Irish Election Forecasting

Eamon McNicholas

B.Sc.(Hons) in Software Development

April 15, 2017

Final Year Project

Advised by: Dr Ian Mcloughlin

Department of Computer Science and Applied Physics
Galway-Mayo Institute of Technology (GMIT)

Contents

About this project

Abstract A brief description of what the project is, in about two-hundred and fifty words.

Authors Explain here who the authors are.

Chapter 1

Introduction

Since the advent of elections for choosing who represents us on a local, national or international scale, the forecasting of these elections has become of interest to a signicant portion of the voting population. Where predictions are concerned, there are also huge monetary sums at stake for polling firms, news companies and gambling markets alike. For the majority of people, polling of the electorate is the most well known of the election forecasting techniques commonly used. Generally, the average of the latest polls is the most extensively accepted best indicator of the probable results of an election [1].

However, in two of the biggest voting events of the last year, the polls from major news firms and polling organisations were way off target [2]. In the UK, the referendum for choosing to stay or leave the European Union result went against the outcome predicted by the majority of major UK pollsters [3]. In what was probably the most popular election in the world, the 2016 US presidential election surprised most when Donald Trump bet Hilary Clinton. A renowned statistical analyst, Nate Silver, created Fivethirtyeight.com, one of the most popular event prediction websites in the world. He gave Trump more of a chance of winning (35 percent) than anyone else, but his prediction was still wide off of the mark based on the actual results [4].

Other options for forecasting comes in the form of creating various models of voting behaviour and regression models of previous elections in order to predict the results of future elections. The voting behaviour models take into account predictors (i.e. phenomeana that can vary at any time) such as promised changes in tax rates or the creation of jobs in a area [1]. This type of model can be useful for local elections[1]. Regression models such as a Bayesian forecasting model that take into account polling data as well as data from previous elections are also becoming a popular means for election forecasting, especially with the recent rise of more easily accessible data on

the internet [5].

In this paper I will describe in detail my process for creating an election forecasting application, the challanges that I encountered as well as the research I conducted on the various areas of election forecasting. I will then discuss how I translated this theory into practice by designing and implementing the ideas acquired from my research. To conclude, I will discuss the outcomes of my project, highlight the lessons learned along with my proposed reccomendations for further research / applications.

Chapter 2

Context

Scope and Context

The purpose of this project is to develop an application that can be used for predicting the results of a general election in Ireland. Someone with a background in political science or data analytics could use this application with a model they have developed for election predictions. A database will be created which consists of the population in the Republic of Ireland who are eligble to register to vote or who are already registered to vote. By including the section of the population who are eligible to register to vote, but who have not actually done so, this allows for dramatic changes in the electorate to be accounted for. The database population will be categorized according to a number of relevant demographic characteristics in this context, including social class, age-range, gender and martial status. Furthermore, the subsection of the population who live in counties with more than one constituency will also be taken into consideration as another relevant characteristic.

The application will have a front end that consists of a web page with two interactive maps of the republic of Ireland. The first map will have display the predicted results and the second map will display the actual results. The prediction model being using in this application will be my own interpretation of how this software can be used. It will be partially based on the model used by an Irish political analyst, Dr. Adrian Kavanagh, who specializes in this area [6].It will be the result of calculating the general probabilty of the results for each constituency by combining polling data and historical averages and then using the d'hondt method to assign seats to each party [6].

Objectives

My main objective in this project was to create a full stack web application for election forecasting. This then generated the following secondary objectives:

- To create a reasonably accurate database of the republic of Ireland's population who are elegible to vote as well as including the population who are eligble to register to vote.
- To create a model for predicting the general election results of a previous election so that it would be possible to compare the predicted and actual results. This would help create a benchmarch against which the accuracy of my predictor model could then be compared.

