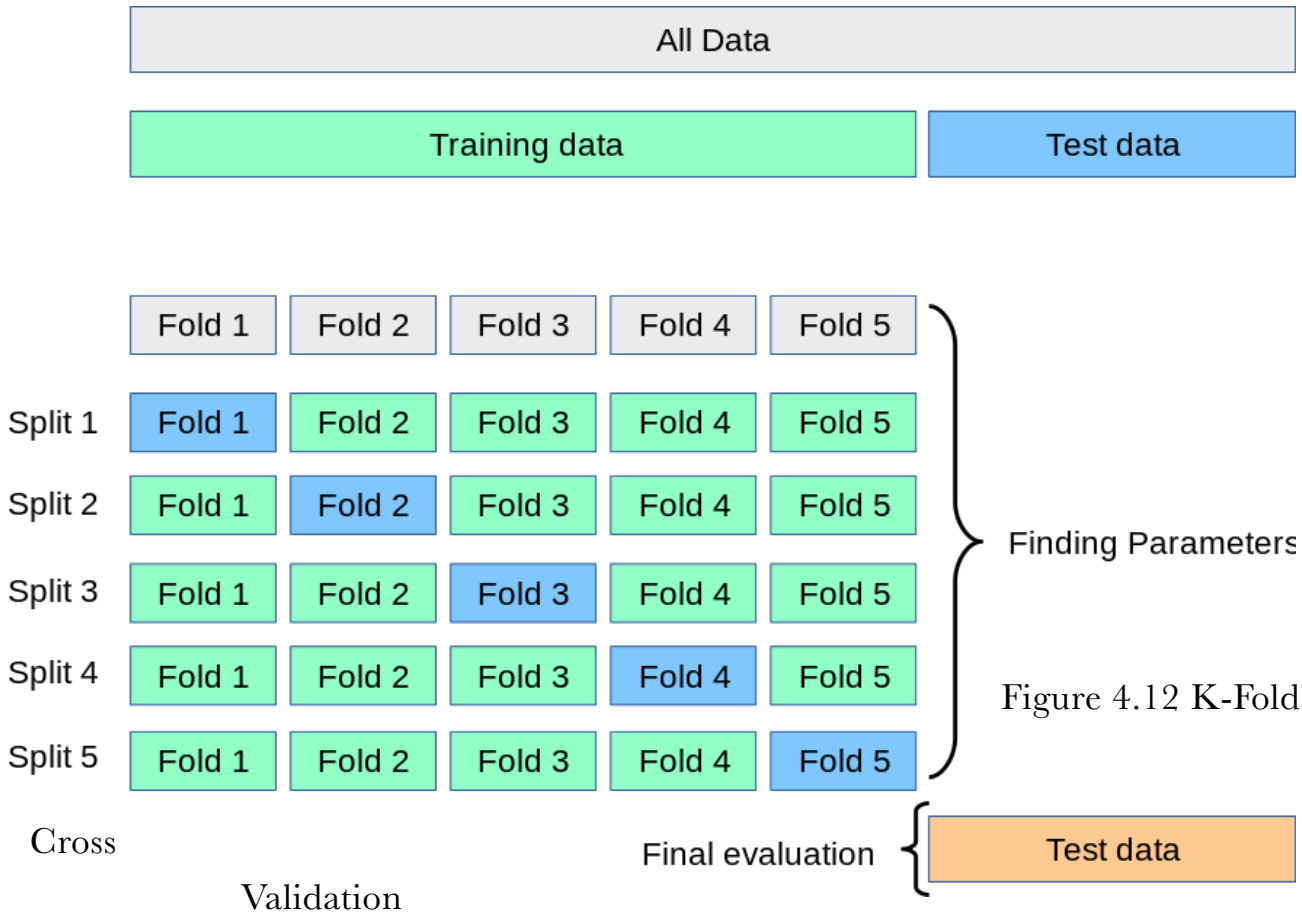


Figure 4.12 K-Fold

4.8 Evaluation

Evaluating a classification model is an integral part of the model development process as it helps to know which model works well with the data and also to know which model will work best in the future. Evaluation can be done using various metrics and for this project I make use of:

4.8.1 Classification Accuracy

Classification accuracy shows how many of the prediction are correct out of the total predictions made.

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

This represents how good a model is in some cases but in cases where the dataset has an uneven distribution it is not a good representative of the model performance.

4.8.2 Confusion Matrix

A confusion matrix is not necessarily a metric to measure model performance but it gives insight in order to produce other classification metrics such as recall and precision. Confusion matrix goes deeper than classification accuracy by showing the correct and incorrect predictions on each class.

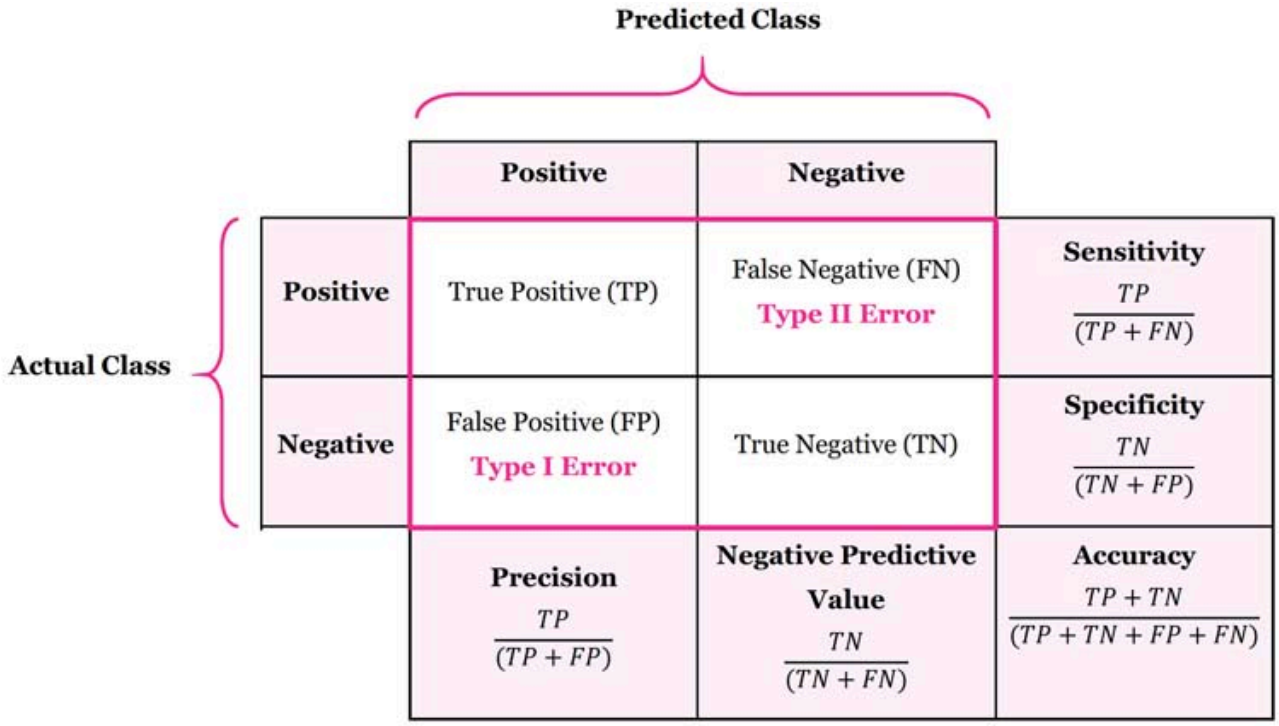


Figure 4.13 Confusion Matrix

The confusion matrix is made up of:

- True Positive: Positive class is predicted as positive. (Correct)
- False Positive: Negative class is predicted as positive. (Wrong)
- True Negative: Negative class is predicted as negative. (Correct)
- False Negative: Positive class is predicted as negative. (Wrong)

The aim is for all predictions to be correct and therefore fall within the true positive or true negative class. False positive is also knowns as type I error while false negative is also known as type II error.

4.8.3 Precision

Precision takes it a step further in accuracy allowing more specific understanding in model performance. It measures how good the model is when the predictions are positive. Precision uses values from the confusion matrix to calculate this. The formula to get its value is:

$$Precision = \frac{TP}{TP + FP}$$

4.8.4 Recall

Recall just like precision takes it one step further in accuracy to understand the model performance. For recall, it measures how good our model is at correctly predicting positive classes. It also uses values from the confusion matrix to calculate the value and the formula is:

$$Recall = \frac{TP}{TP + FN}$$

4.8.5 F1 Score

F1 Score combines both precision and recall to a single number. It is the weighted average of precision and recall. It is a more functional measure for problems with an uneven class distribution than accuracy and this is because both false positive and false negatives are taken into consideration. The F1 score lies between 1 and 0 with one being the best value and 0 being the worst value. The F1 Score formula is:

$$F1\ Score = 2\frac{Precision\ *\ Recall}{Precision\ +\ Recall}$$

4.8.6 AUC ROC Curve

ROC curve summarises the performance of the model at different threshold values by combining confusion matrices at all threshold values. X axis of ROC

curve is the true positive rate (sensitivity) and y axis of the ROC curve is the false positive rate (1- specificity).

True Positive Rate(Sensitivity) Formula:

$$TPR(Sensitivity) = \frac{TP}{TP + FN}$$

False Positive Rate(1-Specificity) Formula:

$$TPR(Sensitivity) = \frac{TP}{TP + FN}$$

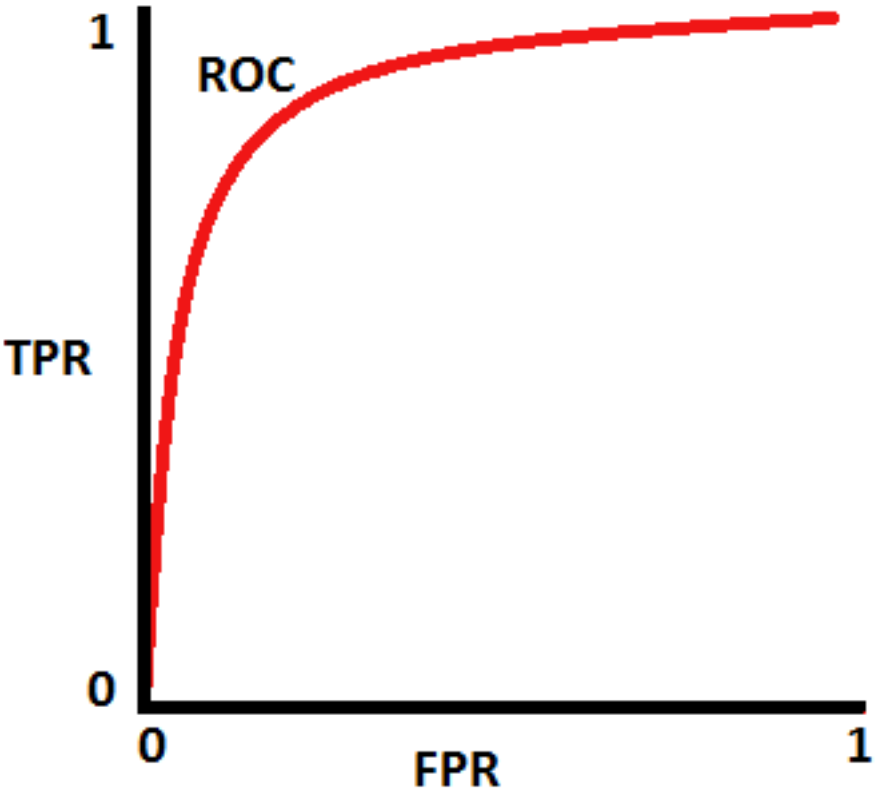


Figure 4.14 ROC Curve

If the threshold is set to 0, the model predicts all samples as positive. In this case, TPR (sensitivity) is 1. However, FPR(1-specificity) is also 1 because there is no negative prediction. If the threshold is set to 1, both TPR and FPR become 0.

Hence, it is not a good choice to set the threshold to 0 or 1. We aim to increase the true positive rate (TPR) while keeping false positive rate (FPR) low. As we can see on the ROC curve, as TPR increases, FPR also increases. So it comes down to decide how many false positives we can tolerate[30].

AUC is the area under the ROC curve between (0,0) and (1,1). AUC fundamentally combines the performance of the model at all threshold values. The best possible value of AUC is 1 which signifies a perfect classifier. The classifier gets better as the AUC gets closer to 1.

In the figure below, classifier A is better than classifier B.

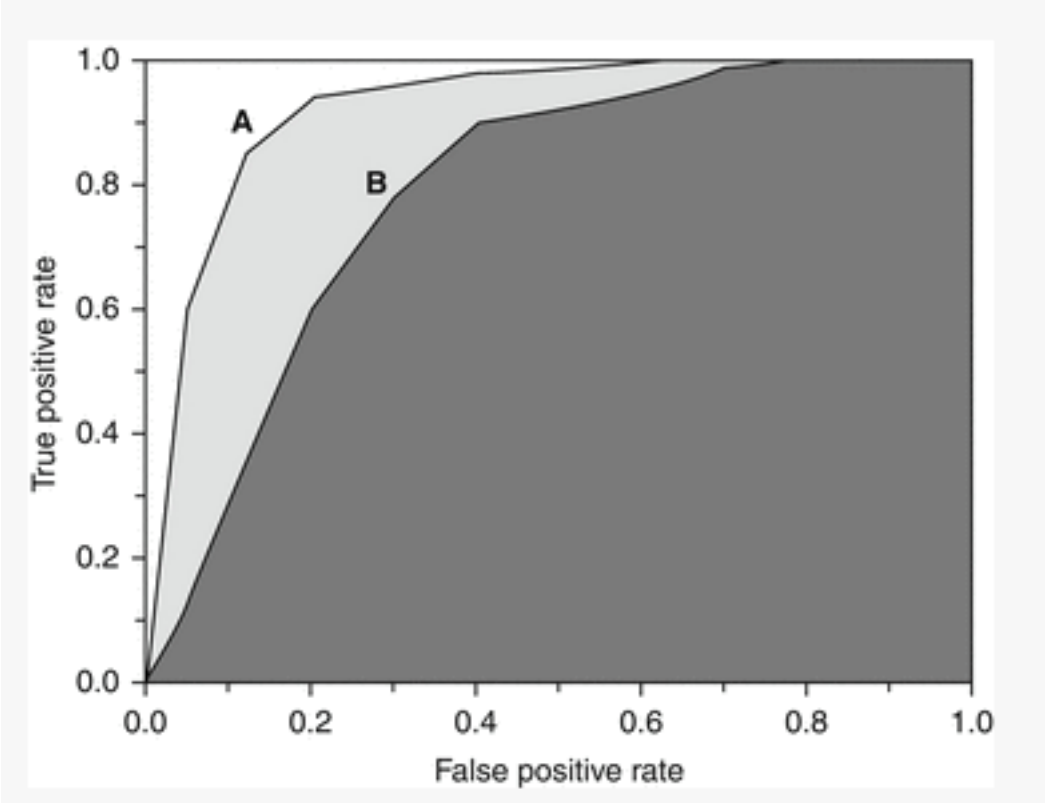


Figure 4.15 Comparing AUC in ROC Curve

Chapter 5

Implementation and Results

This chapter gives an in depth descriptions of what was done to implement the election prediction model. This was implemented using Python 3 as Python has a robust collection of machine learning packages which are necessary for me to make use of machine learning. As I have used four different models, I will discuss how they were implemented and evaluate the results each model brings.