However, as I became more familiar with the research regarding election forecasting, it became apparent that the process of creating a truely comprehensive statistical model would be beyond the scope of this project. Subseqently, I then narrowed the focus of my secondary objectives to illustrating a basic example of how this type of model could be designed and implemented in practice.

My final original objective had been to create an interactive map of Ireland which would allow the user tohover over each constituency to compare the actual results with those predicted. However, I found that it was much less complicated in practice to design two maps of Ireland where the counties were interactive, and the actual or the predicted results would be displayed depending on which map was clicked.

Links and Section Preview

- https://github.com/DevEMCN/Irish-Election-Forecasting—FYP
- https://github.com/DevEMCN/final-year-project-template

Methodology

This section will cover the way I approached and planned out how I was going to design this project, along with the methods used for data extraction.

Technology Review

This section will review the technologies that were used for the development process. It will illustrated how the best fit development environment for

the project was researched, and how these tools were used to implement the project.

System Design

In this chapter, I will provide a comprehensive commentary on the system archictecture including suitable diagrams and screenshots.

System Evaluation

In this section, I will analyse my system design in terms of robustness and performance. I will also highlight any limitations of my design which became clearer to me towards the end of the project.

Chapter 3

Methodology

The following section will describe the methodologies and development approach used to design this project. It is divided into two main parts: 1) Research and 2) Development approach. The research section will also be further subdivided into sections discussing 1) Sofware and 2) Data. I will provide an account of the relevant research to illustrate how the technologies I choose were some of the most suitable for meeting my projects needs. I will also explore the research concerning prediction analysis. Finally, the data section will detail how I sourced the relevant data required for analysis, i.e. including polling, historical and population data.

Research

Software

While researching which method would superior for the purpose of my data analysis, I found a contentious debate between the benefits of R and Python online [7]. This method was to be the foundation upon which the rest of the project would be structured. I therefore knew that it was important to research which one would be the most suitable match for achieving the objectives of my project. I eventually choose Python for the following reasons.

Firstly, I wanted the project to be a web application and Python seemed to be the best language to faciliate this aim. Python allowed me the option of using Python Django web framework or Python Flask framework. This allowed me to structure the entire application in a neat and accessible format for sharing with others interested. Furthermore, a python web application setup would have allowed me to easily implement a REST API for the front end, had I had the additional time and resouces to add this in. My final rea-

son for choosing Python is that I had wanted to learn this language anyway and had not had a prior opportunity to do so before. Just like Java or C, Python is one of the most popular and multipurpose languages commonly used in the IT industry at present. On the otherhand, R is a highly specialized language for data analysis and is therefore not as flexible for other uses. Lastly, from researching Python's data analysis potential, this language seems to have improved exponentially in recent years. This improvement is manifest in the number of libraries available for data analysis, to the point that it is now considered as to be nearly on par as the language R.

Next, I had to choose a database and there were many choices in this context. I had anticpated that I would be using huge amounts of data to build a database for the electorate as this would involve millions of data entries (i.e. including the section of the population who are currently unregistered but still available to register to vote). Neo4j seemed like the best solution to help me work through this challange. It allows for the importing of massive amounts of data into a database at very high speeds. Considering there was to be at least 3 million people added, all with different characteristics (see data section 2.1.2), Neo4j was the most appropriate choice for this purpose. As Neo4j is a graph database, it made the task of illustrating any likely and predicted relationships between various datasets easy to display [8]. As with Python, I had no experience with the use of graph databases or with Neo4j in particular. Therefore, this also influenced my decision making in choosing this database as I felt it would help add more variety to my overall portfolio of programming skills.

To connect Neo4j to the one of the main Python webframeworks, I researched on the neo4j website what options were available [9]. I discovered a python library called py2neo and found an example of its use with the Python Flask web framework [10]. This example was the ideal candidate for use as a starting point for my project. I also looked at other examples of similar applications created by Nicole Whyte, who maintains some excellent Juypter Notebook exemplars using py2neo [11]. I therefore concluded that this was the most suitable option and moved on to researching the front end section of my application.

For the front end, I wanted to visualize the predictions and the results on a interactive map of Ireland. The natural choice here was the very powerful Javascript library d3.JS. This is because while Python may have science libraries that can visualize data, d3 has excellent built-in world map visualisations that make it particularly ideal for the purposes of this application [12].

Data

I based what datasets I was going to use in this application on the work of Nate Silver, a statistician, created of fivethirtyeight.com [13]. Nate Silver is famous for his predictions of the 2008 and 2012 US presidential elections. In 2008, Silver correctly predicted which president would win the electoral college votes in 49 of the 50 states. His models primarily focus on the use of social demographics through polling and examining historical data with the use of regression analysis [14].

Population For creating the database of the population of Ireland (electorate and possible future electorate members), I needed to acquire the relevat statistics from the Central Statistics Office (CSO) website. The latest complete census data available for use during the completion of this project was from 2011 [15].

Polling Data The polling data comes from the Irish Times website. The polling is conducted by the Irish Times Newspaper as well as by Ipsos MRBI, an international market research company [16]. I used the closest available polling data to the election held on the 26th of February 2016, which was data the 4th of February 2016.

Historical I sourced the historical results of the five elections previous to the 2016 election (2011, 2007, 2002, 1997 and 1992) from electionsireland.org [17]. The creator of this website maintains accurate results and data from the Republic of Ireland elections dating back as far as the very first election in 1918.

Prediction Analysis Research

Initially, I focused on the work of Nate Silver, creator of fivethirtyeight.com. In 2008, Nate Silver became a famous name in the area of election forecasting for correctly predicting 49 out of 50 US states in terms of electorate college votes for the 2008 US presedential election [18]. He was again successful in the 2012 presendential election, when he predicted that Obama would win again [14]. I discovered an article by a statistical analyist, Bob O Hara, who provided a more basic explanition regarding how he thought Nate Silver was so accurate in his forecasting [14]. This lead me to begin my research into prediction analysis, discovering such models as regression analysis and bayesian inference. These models are both quite complicated and requrie a strong theoretical foundation in the use of statistics. Unfortunately, applying

them would have been beyond the timeframe with which I had to complete this project. Therefore, in favour of staying on track to complete a prototype of this project before the deadline, I explored alternative prediction possibilities.

While researching election forecasting methodologies previously explored in Ireland, I came across the work of politcal analyst, Dr Adrian Kavanagh, who is also a lecturer in NUI Maynooth. To forecast each constituency in the Republic of Ireland, he used polling data and the d'hondt method [6].The d'hondt method, otherwise known as the jefferson method, is a highest averages method that is used for allocating seats to parties in a political system that uses proportional representation as its basis [19]. As this appeared to serve the same purpose as the more complicated statistical models which I had first happened across, I decided to adapt it for use for the current project.

Development Approach

With a project of this nature, it seemed difficult to try to plan ahead even by one or two steps at a time in the beginning. There was no guide, example, or tutorial to follow and so I spent a lot of time thinking about the best route to pursue next with my project after I each completed step and achieved another milestone. This resulted in many trial and error attempts from ideas I thought of by myself at various intervals during the process. I developed the system by using an incremental approach, i.e. by starting with the database set-up and slowly figuring out where I was going with the application next. However, over half way through, I realized that I had figured out exactly what I wanted to do and how I was going to do it. Building on this confidence, I therefore continued to use an incremental approach for developing all remaining module.

Chapter 4

Technology Review

In this section I will describe in detail each technology used in the process of creating the current project. The whole application is centered around using a Python Flask framework. Therefore, I will begin this section with an overview of the Python modules used, including its development environment. Following, I will describe the database technology (Neo4j). Finally, I will provide an account of the technologies used for building the front-end, i.e. d3.JS, jQuery and Twitter Bootstrap.