5.1 Modules and Packages

Various python packages were used to implement features and achieve final model results. They are:

- Tweepy[31] - Tweepy is a python library for accessing the Twitter API. The API class provides access to the entire twitter RESTful API methods. Each method can accept various parameters and return responses.
- NLTK[32] - NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries.
- BeautifulSoup[33] - Beautiful Soup is a library that makes it easy to scrape information from web pages. It sits atop an HTML or XML parser, providing Pythonic idioms for iterating, searching, and modifying the parse tree.
- Scikit-Learn[34] - Scikit-learn is an open source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data preprocessing, model selection and evaluation, and many other utilities.
- Matplotlib[35] - Matplotlib is a comprehensive library for creating static, animated, and interactive visualisations in Python.

- Pandas[36] - Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.
- Numpy[37] - NumPy is a python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.
- Seaborn[38] - Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.
- Geopandas[39] - GeoPandas is an open source project to make working with geospatial data in python easier. GeoPandas extends the datatypes used by pandas to allow spatial operations on geometric types. Geometric operations are performed by shapely. Geopandas further depends on fiona for file access and descartes and matplotlib for plotting.

5.2 Preprocessing

To preprocess my data in order to make it more simplified for the model, I made use of :

- MinMax Scaler - This transforms features by scaling all of the features to a certain range. Each feature us scaled and translated individually in a certain way it lies within the range of the set. Formula for this to be implemented is:

```
X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
X_scaled = X_std * (max - min) + min
```

To utilise this in my project I had the following code:

```
1 min_max_scaler = preprocessing.MinMaxScaler()
2 x_scaled = min_max_scaler.fit_transform(unscaled.values)
3 scaled_values = pd.DataFrame(x_scaled)
```

- Label Encoder - This is used to convert categorical data, or text data, into numbers, which our predictive models can better understand. In my dataset I used it to transform the y data which is the winning political party from text to a number.

```
1 le = preprocessing.LabelEncoder()
2 master['result'] = le.fit_transform(master.result.values)
```

This transformed the political parties from text data to numerical data as shown below:

- Conservative Party - 0
- Liberal Democrats Party - 1
- Labour Party - 2

5.3 K Fold Cross Validation

For all my models I made use of the K Fold cross validation. This was done using the sklearn library. I used a 10 split K folks with the random state being none and shuffle being none as well. This process involves dividing the data into 10 parts with 9 used to train the model and 1 for testing the model. The process is iterated 10 times so that each split is used to test the model once and is used to train the data 9 times.

```
1 cv = KFold(n_splits=10, random_state=None, shuffle=False)
2 for train_index, test_index in cv.split(train1):
3   X_train, X_test = train1.reindex(train_index), train1.reindex(test_index)
4   y_train, y_test = train_ys1.reindex(train_index),
5 train_ys1.reindex(test_index).reset_index(drop=True)
```

5.4 Decision Tree Model

5.4.1 Training

This was the first model to be tried and the scikit-learn library was used to access the algorithm.

```
1 DTC1 = tree.DecisionTreeClassifier()
```

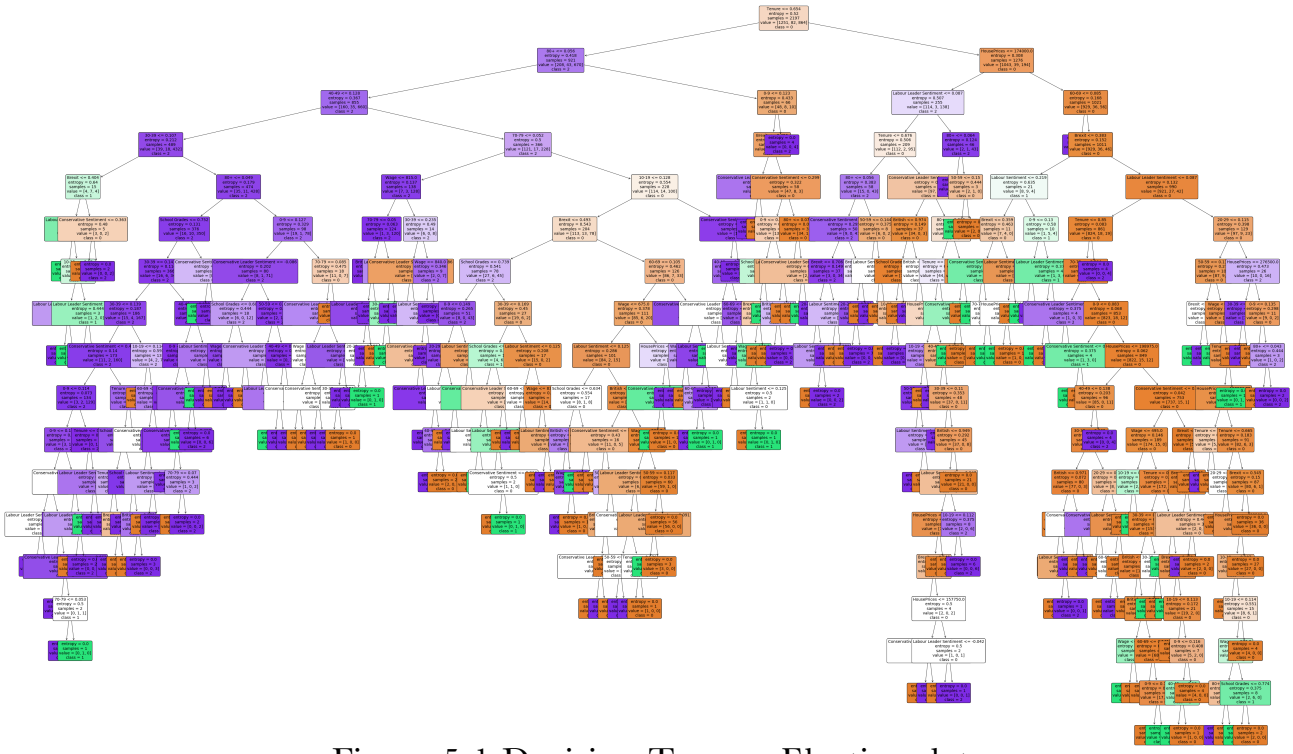
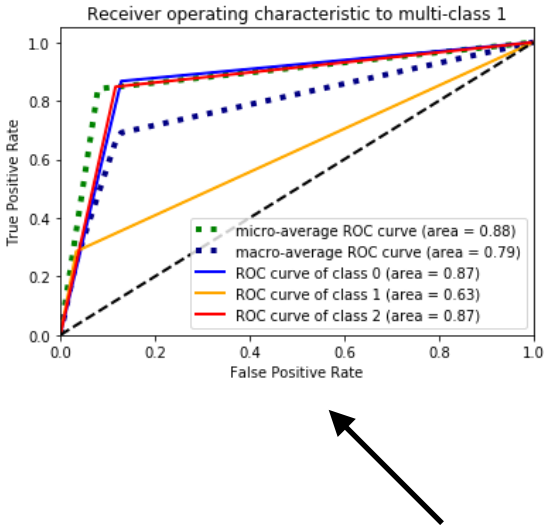


Figure 5.1 Decision Tree on Election data

On the top layer with an entropy of 0.52, the decision tree classifier splits the data on the Tenure feature as it is the feature that causes the most information gain. All data points with a tenure is <= 0.654 goes to the left and all data points > 0.654 goes to the right side. The left side with an entropy of 0.418 is then us split on the 80+ feature. All datapoints <= 0.056 goes to the left side while all > 0.056 goes to the right. While on the right of the root node with an entropy of 0.308, the node is split on the house prices feature. All datapoints with house prices <=174000 goes to the left while all data points > 174000 goes to the right. This goes on as each node is further split on a feature that gives the most information gain until it reaches a node that has 0 entropy and therefore does not need to be split further. This node then becomes a leaf of the decision tree.

5.4.2 Evaluation

There are a number of metrics that are used to measure the performance of the model. As k fold cross validation was used the evaluation ended up being the average of its performance in the 10 iterations of the cross validation.



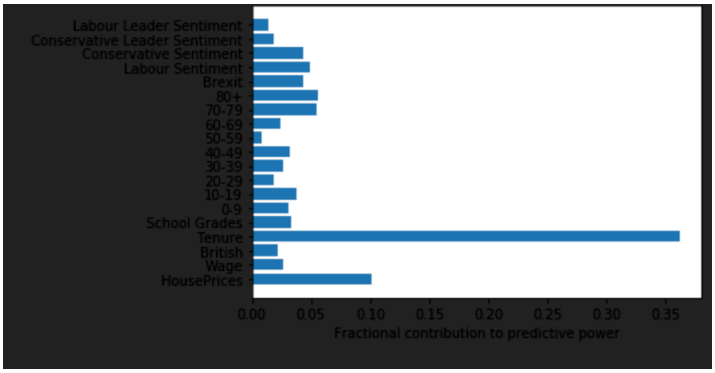


Figure 5.14 Attribute Importance DT

Figure 5.2 ROC Curve for Decision Tree K Fold Split 1

```
Accuracy for Split 1
0.8408163265306122
Confusion Matric for Split 1
Predicted    0  1    2  __all__
Actual
0           97  4   12      113
1            2  1    4        7
2           14  3  108      125
__all__    113  8  124      245
Precision, Recall, F Score for Split 1
(0.6181249405271672, 0.6217547408343869, 0.6197367641658077, None)
```

Figure 5.3 Accuracy and Confusion Matrix for Decision Tree K Fold Split 1

As seen in figure 5.2 and 5.3 the metrics of the first split are:

- Accuracy - 0.841
- Precision - 0.618
- Recall - 0.622
- F1 Score - 0.620

- AUC ROC - 0.79

At the end of the 10 iterations of the cross validations, the mean is calculated from all the splits.

```
Accuracy =0.7918685178989627
Precision =0.5905766426181028
Recall =0.5825708206395604
F1 Score =0.5791573178631849
ROC AUC Score =0.671005672789089
```

Figure 5.4 Metrics of Cross Validation for Decision Tree

From figure 5.4 we can see the final metrics are:

- Accuracy - 0.792
- Precision - 0.591
- Recall - 0.583
- F1 Score - 0.579
- AUC ROC - 0.671

From these metrics we can say that the model is above average. It has a good accuracy but its precision, recall and f1 score are quite average with AUC being a high average.

5.5 Random Forest Model

5.5.1 Training

This model is also made using SciKit-Learn.

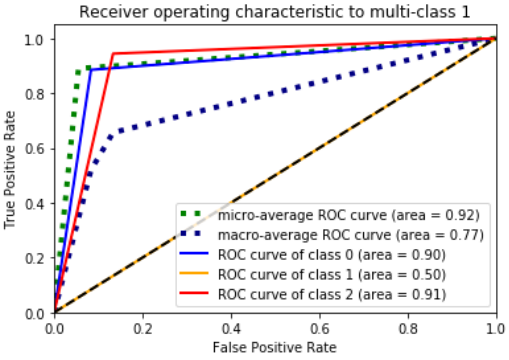
```
RF1 = RandomForestClassifier(n_estimators=1000)
```

It uses a similar algorithm to decision tree but instead of having only one tree it has 1000 as estimators it has been set to 1000. There are 1000 decision trees

splitting at different features in order to make the results as diverse as possible. A sort of vote is had on predictions and the final prediction is calculated as the prediction which is the most common among all the 1000 decision trees.

5.5.2 Evaluation

Again K Fold Cross validation is used in a 10 split iteration. So the data is split into 10 parts and trained 10 times with the test split changed each iteration.



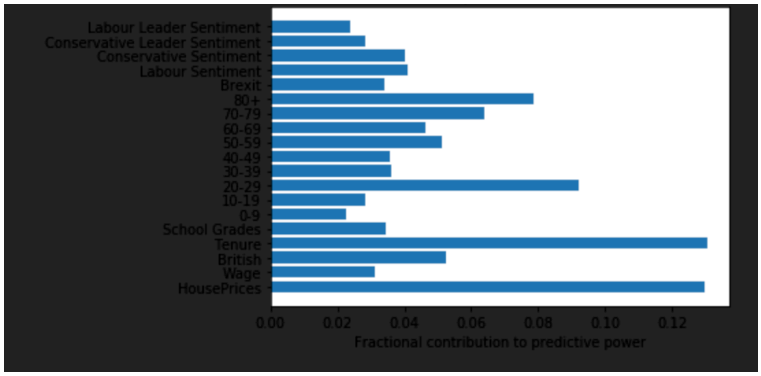


Figure 5.14 Attribute Importance RF

Figure 5.5 ROC Curve for Random Forest K Fold Split 1

```
Accuracy for Split 1
0.8857142857142857
Confusion Matric for Split 1
Predicted    0  1    2  __all__
Actual
0          100  0   13      113
1            3  0    4        7
2            8  0  117      125
__all__    111  0  134      245
Precision, Recall, F Score for Split 1
(0.5913450764197034, 0.6069852507374631, 0.5987773487773488, None)
```

Figure 5.6 Accuracy and Confusion Matrix for Random Forest K Fold Split 1

The metrics of the first split are:

- Accuracy - 0.886
- Precision - 0.591
- Recall - 0.607
- F1 Score - 0.599
- AUC ROC - 0.77

At the end of the 10 iterations of the cross validations, the mean is calculated from all the splits.

```
Accuracy =0.8471779859484778
Precision =0.5964622332749318
Recall =0.5860038124706511
F1 Score =0.5772196851157879
ROC AUC Score =0.7101294792063464
```

Figure 5.7 Metrics of Cross Validation for Random Forest

From figure 5.7 we can see the final metrics from the means are:

- Accuracy - 0.847
- Precision - 0.596
- Recall - 0.586
- F1 Score - 0.577
- AUC ROC - 0.710

From the metrics we can say that this model is good. It has a very good accuracy but its precision, recall and f1 score are on the high average side with the AUC Score being on the good side.

5.6 AdaBoost Model

5.6.1 Training

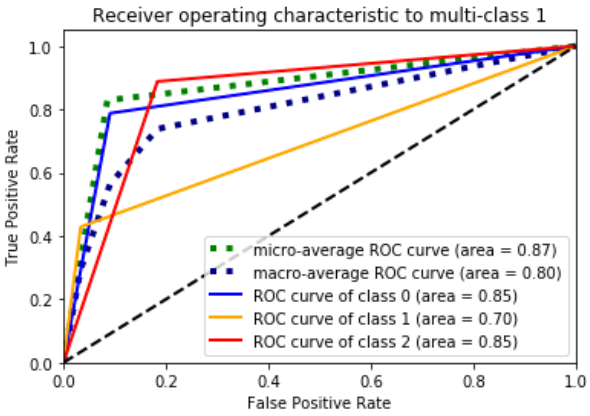
This model is also made using SciKit-Learn.

```
ada1 = AdaBoostClassifier(n_estimators=100)
```

Estimators is set to 100 which means 100 smaller classifiers are used to create the bigger and better classifier. When I fit the training data in it starts by generating a random classifier and by its accuracy assigns a weight to it. Slowly building a better more accurate classifier until a 100 classifiers are built.

5.6.2 Evaluation

With the use of K Fold Cross validation, the data is trained over 10 times with 10 different splits.



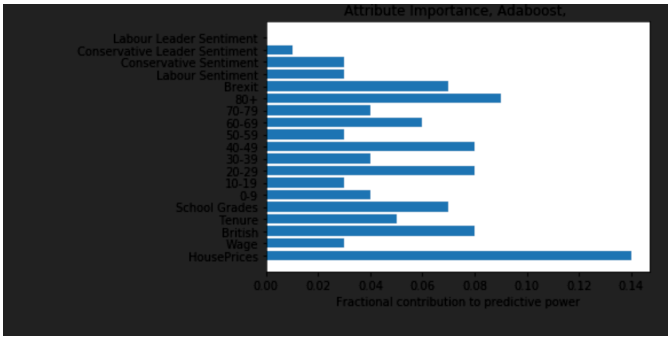


Figure 5.15 Attribute Importance Ada

Figure 5.8 ROC Curve for Ada Boost K Fold Split 1

```
Accuracy for Split 1
0.8285714285714286
Confusion Matric for Split 1
Predicted    0   1    2  __all__
Actual
0           89   4   20      113
1            2   3    2        7
2           10   4  111      125
__all__    101  11  133      245
Precision, Recall, F Score for Split 1
(0.6628339525681892, 0.7013940160134849, 0.6751913835156609, None)
```

Figure 5.9 Accuracy and Confusion Matrix for Ada Boost K Fold Split 1

The metrics of the first split are:

- Accuracy - 0.829
- Precision - 0.663
- Recall - 0.701
- F1 Score - 0.675
- AUC ROC - 0.80

At the end of the 10 iterations of the cross validations, the mean is calculated from all the splits.

```
Accuracy =0.805807962529274
Precision =0.6063406758594871
Recall =0.6114640461593346
F1 Score =0.5999641758369135
ROC AUC Score =0.8029863863296449
```

Figure 5.10 Metrics of Cross Validation for Ada Boost

From figure 5.7 we can see the final metrics from the means are:

- Accuracy - 0.806

- Precision - 0.606
- Recall - 0.611
- F1 Score - 0.5999
- AUC ROC - 0.802

From these metrics we can say this model is good. It has a very high accuracy and AUC Score with nearly good precision, recall and f1 score .

5.7 SVM Model

5.7.1 Training

This model is also made using SciKit-Learn.

```
clfsvmlinear1 = svm.SVC(C= 2.0, degree= 1, kernel= 'rbf')
```

When C = 2 it uses a slightly smaller margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. The degree is set to 1 and the kernel used is ref. On fitting the data into the model it tries to find the best position a hyperplane can split it in order to make correct predictions.

5.7.2 Evaluation

With the use of K Fold Cross validation, the data is trained over 10 times with 10 different splits.

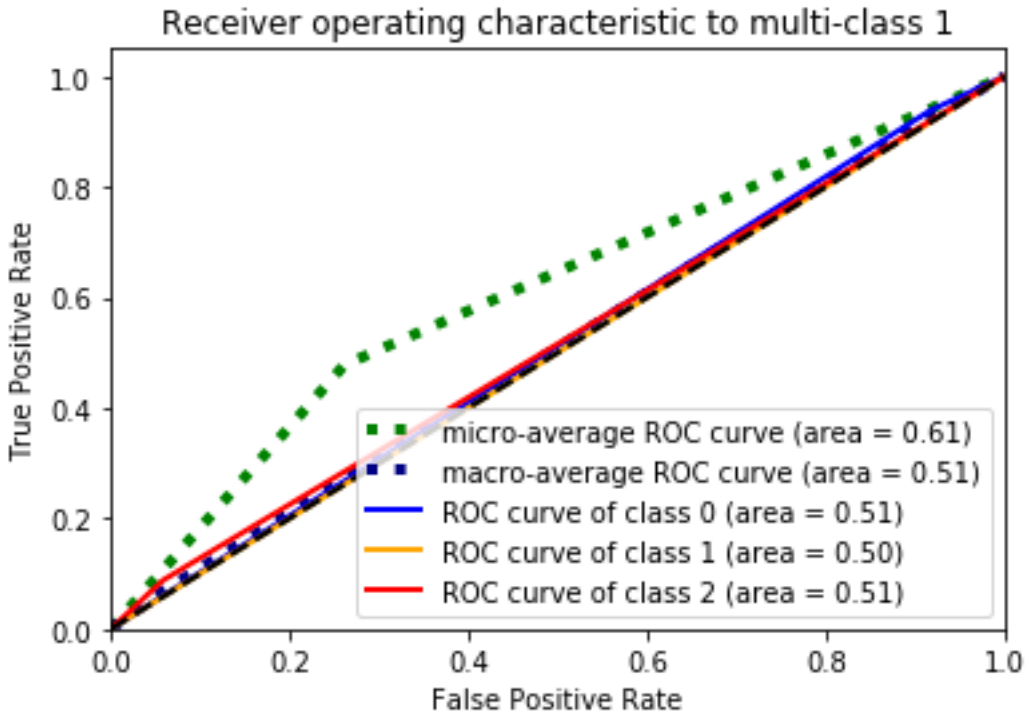


Figure 5.11 ROC Curve for SVM K Fold Split 1

```
Accuracy for Split 1
0.4775510204081633
Confusion Matric for Split 1
Predicted    0  1   2  __all__
Actual
0          106  0   7      113
1            7  0   0        7
2          114  0  11      125
__all__    227  0  18      245
Precision, Recall, F Score for Split 1
(0.5271649335211325, 0.4775510204081633, 0.36607997044971835, None)
```

Figure 5.12 Accuracy and Confusion Matrix for SVM K Fold Split 1

The metrics of the first split are:

- Accuracy - 0.478
- Precision - 0.527
- Recall - 0.478
- F1 Score - 0.366
- AUC ROC - 0.51

At the end of the 10 iterations of the cross validations, the mean is calculated from all the splits.

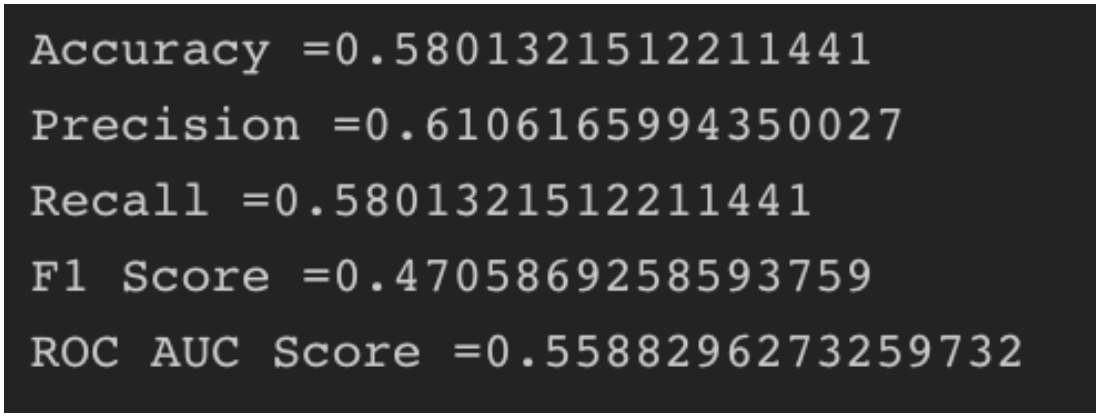


Figure 5.13 Metrics of Cross Validation for SVM

From figure 5.7 we can see the final metrics from the means are:

- Accuracy - 0.58
- Precision - 0.61
- Recall - 0.58
- F1 Score - 0.471
- AUC ROC - 0.558

From these metrics we can say this model has a low average performance a its accuracy, recall and arc score slightly below 0.6. It has a poor f1 score and manages to get slightly above 0.6 on precision.

5.8 Overall Evaluation

Overall the SVM model was the one with the lowest performance as most of the metrics were on the low average side with f1 score dropping to 0.47. The best performing model on the other hand was between the random forest model and the ada boost model but I believe ada boost is better because although it has the lesser average between the two models, Ada Boost model performs better in all the other metrics.

Chapter 6

Testing

In this chapter I would be looking at using the best model from the previous chapter to predict past UK Elections. As the Ada Boost model was the best performing that will be the model used for these predictions.

The def function Election(year): is one i made that when an election year between 2005 and 2019 is entered, a dataset of predictions are returned. Below I will look at each year and see how accurate the election predictions are to the actual results.

6.1 2005 Elections

In the 2005 quite a bit of the data is missing so prediction is not as accurate as can be. Also in 2005 some constituencies were structured different or didn't exist so it did not match with the remaining data.

Predicted	0	1	2	__all__
Actual				
0	119	8	3	130
1	14	12	4	30
2	76	2	125	203
__all__	209	22	132	363

Figure 6.1 2005 Confusion Matrix

	precision	recall	f1-score	support
0	0.56938	0.91538	0.70206	130
1	0.54545	0.40000	0.46154	30
2	0.94697	0.61576	0.74627	203
accuracy			0.70523	363
macro avg	0.68727	0.64372	0.63662	363
weighted avg	0.77856	0.70523	0.70691	363