Python

PyCharm

Pycharm is an intergrated development environment (IDE) for the Python language. It was created by JetBrains and released in 2010 [18]. It provides code assistance and code completion as well as linter integration (i.e. for checking mistakes in the Python code). It also allows for syntax and error highlighting. Furthermore, it offers built-in development tools. These tools include a debugger which comes with a GUI, a choice of version control systems which are pre-integrated (e.g. Git) and it also includes direct database access (e.g. to major databases such as SQL Server and Oracle). Furthermore it provides support for the most commonly used Python web frameworks, i.e. Django, Web2py and Flask. It is also compatible for use with Javascript (and its various flavours, e.g. React.js). Finally, it has a live-editor which facilitates for the real-time simultaneous editing and viewing of code changes as they occur on a webpage. Among other features, Pycharm has also integrated Scientfic tools such as Ipython and supports scientific packages such as NumPy [20].

Virtualenv

Virtualenv is a tool used for the creation of isolated Python environments. It resolves any situation where potential dependencies or versions may produce conflict within your developmental environment. Some examples of where this tool would be useful include:

- If one application you are using requires version 1 of a package or library and another application you are working on requires the 2nd version of the same package/library. By using virtualenv, you can solve this problem by creating two seperate Python environments for each application and each has will have its own installation directories.
- If there is situation whereby you want to keep an application as it is and you don't want any of the libraries that it depends on to be upgraded to another version. Again, virtualenv would be ideal for resolving this dilemma. By maintaining its own installation directories seperate from the global installation folders you can allow the library versions to remain the same as they will not be updated when you update the global environment.

Virtualenv is also very simple to use, with a primary focus on the following the three commands:

- virtualenv ENV – where ENV is the name of the directory you want to position the isolated environment.
- Source bin/activate – changes your environment to the recently created isolated environment. It also adjusts your shell promt to highlight this change or environment.
- Deactivate – changes your path to the default path on your system [21].

Python 2.7

Python is an interpreted, object-oriented language that is similar in its functionality to languages such as Java, Scheme and Ruby. As it is an interpreted language, it does not need to be compiled before running and this allows it to be used in an interactive shell, so that it is possible to test sniplets of code instantly. It is often considered to be a preferred language for prototype development and for any type of throw-away program development that requires quick building. It has object-oriented features such as classes and multiple inheritence. Its syntax was developed to be clear and easy to

read. It is a high-level language as it has data-types such as arrays and dictionaries already built-in. For the grouping of statements like loops and conditionals, it uses indentation (unlike Java, which uses brackets). Python is both strongly and dynamically typed, in that changes to a variable require explicit conversion but the variables do not have a type. An example of how this looks like in practice is displayed as follows [22]:

```
bob = 1 # bob is an integer
bob = "bob" # bob is now a string
```

Python currently has two versions: 2.x and 3.x.. Python 3.0 came out in 2008 and the last updated version of Python released in 2010 was 2.7 of 2.x. This project uses Python 2.7 because it was the default version used in the virtualenv environment of the application [23].

Flask

Python Flask is a micro-web framework and is based on two other Python packages:

- The Werkzeug toolkit – a toolkit that include multiple utilities for WSGI (Web Server Gate Interface, which is an interface between web servers and web applications/frameworks for Python [24])
- Jinja 2 – a templating language for Python based on Django's templating model [25].

The micro part of Flask refers to the fact that the core of the framework is simple but yet to open to extension at the same time. An example of this flexibility is illustrated by the fact that you can choose which database you would prefer to interact with Flask and this decision is not automatically dictated for you by the framework. Another benefit is that while the templating engine is based on Jinja 2, this can also be easily changed and adapted for different purposes as required. Flask is essentially a barebones framework that can be used in conjunction with the many extensions which the Python community have created specifically for it. Finally, It is very easy to get started with Flask. I have provided an example of the simplicity of its use below (call this file run.py) [26]:

```
from flask import Flask, render_template

app = Flask(__name__)
```

```
@app.route('/')
def index():
        return render_template('index.html')

@app.route('/altview')
def option2():
        return render_template('altview.html')
```

and to run that file:

```
$ pip install Flask
$ python run.py
* Running on http://localhost:5000/
```