Figure 6.2 2005 Classification Report

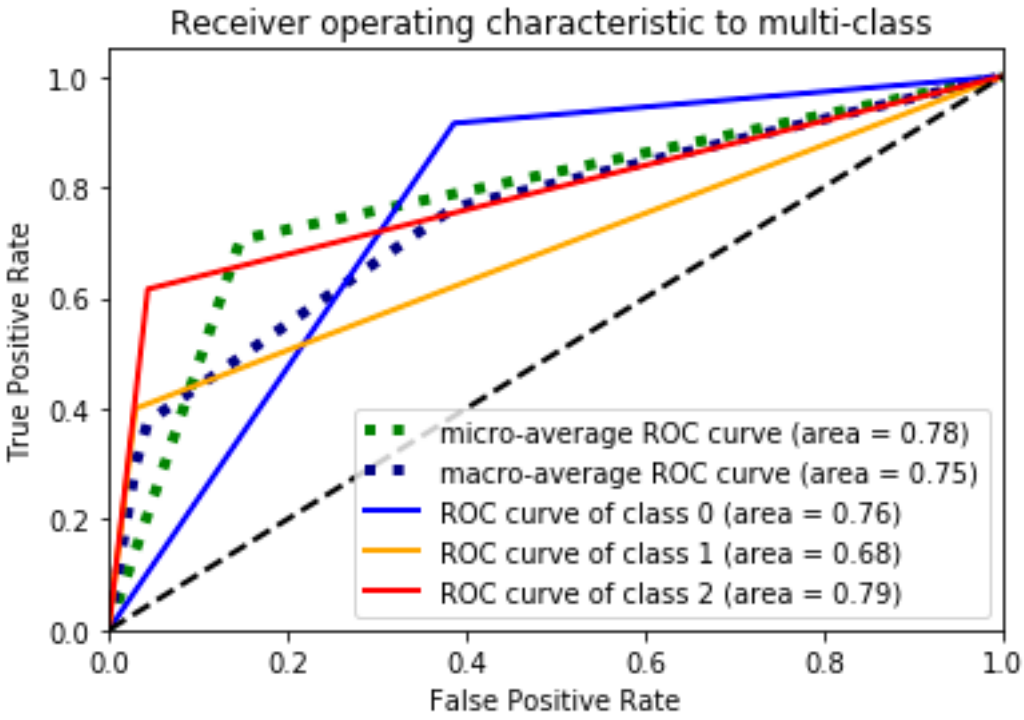


Figure 6.3 2005 ROC Curve

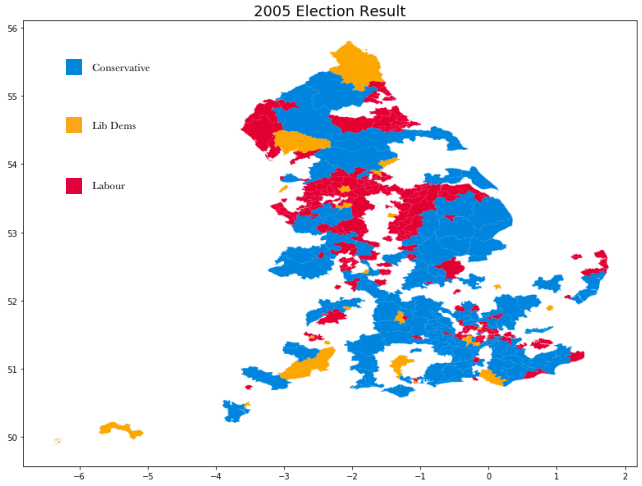


Figure 6.4 2005 Election Result

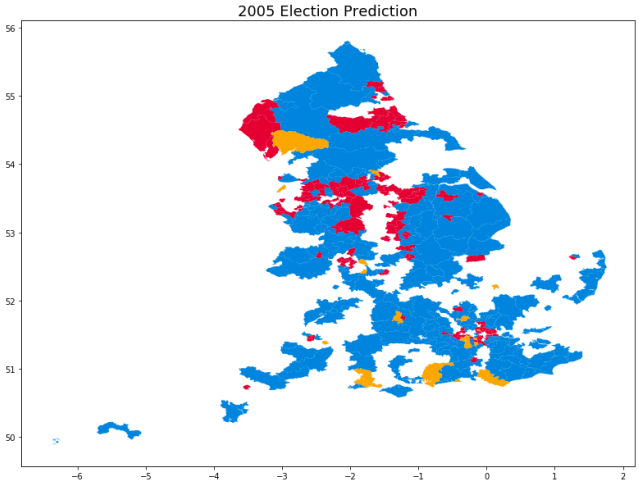


Figure 6.5 2005 Election Prediction

- 2005 Ada Boost Achieved Prediction Accuracy 0.716
- 2005 Ada Boost Achieved Precision 0.685
- 2005 Ada Boost Achieved Recall 0.623
- 2005 Ada Boost Achieved F1 Score 0.620

- 2005 Ada Boost Achieved ROC AUC Score 0.737

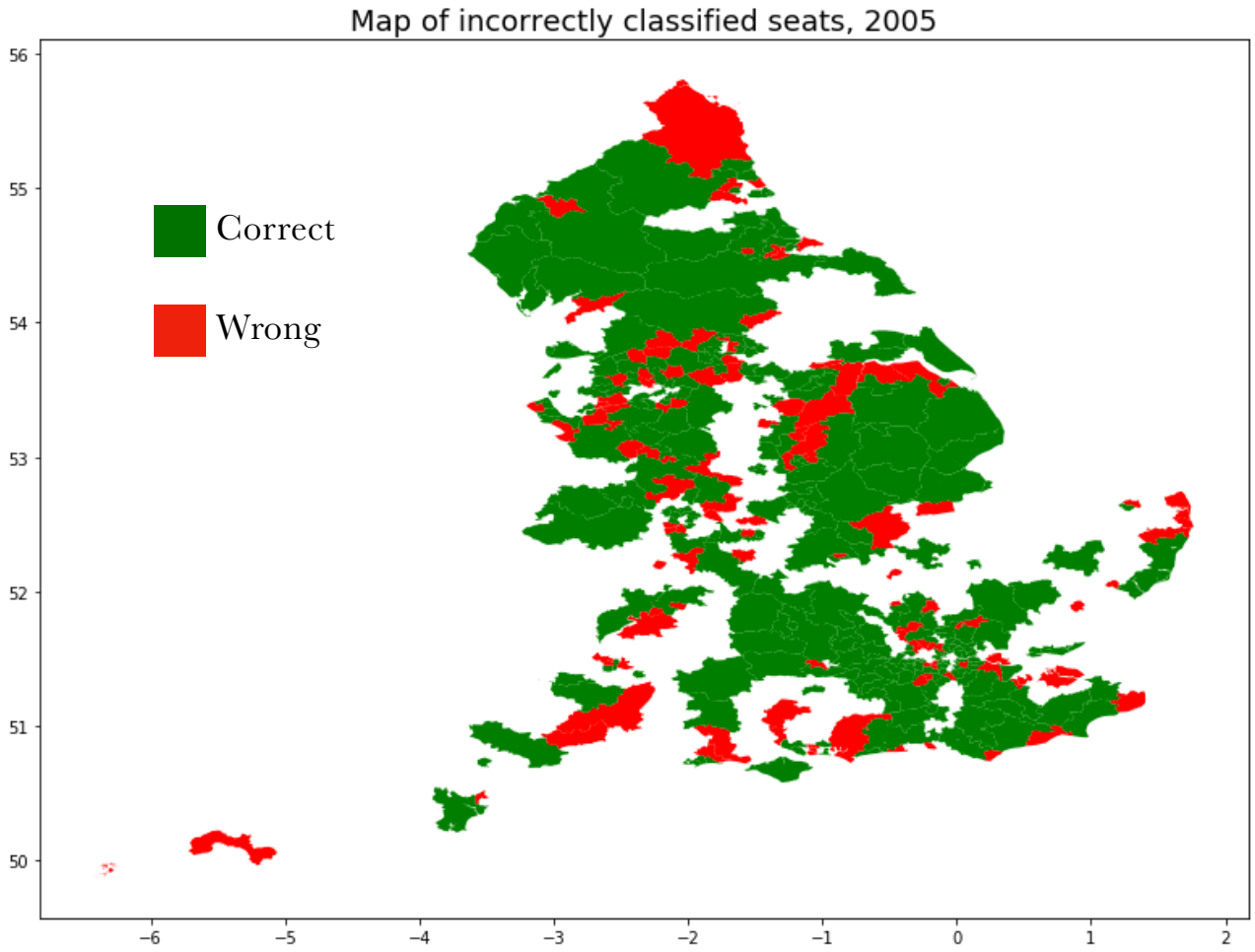


Figure 6.6 2005 Incorrect Predictions

6.2 2010 Elections

Predicted	0	1	2	__all__
Actual				
0	244	3	47	294
1	27	3	10	40
2	9	0	161	170
__all__	280	6	218	504

Figure 6.7 2010 Confusion Matrix

	precision	recall	f1-score	support
0	0.87143	0.82993	0.85017	294
1	0.50000	0.07500	0.13043	40
2	0.73853	0.94706	0.82990	170
accuracy			0.80952	504
macro avg	0.70332	0.61733	0.60350	504
weighted avg	0.79712	0.80952	0.78621	504

Figure 6.8 2010 Classification Report

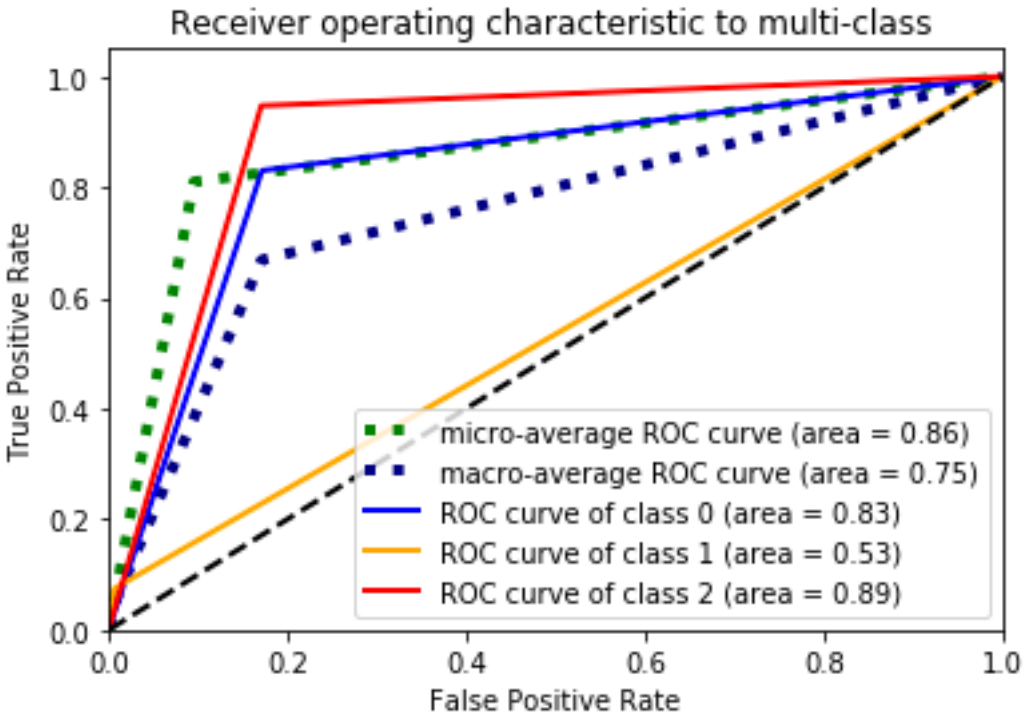


Figure 6.9 2010 ROC Curve

- 2010 Ada Boost Achieved Prediction Accuracy 0.809
- 2010 Ada Boost Achieved Precision 0.703
- 2010 Ada Boost Achieved Recall 0.617
- 2010 Ada Boost Achieved F1 Score 0.604
- 2010 Ada Boost Achieved ROC AUC Score 0.751

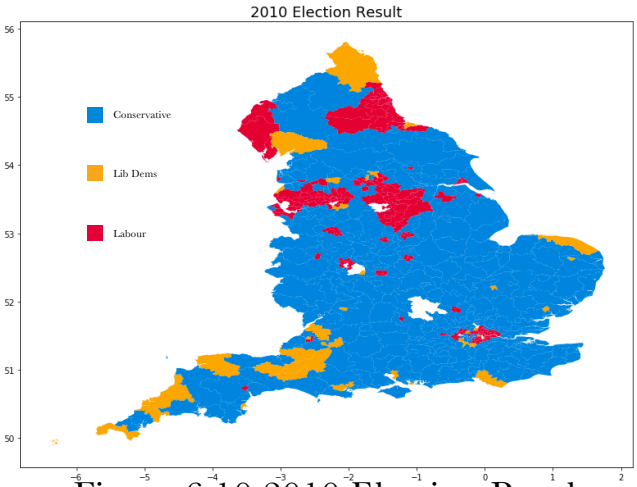


Figure 6.10 2010 Election Result

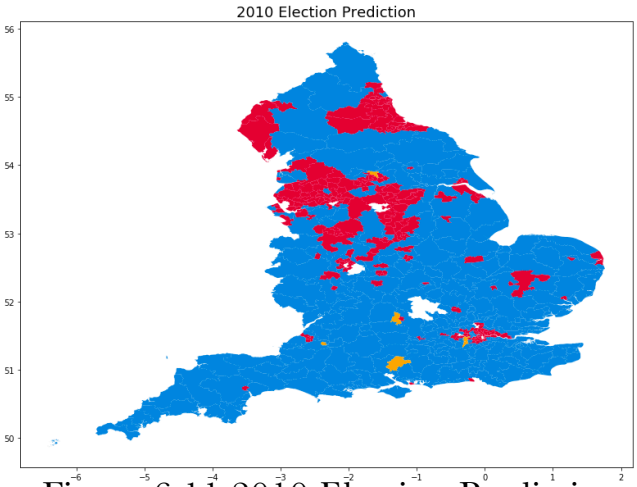


Figure 6.11 2010 Election Prediction

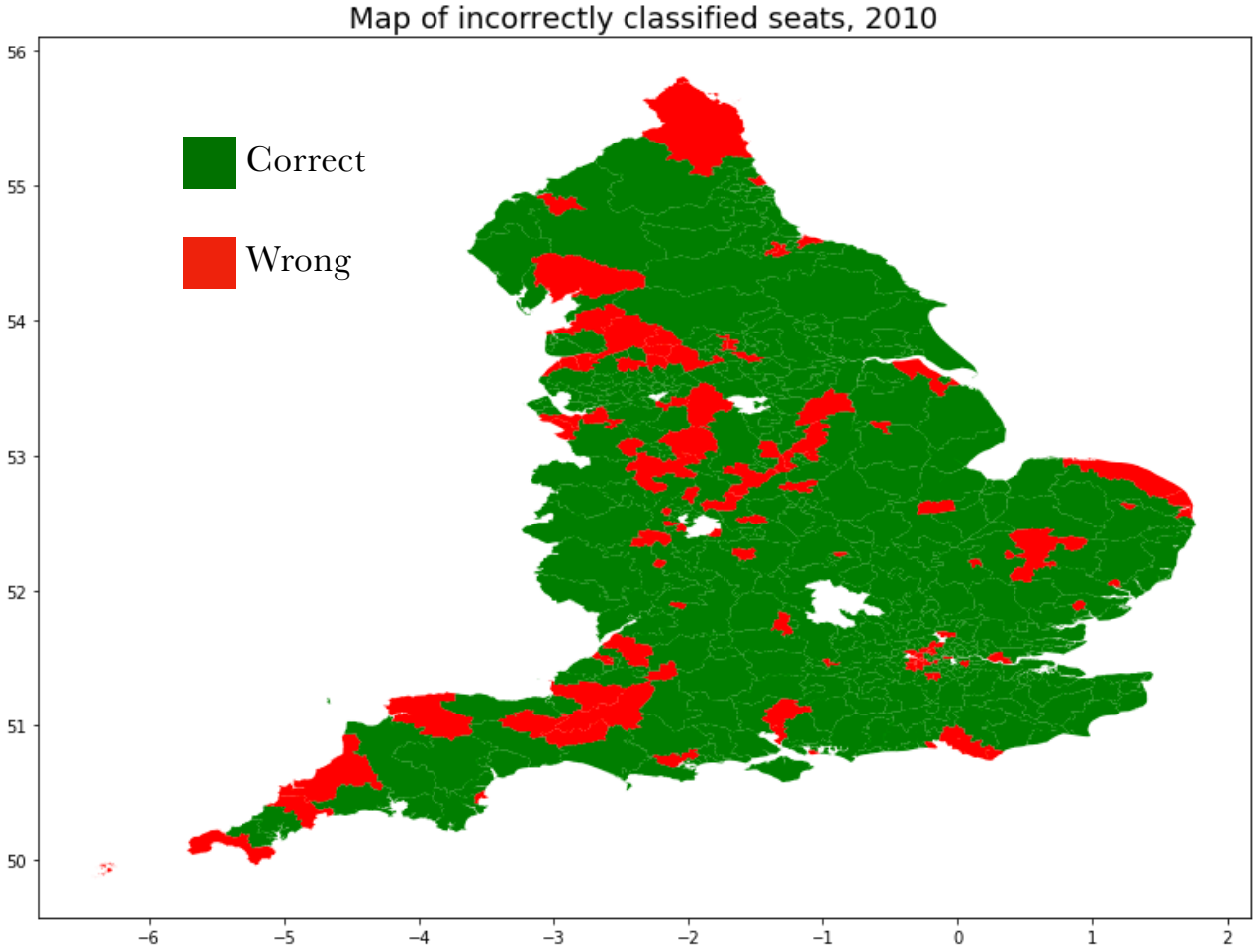


Figure 6.12 2010 Incorrect Predictions

6.3 2015 Elections

Predicted	0	1	2	__all__
Actual				
0	236	16	61	313
1	3	2	1	6
2	6	2	187	195
__all__	245	20	249	514