Py2neo

Py2neo is a toolkit for connecting Python to Neo4j both through Python applications and the command line. Py2neo does not depend on any external libraries or packages and it is simple and intuitve to use in a project. The following section requires knowledge of the Neo4j database and terminology, both of which will be covered later in this chapter. When working with Py2neo, there are three main classes which are used for the majority of the time: Graph, Node and Relationship.

```
from py2neo import Graph, Node, Relationship
```

An example for using Node and Relationship looks like:

```
>>> from py2neo import Node, Relationship
>>> a = Node("Person", name="Alice")
>>> b = Node("Person", name="Bob")
>>> ab = Relationship(a, "KNOWS", b)
>>> ab
(alice)-[:KNOWS]->(bob)
```

To connect to the Neo4j database, the graph class is then used as follows:

```
>>> graph_3 = Graph("http://localhost:7474/db/data/")
```

Here is an example of querying the database taken from my own project code:

```
count_query = "MATCH (n:Widowed:Male) where " + age_labels[a] +
" and " + social_labels[s] + " and n.location in [" + counties + "]
 return count(n)"

count = graph.run(count_query).evaluate()
```

The function run of the class Graph takes a query and calls the database with that query. The 'Evaluate' function then evaluates what is returned and stores it in the count variable [27].

Pandas

Pandas is a Python library that has data structures which are fast, flexible and expressive and which have been designed for use with 'labeled' or 'relational' data. The creators of Pandas are working towards making it the go-to tool for data analysis and manipulation. Some examples of the data that Pandas is suited for include:

- Tabular Data like SQL tables or Excel Spreadsheets/CSV files with diversly-typed columns
- Time-series data that does not have to be fixed frequency and can be used for order and unordered data
- Matrix data that has row and column labels which can be either similar or diversely typed
- Most other forms of statistical data. Data can also be unlabelled and be used in Pandas data structures

For most use cases, only the data structures 'Series' (which is 1 dimensional) and 'DataFrame' (which is 2 dimensional) are used in areas ranging from statistics to engineering data analyses. As Pandas is built on top of the scientific computing package, 'NumPy', it can be used as part of scientific development environment.

The benefits of using Pandas include:

- In the cases where data is missing it handles both floating and non-floating point data
- The group-by functionality makes pandas a powerful tool for applying split/combine operations to data.

- Other Python or NumPy data structures can be easily converted into Pandas Data strutures that will transform the data into a more orderly form
- Data sets can be easily merged, joined or reshaped
- Data is can be aligned to a given set of labels provided by its user. Alternatively, Pandas also offers the option to perform this alignment by itself automatically [28].

Scrapy

Neo4j

```
MATCH(n:Person)-[:ACTS_IN]->(n:Movie) where n.born = 1956 RETURN n
```

D3.JS

```
var paragraphs = document.getElementsByTagName("p");
for (var i = 0; i < paragraphs.length; i++) {
var paragraph = paragraphs.item(i);
paragraph.style.setProperty("color", "white", null);
}

d3.selectAll("p").style("color", "white");

d3.selectAll("p")
.data([4, 8, 15, 16, 23, 42])
.style("font-size", function(d) { return d + "px"; });
```