Figure 6.13 2015 Confusion Matrix

	precision	recall	f1-score	support
0	0.96327	0.75399	0.84588	313
1	0.10000	0.33333	0.15385	6
2	0.75100	0.95897	0.84234	195
accuracy			0.82685	514
macro avg	0.60476	0.68210	0.61402	514
weighted avg	0.87266	0.82685	0.83646	514

Figure 6.14 2015 Classification Report

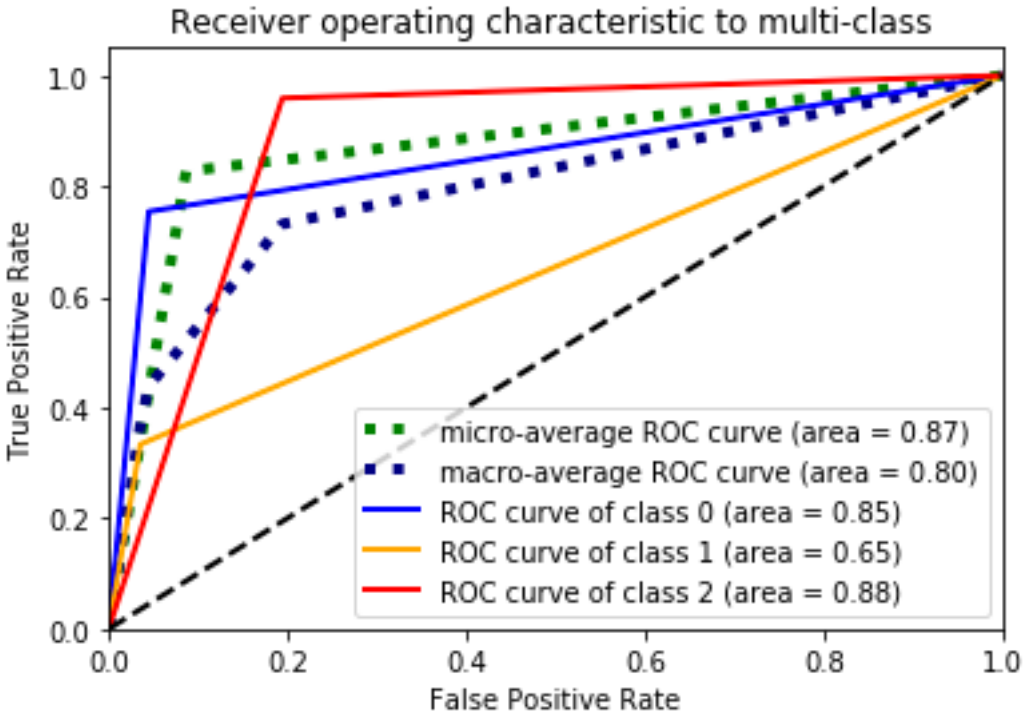


Figure 6.15 2015 ROC Curve

- 2015 Ada Boost Achieved Prediction Accuracy 0.827
- 2015 Ada Boost Achieved Precision 0.605
- 2015 Ada Boost Achieved Recall 0.682
- 2015 Ada Boost Achieved F1 Score 0.614
- 2015 Ada Boost Achieved ROC AUC Score 0.795

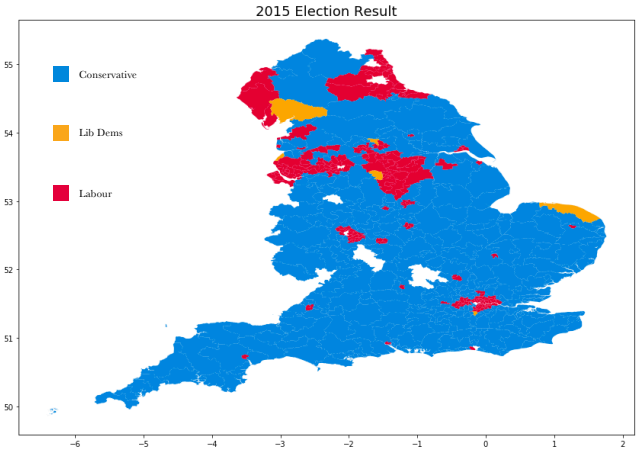


Figure 6.16 2015 Election Result

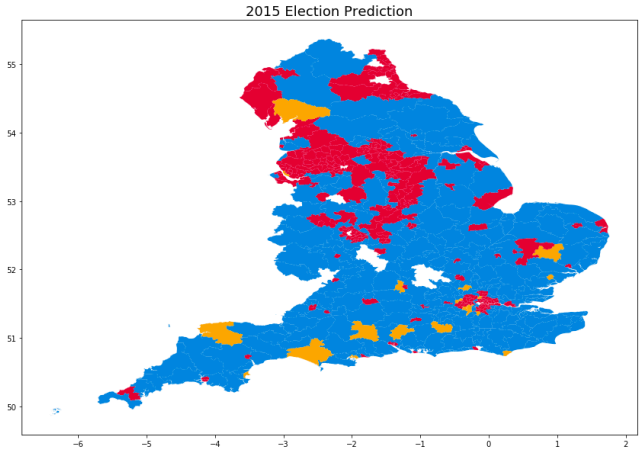


Figure 6.17 2015 Election Precision

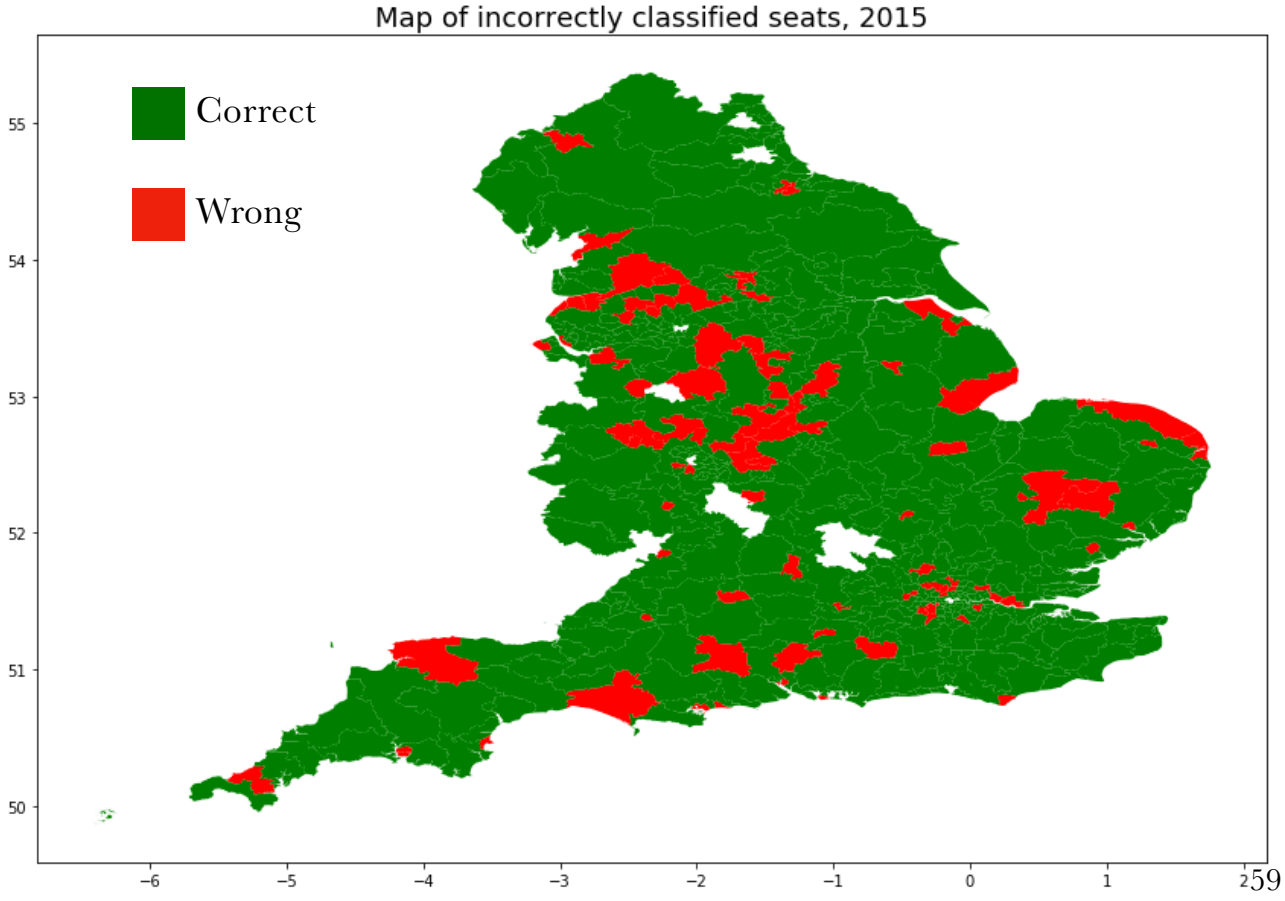


Figure 6.18 2015 Incorrect Predictions

6.4 2017 Elections

Predicted	0	1	2	__all__
Actual				
0	261	4	31	296
1	5	3	0	8
2	25	2	199	226
__all__	291	9	230	530

Figure 6.19 2017 Confusion Matrix

	precision	recall	f1-score	support
0	0.89691	0.88176	0.88927	296
1	0.33333	0.37500	0.35294	8
2	0.86522	0.88053	0.87281	226
accuracy			0.87358	530
macro avg	0.69849	0.71243	0.70501	530
weighted avg	0.87489	0.87358	0.87415	530

Figure 6.20 2017 Classification Report

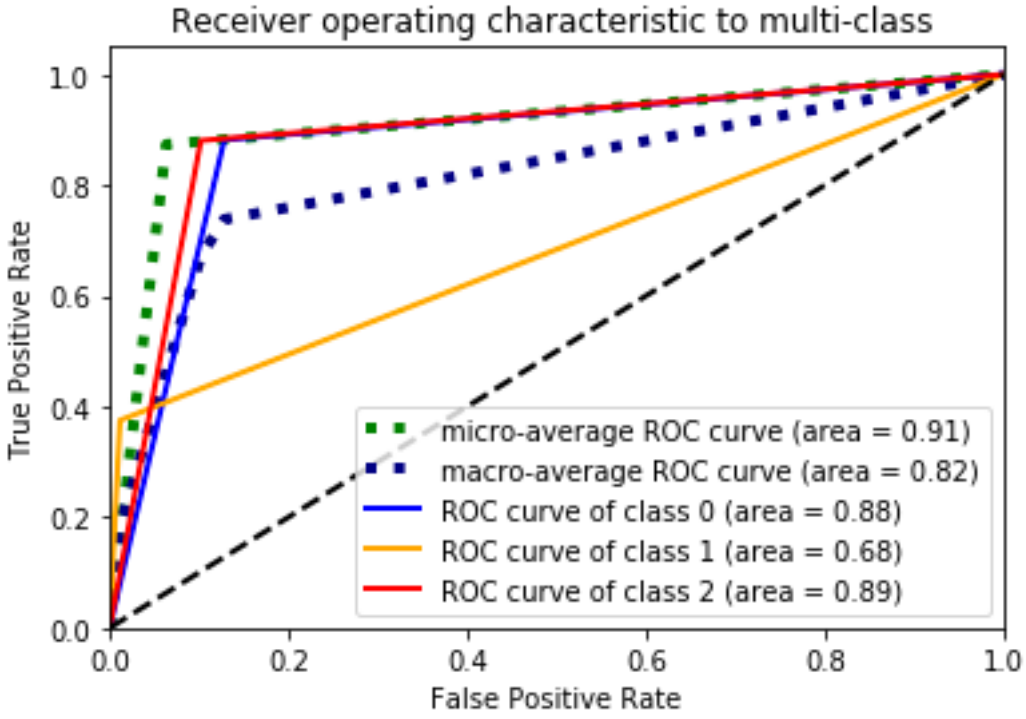


Figure 6.21 2017 ROC Curve

- 2017 Ada Boost Achieved Prediction Accuracy 0.877
- 2017 Ada Boost Achieved Precision 0.726
- 2017 Ada Boost Achieved Recall 0.837
- 2017 Ada Boost Achieved F1 Score 0.764
- 2017 Ada Boost Achieved ROC AUC Score 0.881