Jquery 1.12.x

```
$(\button.continue").html(\Next stop");
```

Bootstrap

Chapter 5

System Design

System Architecture

Data and Data Analysis

Historical Data

```
import scrapy

# this spider extracts the name and party of each person who ran
#for election in the constitunecy of every election back to 1992
# note that the names are never used in this application, but
# were downloaded at the time just in case
class ConDataSpider(scrapy.Spider):
name = "con_data"
start_urls = ['http://electionsireland.org/result.cfm?election=2016&cons=32']
#['http://electionsireland.org/result.cfm?election=1992&cons=32']
#['http://electionsireland.org/result.cfm?election=1997&cons=32']
#['http://electionsireland.org/result.cfm?election=2002&cons=32']
#['http://electionsireland.org/result.cfm?election=2007&cons=32']
#['http://electionsireland.org/result.cfm?election=2011&cons=32']


def parse(self, response):
# select the exact table the required data is in
for row in response.css('body table:nth-child(5)'):
```

```
yield {
# get the name of the candidate
'name': row.css('tr td:nth-child(2) a::text').extract(),
# get the party of the candidate
'party': row.css('tr td:nth-child(4) a::attr(href)').extract(),
}

next_page = response.css('table.rhtable table:nth-child(1) a::attr(href)')
.extract()[1]
#print(next_page)
if next_page is not None: # keep going until no pages left to traverse
next_page = response.urljoin(next_page)
yield scrapy.Request(next_page, callback=self.parse)
```

Population Data

Data Analysis

Population

General Population

```
# put the connaught males csv into a pandas dataframe
df = pd.read_csv('app/data/population/population_data/connaughtmales.csv',
 header=None, delimiter='\t', converters={"4": int})
# append to the temp list group_list 1 - the number of people
# that fit the criteria passed in to the dataframe
for county in self.connaught:
group_list_1 = []
group_list_1.append(df.loc[(df[1] == county) & (df[2] == 'Single')
 & (df[3] == '20 - 24 years'), 4].sum())
group_list_1.append(df.loc[(df[1] == county) & (df[2] == 'Single') &
((df[3] == '25 - 29 years') | (df[3] == '30 - 34 years')), 4].sum())
group_list_1.append(df.loc[(df[1] == county) & (df[2] == 'Single') &
((df[3] == '35 - 39 years') | (df[3] == '40 - 44 years') | (
df[3] == '45 - 49 years')), 4].sum())
group_list_1.append(df.loc[(df[1] == county) & (df[2] == 'Single') &
((df[3] == '50 - 54 years') | (df[3] == '55 - 59 years') | (
df[3] == '60 - 64 years')), 4].sum())
group_list_1.append(df.loc[(df[1] == county) & (df[2] == 'Single') &
((df[3] == '65 - 69 years') | (df[3] == '70 - 74 years') | (
```

```
df[3] == '75 - 79 years') | (df[3] == '80 - 85 years') |
(df[3] == '85 years and over')), 4].sum())

# this python script is used to create CSV files that are to be imported into
# the Neo4j database usingNeo4js import functionality

connaught = ConnaughtPopulation() # create an instance of the province
# switch for males when running the script each time
#connaught.populate_female_lists()
connaught.populate_male_lists()
#cf = connaught.get_female_lists() # switch for males
 # call the get male lists to get the data from the connaught class
cm = connaught.get_male_lists()

counties = ['Galway', 'Leitrim', 'Mayo', 'Roscommon', 'Sligo']
status = ['Single', 'Married', 'Divorced', 'Widowed'] # used in the loops

with open('app/data/province_populations_data/connaught_population_male.csv',
 'w') as csvfile: # open the relevant empty csv
 # list of the headers for the csv file
fieldnames = ['id', 'gender','location', 'martial_status']
writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

writer.writeheader() # write the headers to the csv
count = 0 # used for creating a unique id each person to be added
for category_index, category in enumerate(cm): # switch for males
for county_index,county in enumerate(category):
for age_index, age_group in enumerate(county):
for i in range(age_group):
if age_index == 0:
writer.writerow({'id': 'cm1824' + str(count),
'gender': 'male',
'location': counties[county_index],
'martial_status': status[category_index]})
count += 1
if age_index == 1:
writer.writerow({'id': 'cm2534' + str(count),
'gender': 'male',
'location': counties[county_index],
'martial_status': status[category_index]})
count += 1
```

```
if age_index == 2:
writer.writerow({'id': 'cm3549' + str(count),
'gender': 'male',
'location': counties[county_index],
'martial_status': status[category_index]})
count += 1
if age_index == 3:
writer.writerow({'id': 'cm5064' + str(count),
'gender': 'male',
'location': counties[county_index],
'martial_status': status[category_index]})
count += 1
if age_index == 4:
writer.writerow({'id': 'cm65plus' + str(count),
'gender': 'male',
'location': counties[county_index],
'martial_status': status[category_index]})
count += 1        content...
```

Social Classes

Historical

```
import pandas as pd
from collections import Counter


# this calculates the average % of each party in each constituency in 2007
class Hist2007():
def __init__(self):
# scraped results from electionireland.org
self.results_2007_df = pd.read_json("results2007.json")
# names of each constitunecy
self.constituencies_2007_df = pd.read_json("cons_2011_2007.json")
# get the parties from the results
self.results_2007_unfiltered = list(self.results_2007_df['party'])
self.constituencies_2007_uf = list(self.constituencies_2007_df['county'])  # uf
self.constituencies_2007 = []
self.results_2007 = []
# seats in constituncies
self.number_of_seats = [5,5,4,4,4,3,5,3,3,3,4,4,4,3,3,3,5,5,4,
```

```
4,3,5,4,5,3,3,4,3,5,5,3,4,4,5,3,3,3,3,3,3,4,5,5]
self.historical_percentages = []

# this function gets the parties of the people elected in each constituency
def filter_results_2007(self):
for index, const in enumerate(self.results_2007_unfiltered):
group_list = []
# depending on the number of seats in the constituency
for seat in range(self.number_of_seats[index]):
group_list.append(const[seat])
self.results_2007.append(group_list)

def flatten_df(self):
for sublist in self.constituencies_2007_uf:
for val in sublist:
self.constituencies_2007.append(val)

def set_percentages(self):
self.filter_results_2007()
# main parties - FF, FG, SF, LB and IND. Note that anything other than these
# 5 will be added to IND the following variables holds a list of
# lists of the percentages of each party in each constituency
FF = []
FG = []
SF = []
LB = []
IND = []
# GP,Ind, PD, IFG, CS, WP, SP, SWP. IFF, IHA, NP
#self.number_of_seats = []
for index,constituency in enumerate(self.results_2007):
seats = self.number_of_seats[index]
# Counter counts all occurences of what it is given as a params
counts = Counter(constituency)
FF.append((counts['FF'] / seats) * 100)
FG.append((counts['FG'] / seats) * 100)
SF.append((counts['SF'] / seats) * 100)
LB.append((counts['LB'] / seats) * 100)
IND.append(((counts['Ind'] + counts['IND'] + counts['IFG']
+ counts['GP'] + counts['PD'] + counts['CC']
+ counts['CS'] + counts['WP']
+ counts['SWP'] + counts['IFF'] + counts['IHA'] + counts['NP']) / seats) * 100)
```

```
self.historical_percentages.append(FF)
self.historical_percentages.append(FG)
self.historical_percentages.append(SF)
self.historical_percentages.append(LB)
self.historical_percentages.append(IND)

def set_data(self):
#his2011 = Hist2007()
#his2011.set_data()
#self.number_of_seats = his2011.get_number_of_seats()
self.flatten_df()
self.set_percentages()

def get_data(self):
return self.historical_percentages # return the percentage of wins for each par
```

Database (Neo4j)

Overview

Database creation

Python w/Flask

Overview

Prediction

Front-End

Chapter 6

System Evaluation

As many pages as needed.

- Prove that your software is robust. How? Testing etc.
- Use performance benchmarks (space and time) if algorithmic.
- Measure the outcomes / outputs of your system / software against the objectives from the Introduction.
- Highlight any limitations or opportuni-ties in your approach or technologies used.