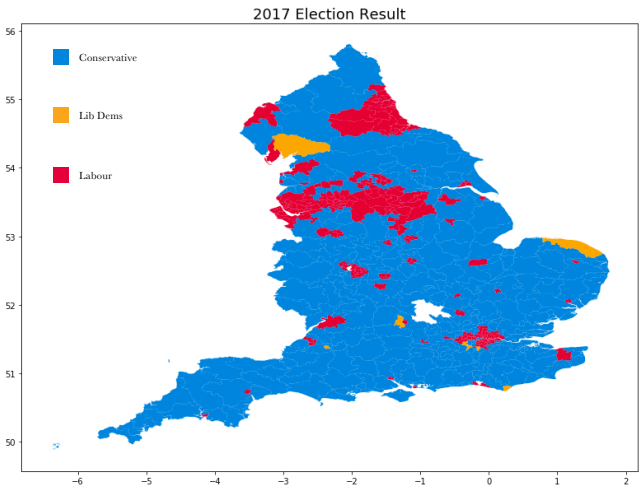


Figure 6.22 2017

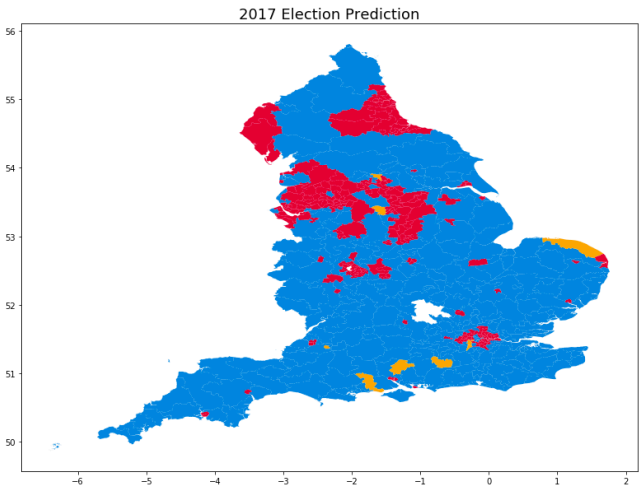


Figure 6.23 2017

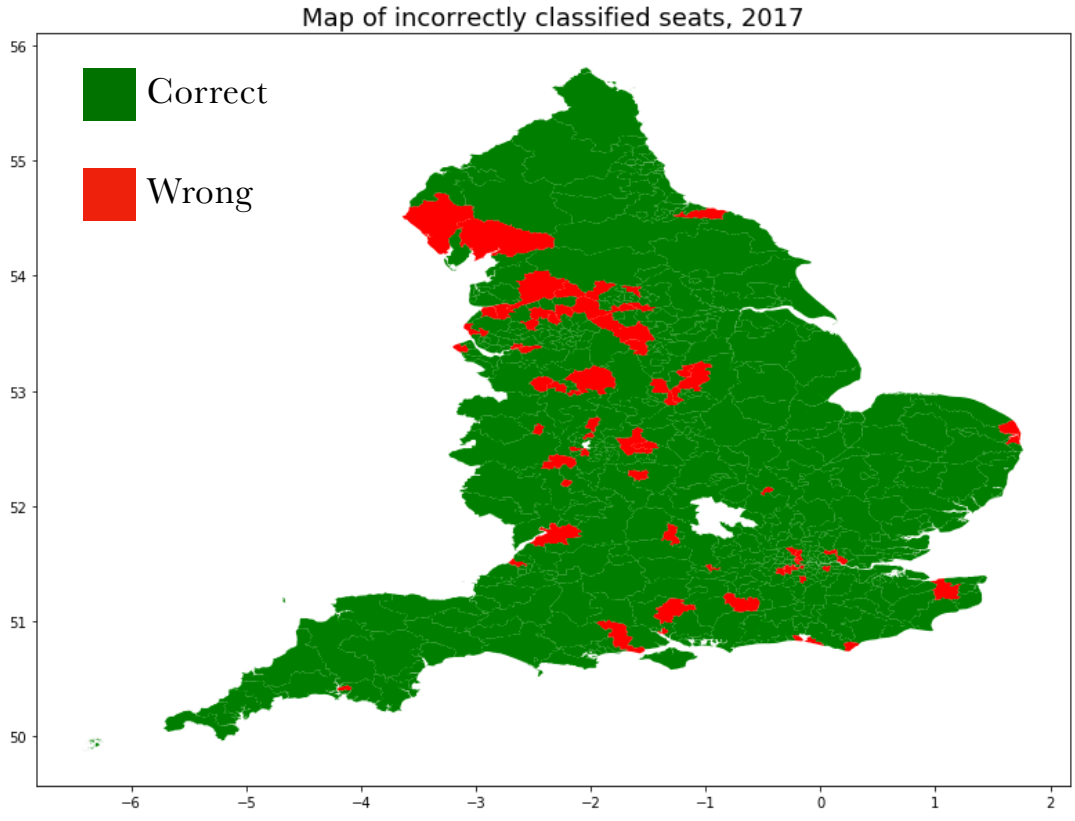


Figure 6.24 2017 Incorrect Predictions

6.5 2019 Elections

Predicted	0	1	2	__all__
Actual				
0	277	0	67	344
1	6	1	0	7
2	19	0	160	179
__all__	302	1	227	530

Figure 6.25 2019 Confusion Matrix

	precision	recall	f1-score	support
0	0.91722	0.80523	0.85759	344
1	1.00000	0.14286	0.25000	7
2	0.70485	0.89385	0.78818	179
accuracy			0.82642	530
macro avg	0.87402	0.61398	0.63192	530
weighted avg	0.84659	0.82642	0.82612	530

Figure 6.26 2019 Classification Report

- 2019 Ada Boost Achieved Prediction Accuracy 0.826
- 2019 Ada Boost Achieved Precision 0.874
- 2019 Ada Boost Achieved Recall 0.613
- 2019 Ada Boost Achieved F1 Score 0.632
- 2019 Ada Boost Achieved ROC AUC Score 0.753

Figure 6.31 Subset of 2019 Prediction Table

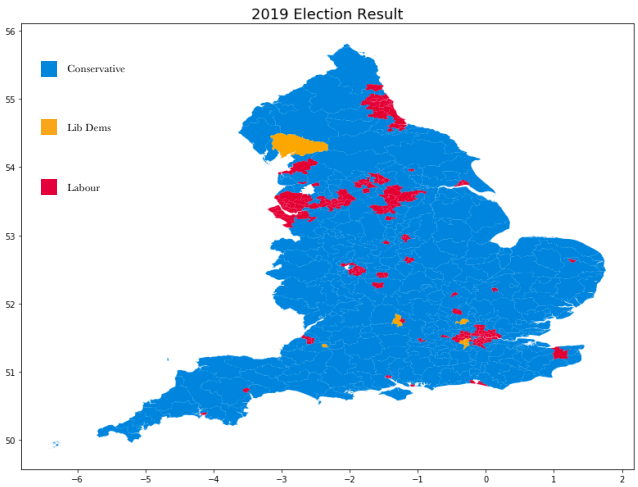


Figure 6.28 2019 Election Result

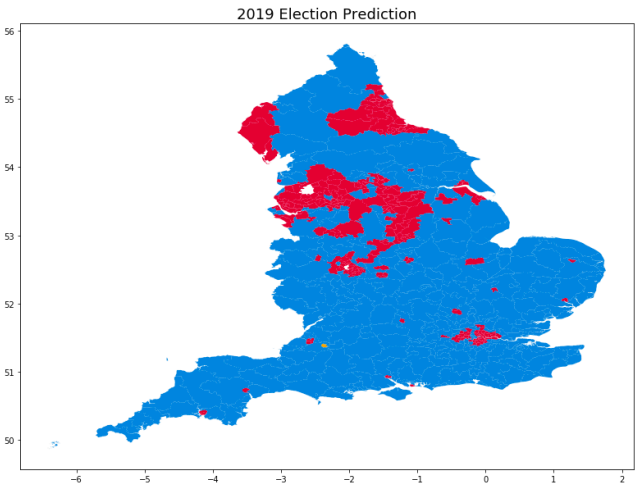


Figure 6.29 2019 Election Prediction

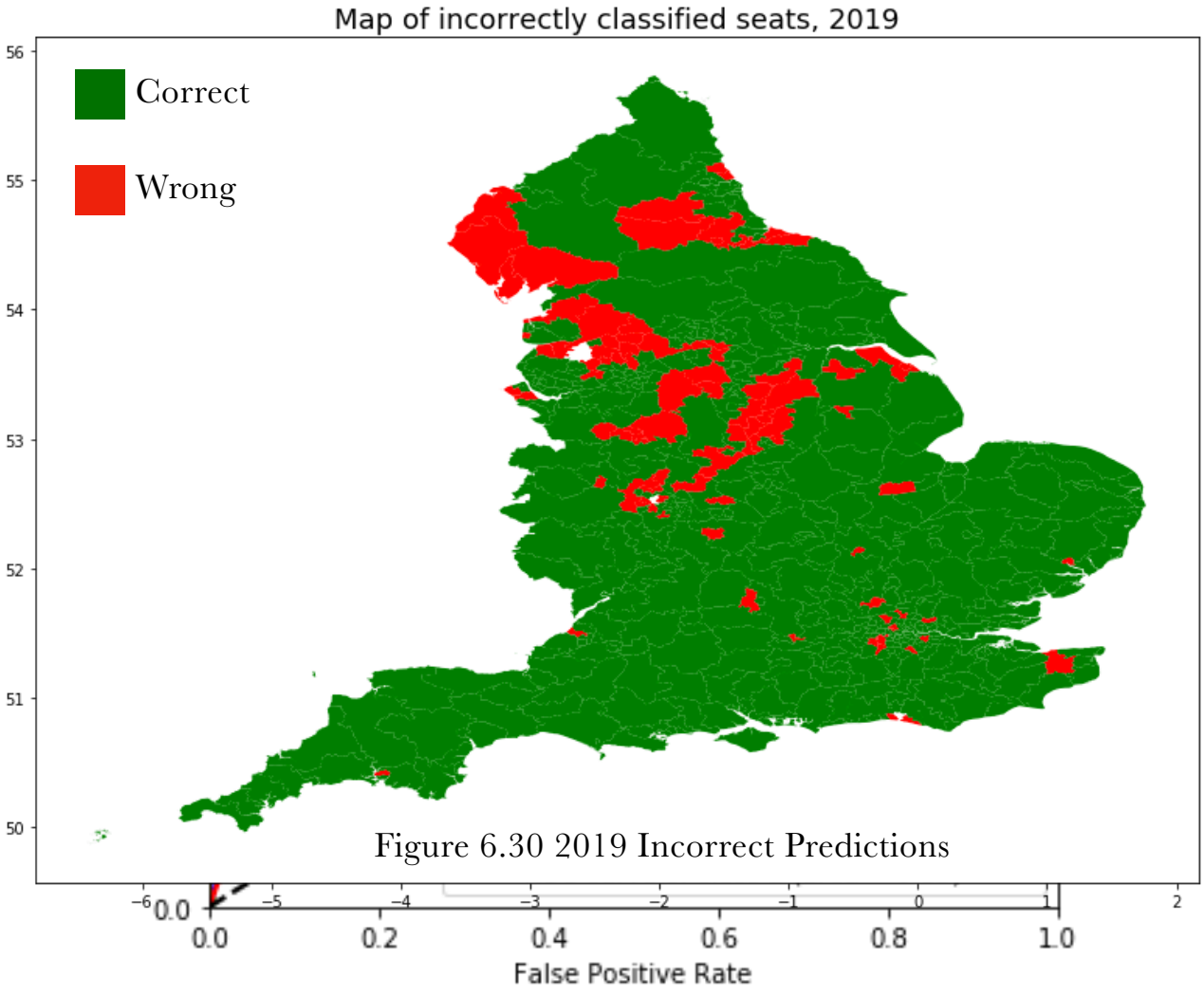


Figure 6.30 2019 Incorrect Predictions

Figure 6.27 2019 ROC Curve

6.6 Summary

From these tests we can see that indeed the adaBoost classifier that I have created is has a high performance and accuracy as in all the years, the lowest accuracy was over 0.7 and the highest was 0.87 which is a very good accuracy score.The charts are also shown to see the distribution of the predictions and make it more understandable.

Chapter 7

Conclusion

So far this paper has outlined the motivation for the project, the existing literature, the inspiration and how to create and choose the adequate model to make a certain prediction. This chapter puts that work into context, discussing how the project and its results compare to similar work. This chapter also explores the directions that the project may take beyond this paper and potential improvements.

7.1 Future Work

Majority of the requirements were fulfilled apart from the part in which I could not access Twitter data. Therefore these are things I would like to add to this project in the future.

- Use live data from social media such as Twitter where I can mine live text to expand the dataset.
- Expand model to be able to predict elections of other countries.

529	York Outer	Con,	Constituencies
0	Aldershot	Con	
1	Aldridge-Brownhills	Con	
2	Altrincham and Sale West	Con	
3	Amber Valley	Con	
4	Arundel and South Downs	Con	
5	Ashfield	Lab	
6	Ashford	Con	
7	Ashton-under-Lyne	Lab	
8	Aylesbury	Con	
9	Banbury	Con	
10	Barking	Lab	
11	Barnsley Central	Lab	
12	Barnsley East	Lab	
13	Barrow and Furness	Lab	
14	Basildon and Billericay	Con	
15	Basingstoke	Con	
16	Bassetlaw	Lab	
17	Bath	LD	
18	Batley and Spen	Lab	
19	Battersea	Con	
20	Beaconsfield	Con	
21	Beckenham	Con	
22	Bedford	Con	
23	Bermondsey and Old Southwark	Lab	
24	Berwick-upon-Tweed	Con	
25	Bethnal Green and Bow	Lab	
26	Beverley and Holderness	Con	
27	Bexhill and Battle	Con	
28	Bexleyheath and Crayford	Con	
29	Birkenhead	Lab	
30	Birmingham, Edgbaston	Lab	

7.2 Conclusion

This paper aimed to create an adequate model for predicting UK election outcome in England. Decision tree, Random forest, adaBoost, SVM models were looked into and evaluated and after adequate testing adaBoost was chosen as the most adequate out of the options. The model did not let down as it was able to score a high prediction accuracy in all elections given to it.