Chapter 7

Conclusion

About three pages.

- Briefly summarise your context and ob-jectives (a few lines).
- Highlight your findings from the evalua-tion section / chapter and any opportuni-ties identified.

Bibliography

[1] M. Stegmaier and H. Norpoth, "Election forecasting," *Oxford Bibliographies Online Datasets.*

[2] O. Williams-Grut, "What you would have won betting on leicester winning the league, brexit, and trump becoming president." http://uk.businessinsider.com/odds-leicester-brexit-trump-betting-winning-ladbrokes-2016-11.

[3] P. Barnes, "Eu referendum: Did the polls all get it wrong again? - bbc news." http://www.bbc.com/news/uk-politics-eu-referendum-36648769.

[4] N. Silver, "Why fivethirtyeight gave trump a better chance than almost anyone else." http://fivethirtyeight.com/features/why-fivethirtyeight-gave-trump-a-better-chance-than-almost-anyone-else/.

[5] S. E. Rigdon, S. H. Jacobson, W. K. T. Cho, E. C. Sewell, and C. J. Rigdon, "A bayesian prediction model for the u.s. presidential election," *American Politics Research,*.

[6] D. A. Kavanagh, " irish elections: Geography, facts and analyses." https://adriankavanaghelections.org/about/.

[7] M. Theuwissen, "R vs python for data science: The winner is." http://www.kdnuggets.com/2015/05/r-vs-python-data-science.html.

[8] "Top 10 reasons for choosing neo4j for your graph database." https://neo4j.com/top-ten-reasons/.

[9] "Using neo4j from python - neo4j graph database." https://neo4j.com/developer/python/.

[10] "Flask and neo4j." http://nicolewhite.github.io/neo4j-flask/pages/required-packages.html.

[11] "neo4j-jupyter." https://github.com/nicolewhite/neo4j-jupyter.

[12] "World map." https://bl.ocks.org/mbostock/4180634.

[13] N. Silver, "Frequently asked questions, last revised 8/7/08." https://fivethirtyeight.com/features/frequently-asked-questions-last-revised/.

[14] B. O'Hara, "How did nate silver predict the us election?." https://www.theguardian.com/science/grrlscientist/2012/nov/08/nate-sliver-predict-us-election.

[15] "Statbank - data and statistics." http://www.cso.ie/px/pxeirestat/statire/SelectTable/Omrade0.asp?Planguage=0.

[16] "Irish times/ipsos mrbi poll." http://www.irishtimes.com/news/politics/poll/february-4th.

[17] "Electionsireland.org: Home page." http://electionsireland.org/.

[18] N. Silver, "Fivethirtyeight's 2012 forecast." https://fivethirtyeight.blogs.nytimes.com/fivethirtyeights-2012-forecast/.

[19] M. Gallagher, "Proportionality, disproportionality and electoral systems," *Electoral Studies*.

[20] "Pycharm :: Features." https://www.jetbrains.com/pycharm/features/.

[21] "Virtualenv — virtualenv 15.1.0 documentation." https://virtualenv.pypa.io/en/stable/.

[22] "Is python strongly typed?." http://stackoverflow.com/questions/11328920/is-python-strongly-typed.

[23] "Whetting your appetite — python 2.7.13 documentation." https://docs.python.org/2/tutorial/appetite.html.

[24] "Welcome — werkzeug (the python wsgi utility library)." http://werkzeug.pocoo.org/.

[25] "Welcome to jinja2 — jinja2 documentation (2.9)." http://jinja.pocoo.org/docs/2.9/l.

[26] "Welcome to flask — flask documentation (0.12)." http://flask.pocoo.org/docs/0.12/.

[27] "The py2neo v3 handbook — the py2neo v3 handbook." `http://py2neo.org/v3/`.

[28] "pandas: powerful python data analysis toolkit — pandas 0.19.2 documentation." `http://pandas.pydata.org/pandas-docs/stable/`.