Bibliography

[1] BBC. (2019) General election 2019: What is an opinion poll? [online] Available at: www.bbc.co.uk/news/election-2019-50247874

[2] BBC. (2019) Exit polls: How accurate are they? [online] Available at: www.bbc.co.uk/news/election-2019-50716626

[3] Britannica. Parliamentary System.[online] Available at: www.britannica.com/topic/parliamentary-system

[4] Electoral Reform Society. (2017) First Past the Post. [online] Available at: www.electoral-reform.org.uk/voting-systems/types-of-voting-system/first-past-the-post

[5] Grofman, B., Trechsel, A. and Franklin, M., 2014. The internet and democracy in global perspective. *Voters, candidates, parties, and social movements introduction*.

[6] Dimova, G., 2019. Democracy Beyond Elections: Government Accountability in the Media Age. Springer Nature.

[7] Bimber, B. and Davis, R., 2003. Campaigning online: The Internet in US elections. Oxford University Press.

[8] MacLeod, A. ed., 2019. Propaganda in the information age: Still manufacturing consent. Routledge.

[9] Franch, F., 2013. (Wisdom of the Crowds) 2: 2010 UK election prediction with social media. *Journal of Information Technology & Politics*, *10*(1), pp.57-71.

[10] Autoregressive integrated moving average.[online] Available at: en.wikipedia.org/wiki/Autoregressive_integrated_moving_average

[11] Sang, E.T.K. and Bos, J., 2012, April. Predicting the 2011 dutch senate election results with twitter. In *Proceedings of the workshop on semantic analysis in social media* (pp. 53-60).

[12] What is Web Scraping ?[online] Available at: www.webharvy.com/articles/what-is-web-scraping.

[13] Dogucu, M. and Çetinkaya-Rundel, M., 2020. Web Scraping in the Statistics and Data Science Curriculum: Challenges and Opportunities. *Journal of Statistics Education*, pp.1-11.

[14] Top 10 Best Scraping APIs.[online] Available at: blog.api.rakuten.net/top-scraping-apis

[15] Dewi, L.C. and Chandra, A., 2019. Social Media Web Scraping using Social Media Developers API and Regex. *Procedia Computer Science*, *157*, pp.444-449.

[16] León-Borges, J.A., Noh-Balam, R.I., Gómez, L.R. and Strand, M.P., 2017. The machine learning in the prediction of elections-El aprendizaje automático en la predicción de las elecciones. *ReCIBE, Revista electrónica de Computación, Informática, Biomédica y Electrónica*, *4*(2).

[17] Tsakalidis, A., Papadopoulos, S., Cristea, A.I. and Kompatsiaris, Y., 2015. Predicting elections for multiple countries using Twitter and polls. *IEEE Intelligent Systems*, *30*(2), pp.10-17.

[18] Ramteke, J., Shah, S., Godhia, D. and Shaikh, A., 2016, August. Election result prediction using Twitter sentiment analysis. In *2016 international conference on inventive computation technologies (ICICT)* (Vol. 1, pp. 1-5). IEEE.

[19] Sharma, P. and Moh, T.S., 2016, December. Prediction of Indian election using sentiment analysis on Hindi Twitter. In *2016 IEEE international conference on big data (big data)* (pp. 1966-1971). IEEE.

[20] Garbade, M.J., 2018. A Simple Introduction to Natural Language Processing. Medium. Tillgängling online: https://becominghuman. ai/a-simple-introduction-to-naturallanguage-processing-ea66a1747b32 (2019-11-01).

[21] Oyebode, O. and Orji, R., 2019, October. Social media and sentiment analysis: The nigeria presidential election 2019. In *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)* (pp. 0140-0146). IEEE.

[22] Digite. (2020). *What Is Kanban? An Overview Of The Kanban Method*. [online] Available at: https://www.digite.com/ kanban/what-is-kanban/

[23] Zenkit. (2020). *Agile Methodology: An Overview | Zenkit*. [online] Available at: https://zenkit.com/en/blog/agile- methodology-an-overview/

[24] Towardsdatascience. (2020). *"Leveraging on NLP to gain insights in Social Media, News & Broadcasting"* [online] Available at: https://towardsdatascience.com/leveraging-on-nlp-to-gain-insights-in-social-media-news-broadcasting-ca89752ef638

[25] Towardsdatascience. (2020). *"SENTIMENTAL ANALYSIS USING VADER interpretation and classification of emotions"* [online] Available at: https://towardsdatascience.com/sentimental-analysis-using-vader-a3415fef7664

[26] Towardsdatascience. (2018). "Introduction to Data Preprocessing in Machine Learning" [online] https://towardsdatascience.com/introduction-to-data-preprocessing-in-machine-learning-a9fa83a5dc9d

[26] Towardsdatascience. (2020). "All Machine Learning Models Explained in 6 Minutes" [online] https://towardsdatascience.com/all-machine-learning-models-explained-in-6-minutes-9fe30ff6776a

[27] Towardsdatascience. (2019). "Decision Tree Classification" [online] https://towardsdatascience.com/https-medium-com-lorrli-classification-and-regression-analysis-with-decision-trees-c43cdbc58054

[28] Towardsdatascience. (2019). "Understanding Random Forest" [online] https://towardsdatascience.com/understanding-random-forest-58381e0602d2

[29] analyticsvidhya. (2017). "Understanding Support Vector Machine(SVM) algorithm from examples (along with code)" [online] https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

[30] Towardsdatascience. (2020). "How to Best Evaluate a Classification Model" [online] https://towardsdatascience.com/how-to-best-evaluate-a-classification-model-2edb12bcc587

[31] Tweepy Documentation [online] http://docs.tweepy.org/en/latest/

[32] NLTK 3.5 Documentation [online] https://www.nltk.org/

[33] BeautifulSoup4 4.91 Documentation [online] https://pypi.org/project/beautifulsoup4/

[34] Scikit-learn 0.21.3 Documentation [online] https://scikit-learn.org/0.21/documentation.html

[35] Matplotlib 3.3.2 Documentation [online] https://matplotlib.org/users/index.html

[36] Pandas Documentation [online] https://pandas.pydata.org/docs/user_guide/index.html

[37] Numpy 1.19 Documentation [online] https://numpy.org/doc/stable/

[38] Seaborn [online] https://seaborn.pydata.org/introduction.html

[39] GeoPandas [online] https://geopandas.org/

Appendix A

Decision Tree Model

```
toprow = ["Random", "Med. House Price", "Med. Wage", "Prop. British",
"Tenure", "Grades", "0-9", "10-19", "20-29", "30-39", "40-49", "50-59",
"60-69", "70-79", "80+", "Brexit",
          "Labour Sentiment","Conservative Sentiment","Conservative Leader
Sentiment","Labour Leader Sentiment","result2017"]
toprow= list(train1.columns.values)
scores=[]
precisions=[]
recalls=[]
fscores=[]
p=1
DTC1 = tree.DecisionTreeClassifier()
for train_index, test_index in cv.split(train1):
    X_train, X_test = train1.reindex(train_index), train1.reindex(test_index)
    y_train, y_test = train_ys1.reindex(train_index),
train_ys1.reindex(test_index).reset_index(drop=True)
    DTC1.fit(X_train, y_train)
    print('Accuracy for Split '+ str(p))
    print(DTC1.score(X_test, y_test))
    scores.append(DTC1.score(X_test, y_test))
    pred = DTC1.predict(X_test)
    print('Confusion Matric for Split '+ str(p))
    print(ConfusionMatrix(y_test, pred))
    print('Precision, Recall, F1 Score for Split '+str(p))
    print(precision_recall_fscore_support(y_test, pred, average='macro'))
    precision,recall,fscore,support=precision_recall_fscore_support(y_test,
pred, average='macro')
    precisions.append(precision)
    recalls.append(recall)
    fscores.append(fscore)

    predbin = label_binarize(pred,classes=[0,1,2,])
    y_testbin = label_binarize(y_test,classes=[0,1,2,])
    n_classes = predbin.shape[1]

    fpr = dict()
    tpr = dict()
    roc_auc = dict()
```

```
    for i in range(n_classes):
        fpr[i], tpr[i], _ = roc_curve(y_testbin[:, i], predbin[:, i])
        roc_auc[i] = auc(fpr[i], tpr[i])

    fpr["micro"], tpr["micro"], _ = roc_curve(y_testbin.ravel(),
predbin.ravel())
    roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

    # First aggregate all false positive rates
    all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))

    # Then interpolate all ROC curves at this points
    mean_tpr = np.zeros_like(all_fpr)
    for i in range(n_classes):
        mean_tpr += interp(all_fpr, fpr[i], tpr[i])

    # Finally average it and compute AUC
    mean_tpr /= n_classes

    fpr["macro"] = all_fpr
    tpr["macro"] = mean_tpr
    roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

    # Plot all ROC curves
    plt.figure()
    plt.plot(fpr["micro"], tpr["micro"],
             label='micro-average ROC curve (area = {0:0.2f})'
                   ''.format(roc_auc["micro"]),
             color='green', linestyle=':', linewidth=4)

    plt.plot(fpr["macro"], tpr["macro"],
             label='macro-average ROC curve (area = {0:0.2f})'
                   ''.format(roc_auc["macro"]),
             color='navy', linestyle=':', linewidth=4)

    colors = cycle(['blue', 'orange', 'red'])
    for i, color in zip(range(n_classes), colors):
        plt.plot(fpr[i], tpr[i], color=color,  linewidth=2,
                 label='ROC curve of class {0} (area = {1:0.2f})'
                 ''.format(i, roc_auc[i]))

    plt.plot([0, 1], [0, 1], 'k--',  linewidth=2)
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
```

```
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic to multi-class '+str(p))
    p+=1
    plt.legend(loc="lower right")
    plt.show()
plt.barh(toprow, DTC1.feature_importances_)
plt.title("Attribute Importance, ")
plt.xlabel("Fractional contribution to predictive power")
plt.show()
acc = np.mean(scores)
print('Accuracy ='+ str(acc))
precisionavg = np.mean(precisions)
print('Precision ='+ str(precisionavg))
recallavg = np.mean(recalls)
print('Recall ='+ str(recallavg))
fscoreavg = np.mean(fscores)
print('F1 Score ='+ str(fscoreavg))
print('ROC AUC Score ='+ str(roc_auc["macro"]))
```

Appendix B

Random Forest Model

```
scores=[]
precisions=[]
recalls=[]
fscores=[]
p=1
RF1 = RandomForestClassifier(n_estimators=1000)
for train_index, test_index in cv.split(train1):
    X_train, X_test = train1.iloc[train_index], train1.iloc[test_index]
    y_train, y_test = train_ys1.reindex(train_index),
train_ys1.reindex(test_index).reset_index(drop=True)
    RF1.fit(X_train, y_train)
    print('Accuracy for Split '+ str(p))
    print(RF1.score(X_test, y_test))
    scores.append(RF1.score(X_test, y_test))
    pred = RF1.predict(X_test)
    print('Confusion Matric for Split '+ str(p))
    print(ConfusionMatrix(y_test, pred))
    print('Precision, Recall, F Score for Split '+str(p))
    print(precision_recall_fscore_support(y_test, pred, average='macro'))
    precision,recall,fscore,support=precision_recall_fscore_support(y_test,
pred, average='macro')
    precisions.append(precision)
    recalls.append(recall)
    fscores.append(fscore)

    predbin = label_binarize(pred,classes=[0,1,2,])
    y_testbin = label_binarize(y_test,classes=[0,1,2,])
    n_classes = predbin.shape[1]

    fpr = dict()
    tpr = dict()
    roc_auc = dict()

    for i in range(n_classes):
        fpr[i], tpr[i], _ = roc_curve(y_testbin[:, i], predbin[:, i])
        roc_auc[i] = auc(fpr[i], tpr[i])

    fpr["micro"], tpr["micro"], _ = roc_curve(y_testbin.ravel(),
predbin.ravel())
```

```
    roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

    # First aggregate all false positive rates
    all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))

    # Then interpolate all ROC curves at this points
    mean_tpr = np.zeros_like(all_fpr)
    for i in range(n_classes):
        mean_tpr += interp(all_fpr, fpr[i], tpr[i])

    # Finally average it and compute AUC
    mean_tpr /= n_classes

    fpr["macro"] = all_fpr
    tpr["macro"] = mean_tpr
    roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

    # Plot all ROC curves
    plt.figure()
    plt.plot(fpr["micro"], tpr["micro"],
             label='micro-average ROC curve (area = {0:0.2f})'
                   ''.format(roc_auc["micro"]),
             color='green', linestyle=':', linewidth=4)

    plt.plot(fpr["macro"], tpr["macro"],
             label='macro-average ROC curve (area = {0:0.2f})'
                   ''.format(roc_auc["macro"]),
             color='navy', linestyle=':', linewidth=4)

    colors = cycle(['blue', 'orange', 'red'])
    for i, color in zip(range(n_classes), colors):
        plt.plot(fpr[i], tpr[i], color=color,  linewidth=2,
                 label='ROC curve of class {0} (area = {1:0.2f})'
                 ''.format(i, roc_auc[i]))

    plt.plot([0, 1], [0, 1], 'k--', linewidth=2)
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic to multi-class '+str(p))
    p+=1
    plt.legend(loc="lower right")
    plt.show()
plt.barh(toprow, RF1.feature_importances_)
```

```
plt.title("Attribute Importance, Adaboost, ")
plt.xlabel("Fractional contribution to predictive power")
plt.show()
acc = np.mean(scores)
print('Accuracy ='+ str(acc))
precisionavg = np.mean(precisions)
print('Precision ='+ str(precisionavg))
recallavg = np.mean(recalls)
print('Recall ='+ str(recallavg))
fscoreavg = np.mean(fscores)
print('F1 Score ='+ str(fscoreavg))
print('ROC AUC Score ='+ str(roc_auc["macro"]))
```

Appendix C

AdaBoost Model

```
scores=[]
precisions=[]
recalls=[]
fscores=[]
p=1
ada1 = AdaBoostClassifier(n_estimators=100)
for train_index, test_index in cv.split(train1):
    X_train, X_test = train1.iloc[train_index], train1.iloc[test_index]
    y_train, y_test = train_ys1.reindex(train_index),
train_ys1.reindex(test_index).reset_index(drop=True)
    ada1.fit(X_train, y_train)
    print('Accuracy for Split '+ str(p))
    print(ada1.score(X_test, y_test))
    scores.append(ada1.score(X_test, y_test))
    pred = ada1.predict(X_test)
    print('Confusion Matric for Split '+ str(p))
    print(ConfusionMatrix(y_test, pred))
    print('Precision, Recall, F Score for Split '+str(p))
    print(precision_recall_fscore_support(y_test, pred, average='macro'))
    precision,recall,fscore,support=precision_recall_fscore_support(y_test,
pred, average='macro')
    precisions.append(precision)
    recalls.append(recall)
    fscores.append(fscore)

    predbin = label_binarize(pred,classes=[0,1,2,])
    y_testbin = label_binarize(y_test,classes=[0,1,2,])
    n_classes = predbin.shape[1]

    fpr = dict()
    tpr = dict()
    roc_auc = dict()

    for i in range(n_classes):
        fpr[i], tpr[i], _ = roc_curve(y_testbin[:, i], predbin[:, i])
        roc_auc[i] = auc(fpr[i], tpr[i])

    fpr["micro"], tpr["micro"], _ = roc_curve(y_testbin.ravel(),
predbin.ravel())
```

```
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

# First aggregate all false positive rates
all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))

# Then interpolate all ROC curves at this points
mean_tpr = np.zeros_like(all_fpr)
for i in range(n_classes):
    mean_tpr += interp(all_fpr, fpr[i], tpr[i])

# Finally average it and compute AUC
mean_tpr /= n_classes

fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

# Plot all ROC curves
plt.figure()
plt.plot(fpr["micro"], tpr["micro"],
         label='micro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["micro"]),
         color='green', linestyle=':', linewidth=4)

plt.plot(fpr["macro"], tpr["macro"],
         label='macro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["macro"]),
         color='navy', linestyle=':', linewidth=4)

colors = cycle(['blue', 'orange', 'red'])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, linewidth=2,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', linewidth=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic to multi-class '+str(p))
p+=1
plt.legend(loc="lower right")
plt.show()
```

```
plt.barh(toprow, ada1.feature_importances_)
plt.title("Attribute Importance, Adaboost, ")
plt.xlabel("Fractional contribution to predictive power")
plt.show()
acc = np.mean(scores)
print('Accuracy ='+ str(acc))
precisionavg = np.mean(precisions)
print('Precision ='+ str(precisionavg))
recallavg = np.mean(recalls)
print('Recall ='+ str(recallavg))
fscoreavg = np.mean(fscores)
print('F1 Score ='+ str(fscoreavg))
print('ROC AUC Score ='+ str(roc_auc["macro"]))
```

Appendix D

SVM Model

```
p=1
scores=[]
precisions=[]
recalls=[]
fscores=[]
clfsvmlinear1 = svm.SVC(C= 2.0, degree= 1, kernel= 'rbf')
for train_index, test_index in cv.split(train1):
    X_train, X_test = train1.iloc[train_index], train1.iloc[test_index]
    y_train, y_test = train_ys1.reindex(train_index),
train_ys1.reindex(test_index).reset_index(drop=True)
    clfsvmlinear1.fit(X_train, y_train)
    print('Accuracy for Split '+str(p))
    print(clfsvmlinear1.score(X_test, y_test))
    scores.append(clfsvmlinear1.score(X_test, y_test))
    pred = clfsvmlinear1.predict(X_test)
    print('Confusion Matric for Split '+str(p))
    print(ConfusionMatrix(y_test, pred))
    precision,recall,fscore,support=precision_recall_fscore_support(y_test,
pred, average='weighted')
    precisions.append(precision)
    recalls.append(recall)
    fscores.append(fscore)

    print('Precision, Recall, F Score for Split '+str(p))
    print(precision_recall_fscore_support(y_test, pred, average='weighted'))
    predbin = label_binarize(pred,classes=[0,1,2,])
    y_testbin = label_binarize(y_test,classes=[0,1,2,])
    n_classes = predbin.shape[1]

    fpr = dict()
    tpr = dict()
    roc_auc = dict()

    for i in range(n_classes):
        fpr[i], tpr[i], _ = roc_curve(y_testbin[:, i], predbin[:, i])
        roc_auc[i] = auc(fpr[i], tpr[i])

    fpr["micro"], tpr["micro"], _ = roc_curve(y_testbin.ravel(),
predbin.ravel())
```

```
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

# First aggregate all false positive rates
all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))

# Then interpolate all ROC curves at this points
mean_tpr = np.zeros_like(all_fpr)
for i in range(n_classes):
    mean_tpr += interp(all_fpr, fpr[i], tpr[i])

# Finally average it and compute AUC
mean_tpr /= n_classes

fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

# Plot all ROC curves
plt.figure()
plt.plot(fpr["micro"], tpr["micro"],
         label='micro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["micro"]),
         color='green', linestyle=':', linewidth=4)

plt.plot(fpr["macro"], tpr["macro"],
         label='macro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["macro"]),
         color='navy', linestyle=':', linewidth=4)

colors = cycle(['blue', 'orange', 'red'])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, linewidth=2,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', linewidth=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic to multi-class '+str(p))
p+=1
plt.legend(loc="lower right")
plt.show()
```

```
acc = np.mean(scores)
# print('Scores ='+ str(scores))
print('Accuracy ='+ str(acc))
precisionavg = np.mean(precisions)
print('Precision ='+ str(precisionavg))
recallavg = np.mean(recalls)
print('Recall ='+ str(recallavg))
fscoreavg = np.mean(fscores)
print('F1 Score ='+ str(fscoreavg))
print('ROC AUC Score ='+ str(roc_auc["macro"]))
```

Appendix E

AdaBoost Model for Election()

```
def Election(year):
    train = master.copy()
    train = train.loc[train['Year']!= int(year)]
    year_to_use = "result" + str(year)
    train_ys = train['result']

    test = master.copy()
    test = test.loc[test['Year']== int(year)]
    #Removing unnecessary columns from the training X data

    original_training_data = train
    original_test_data = test
    train = train.drop(columns=['id', 'PCON13CD', 'PCON13CDO',
'Constituencies', 'geometry', 'result','Year'])

    test_ys = test['result']
    test_ys=test_ys.to_numpy()

    #Removing unnecessary columns from the X test data
    test = test.drop(columns=['id', 'PCON13CD', 'PCON13CDO',
'Constituencies', 'geometry', 'result','Year'])
    # Initiating a Random Forest Classifier
    ada = AdaBoostClassifier(n_estimators=100)

    #Fitting the Classifier to the training Data
    ada.fit(train,train_ys)

    #Using the model to classify training data
    y_train_predictions = ada.predict(train)

    #Using the model to classify test data
    y_predictions = ada.predict(test)
    print(classification_report(test_ys, y_predictions, digits=5))

    precision,recall,fscore,support=precision_recall_fscore_support(test_ys,
y_predictions, average='macro')
```

```
    predbin = label_binarize(y_predictions,classes=[0,1,2,])
    y_testbin = label_binarize(test_ys,classes=[0,1,2,])
    n_classes = predbin.shape[1]

    fpr = dict()
    tpr = dict()
    roc_auc = dict()

    for i in range(n_classes):
        fpr[i], tpr[i], _ = roc_curve(y_testbin[:, i], predbin[:, i])
        roc_auc[i] = auc(fpr[i], tpr[i])

    fpr["micro"], tpr["micro"], _ = roc_curve(y_testbin.ravel(),
predbin.ravel())
    roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

    # First aggregate all false positive rates
    all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))

    # Then interpolate all ROC curves at this points
    mean_tpr = np.zeros_like(all_fpr)
    for i in range(n_classes):
        mean_tpr += interp(all_fpr, fpr[i], tpr[i])

    # Finally average it and compute AUC
    mean_tpr /= n_classes

    fpr["macro"] = all_fpr
    tpr["macro"] = mean_tpr
    roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

    # Plot all ROC curves
    plt.figure()
    plt.plot(fpr["micro"], tpr["micro"],
             label='micro-average ROC curve (area = {0:0.2f})'
                   ''.format(roc_auc["micro"]),
             color='green', linestyle=':', linewidth=4)

    plt.plot(fpr["macro"], tpr["macro"],
             label='macro-average ROC curve (area = {0:0.2f})'
                   ''.format(roc_auc["macro"]),
             color='navy', linestyle=':', linewidth=4)

    colors = cycle(['blue', 'orange', 'red'])
```

```
    for i, color in zip(range(n_classes), colors):
        plt.plot(fpr[i], tpr[i], color=color, linewidth=2,
                 label='ROC curve of class {0} (area = {1:0.2f})'
                 ''.format(i, roc_auc[i]))

    plt.plot([0, 1], [0, 1], 'k--', linewidth=2)
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic to multi-class ')
    plt.legend(loc="lower right")
    plt.show()

    print(str(year)+" Got training data accuracy of " +
str(accuracy_score(y_train_predictions,train_ys)))
    print(str(year)+" Ada Boost achieved prediction accuracy " +
str(accuracy_score(y_predictions, test_ys)))

    print(str(year)+" Ada Boost achieved precision " + str(precision))
    print(str(year)+" Ada Boost achieved recall " + str(recall))
    print(str(year)+" Ada Boost achieved fscore " + str(fscore))
    print(str(year)+" Ada Boost achieved ROC AUC Score "+
str(roc_auc["macro"]))
    print('\n')
    #Inserting the labels back into the training and test data dataframes
    original_training_data.insert(10,"RandomForest
"+year,y_train_predictions)
    original_test_data.insert(10,"RandomForest "+year,y_predictions)

    print(ConfusionMatrix(test_ys, y_predictions))
    print('\n')
    # Plotting the Random Forest Classifications and the Actual Results
    cmap = matplotlib.colors.LinearSegmentedColormap.from_list("",
['#0087DC','#FAA61A','#E4003B'])
    original_test_data.plot(column='result', figsize=(20,10), cmap=cmap)
    plt.title(str(year) + " Election Result", fontsize=18)

    original_test_data.plot(column='RandomForest '+year, figsize=(20,10),
cmap=cmap)
    plt.title(str(year) + " Election Prediction", fontsize=18)



    #Working out which constituencies have been incorrectly classified.
    missclassified = []
```

```
    resultpredictions = original_test_data['RandomForest '+year].values
    result = original_test_data['result'].values
    for i in range(original_test_data.shape[0]):
        if(resultpredictions[i] == result[i]):
            missclassified.append(1)
        else:
            missclassified.append(0)
    original_test_data.insert(0,"AccuratePrediction", missclassified)
    #Printing out which constituencies have been missclassified.

    constituencylist = original_test_data['Constituencies'].values
    partylist = original_test_data['result'].values

    print(str(year)+" Incorrectly classified constituencies were:")
    for i in range(len(constituencylist)):
        if(missclassified[i]==0):
            print(constituencylist[i])

    cmap = matplotlib.colors.LinearSegmentedColormap.from_list("",
['red','green'])
    # Plot to highlight which constituencies have been missclassified
    original_test_data.plot(column='AccuratePrediction',
figsize=(20,10),cmap=cmap)
    plt.title("Map of incorrectly classified seats, " + str(year),
fontsize=18)
    plt.show()
    original_test_data['RandomForest '+year] =
le.inverse_transform(original_test_data['RandomForest '+year])
    return original_test_data[['Constituencies','RandomForest
‘+year]].reset_index(drop=True)
```

Appendix F

Web Scraping and NLP

```
class Analysis:
    def run(self):
        headers = {
            "User-Agent":
            "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/44.0.2403.157 Safari/537.36"
            }
        #try DBScan
        response = requests.get(self.url, headers=headers)
        soup = BeautifulSoup(response.text, 'html.parser')
        headline_results1 = soup.find_all('div', class_='JheGif nDgy9d')
        headline_results = soup.find_all('div', class_='Y3v8qd')
        #open the spreadsheet we will write to
        with open('%s%s.csv' % (self.term,self.start[-4:]), 'a') as file:
            w = csv.writer(file)
            for text1,text in zip(headline_results1,headline_results):
                blob1 = TextBlob(text1.get_text())
                blob = TextBlob(text.get_text())
                w.writerow([blob1+'. '+blob,'%s' % (self.start)])
        sia = SIA()
        results = []

        for text1,text in zip(headline_results1,headline_results):
            blob1 = TextBlob(text1.get_text())
            blob = TextBlob(text.get_text())
            blob_all = blob1+'. '+blob
            pol_score = sia.polarity_scores(str(blob_all))
            pol_score['headline'] = str(blob_all)
            results.append(pol_score)

              #write header row to spreadsheet
        dfr = pd.DataFrame.from_records(results)
        self.sentiment = dfr["compound"].mean()
        print('it is '+str(self.sentiment))

    def __init__(self, term,start,end):
        self.term = term
        self.start = start
        self.end = end
```

```
        self.sentiment = 0
        self.subjectivity = 0
        self.url = 'https://www.google.com/search?
tbs=cdr%3A1%2Ccd_min%{1}%2Ccd_max%{2}&q={0}
&source=lnms&tbm=nws'.format(self.term,self.start,self.end)
```