Progetto di Ingegneria informatica

Simulatore di sistemi elettorali

Laura Amabili
Anno Accademico 2020/2021

Introduzione

Obiettivo

L'obiettivo del progetto assegnatomi è realizzare un simulatore di sistemi elettorali.
Partendo da quello sviluppato da un altro studente in passato, mi sono incaricata di ampliare il campo di leggi elettorali simulate con lo scopo di rendere più completo il suo lavoro.

Per poter implementare nuove features su una base di codice progettata da terzi, è stato necessario assecondare lo schema progettuale già esistente al fine di proseguire con uno sviluppo coerente nella forma e nei contenuti.
Tra i metodi ereditati dal progetto iniziale vi sono quelli relativi alla creazione della struttura della classe, l'istanziazione della classe con i dati e l'esecuzione.

La legge implementata

Ho deciso di implementare la Legge Calderoli - nota anche come Porcellum - che venne utilizzata per le elezioni politiche italiane nel 2006, 2008 e 2013.

La scelta nasce dall'esempio stimolante che il Porcellum rappresenta rispetto alla legge elettorale già portata come modello. Inoltre, il poter reperire i dati direttamente dall'archivio del Ministero rappresenta certamente un valore aggiunto poiché consente di avere un campione corretto su cui basare la simulazione.

La Legge Calderoli – rispetto alle modalità di elezione delle Europee - è caratterizzata da:
- Liste bloccate (l'elettore non può esprimere una preferenza nei confronti di un candidato)
- Coalizioni di liste (raggruppamento di partiti con il fine di superare soglie di sbarramento)
- Premio di maggioranza (l'incremento di seggi spettante ad una lista che ha ottenuto uno specifico risultato elettorale)

Architettura

Il codice è strutturato in modo tale da avere due principali tipologie di classi: entità politiche ed entità geografiche.
Le prime sono utilizzate per la gestione di partiti, delle coalizioni e dei candidati e determinano chi verrà eletto.
Le seconde hanno lo scopo di gestire e manipolare i voti per assegnare i seggi.

Inoltre la struttura include un Hub globale per potersi riferire alle istanze delle classi e per la consultazione delle relazioni tra le esse.

A causa della poca duttilità delle classi preconfigurate è risultato essere preferibile l'utilizzo di un approccio a metaclassi. Le metaclassi permettono con facilità l'aggiunta di attributi e metodi alle classi da esse istanziate. Nella struttura sono presenti varie tipologie di metaclassi, ciascuna con uno scopo preciso e indipendente. La loro combinazione produce una scalabilità tale da assecondare al meglio le esigenze del progetto specifico.

Realizzazione

Sviluppo

L'implementazione della Legge Calderoli è stata realizzata tramite quattro fasi:

1. **Studio della struttura precedentemente sviluppata.**

Studio approfondito delle funzioni già presenti nel codice in modo tale che le parti aggiunte potessero agire in sinergia ed essere così ottimizzate al massimo.

2. **Comprensione della Legge Elettorale da implementare.**

Una volta stabilito il modus operandi con cui era possibile integrare al meglio i due progetti, ho ritenuto necessario documentarmi sull'aspetto legislativo della leggi Calderoli in modo tale da simulare un'elezione nel modo più vicino possibile alla realtà.

Al termine della fase di studio e documentazione è risultata chiara
la necessità di utilizzare classi dedicate per la gestione delle coalizioni e dei partiti e la presenza di classi differenti per le tre zone geografiche principali: l'area nazionale, la Valle D'Aosta - che viene considerata a parte - e l'Estero.
Le entità geografiche della Nazione e dell'Estero sono poi state divise in circoscrizioni.

Nella prima è stato necessario far sì che la Nazione stessa creasse una prima distribuzione dei seggi tra i partiti per poter essere poi utilizzata per la successiva correzione della distribuzione circoscrizionale dei seggi ai partiti.

L'Estero opera invece occupandosi della propria distribuzione dei seggi divisa per circoscrizioni: ogni circoscrizione, dopo aver generato la sua distribuzione dei seggi, la passa al livello soprastante che - una volta raccolte tutte le distribuzioni circoscrizionali - unisce i seggi rispetto ai partiti e trova la distribuzione generale estera.

La Valle D'Aosta è stata invece gestita separatamente rispetto alla Nazione principale poiché, essendo caratterizzata da un collegio uninominale (a differenza del collegio plurinominale delle altre regioni), il seggio è assegnato in maniera puramente proporzionale senza la necessità di correzioni e altre elaborazioni.

3. **Strutturare il progetto.**

La fase di progettazione è seguita da quella di realizzazione, che comprende delle scelte strutturali significative:

- La decisione di gestire le tre aree geografiche come tre lanes separate che operano in parallelo. Ciò ha permesso di dividere il lavoro in tre sezioni distinte che non comunicano tra loro e non si scambiano dati.
- La decisione di raggruppare tutti i partiti, non facenti parte di una coalizione di liste, dentro una coalizione chiamata 'NO COALIZIONE', che viene gestita in modo speciale dai filtri dentro la classe coalizione. Tale scelta implementativa ha permesso di non dover combinare più sorgenti di dati dentro ai metodi, permettendomi così di prelevare tutte le informazioni cercate direttamente dalla stessa fonte.

Le funzionalità implementate sono quelle atte ad elaborare i voti secondo i processi previsti dalla Legge Calderoli:
- Gestione del premio di maggioranza
- Suddivisione dei seggi di una coalizione tra i suoi partiti a livello nazionale
-Creazione di una distribuzione, alle coalizioni e singole liste, dei seggi tra le varie circoscrizioni
- Suddivisione dei seggi di una coalizione tra i suoi partiti a livello circoscrizionale
- Correzione delle distribuzioni generate ad ogni livello
Inoltre, ho deciso di realizzare sia le procedure dedite alla gestione delle coalizioni sia quelle specifiche per la gestione delle soglie di sbarramento nel modo definito dalla Legge Calderoli.
Queste funzionalità sono implementate nel file dedicato nella directory src/Commons/porcellum.py.

4. **Acquisizione dati e sviluppo.**

Una volta realizzata la struttura del progetto ho proceduto a selezionare i dati più significativi per il testing delle funzionalità per poter poi iniziare lo sviluppo del codice secondo quanto precedentemente progettato.

Dati

I dati sono disponibili sul sito dell'archivio azionale del ministero ho selezionato quelli delle elezioni della Camera dei Deputati per l'anno 2013.

Dopo aver acquisito i dati è stata necessaria una pulizia con lo scopo di renderli compatibili e adeguati alla mia implementazione.

Nello specifico le limitazioni riscontrate non mi permettevano di accedere alla prima colonna dei dati poichè usata per indicizzazione.
Nei miei dati questa specifica colonna era usata per l'identificazione della circoscrizione da cui provenivano i voti, informazione per me basilare e che ho quindi ho ritenuto necessario duplicare aggiungendo una seconda colonna contenente la regione di derivazione.

L'elaborazione finale dei dati prevede la suddivisione in tre diverse categorie catalogate secondo specifici path:

1. I dati a livello regionale delle votazioni ai singoli partiti nella Nazione
2. I dati a livello circoscrizionale delle votazioni ai singoli partiti all'Estero
3. I dati della Valle D'Aosta ai singoli partiti

Ho dunque suddiviso i dati nelle rispettive cartelle di destinazione.

Problematiche

Durante lo sviluppo ho incontrato alcune criticità:

1. Comprensione del codice e della struttura.

Inizialmente ho riscontrato delle difficoltà ad operare su codice non da me sviluppato. In particolare la presenza di metaclassi e la conseguente creazione a runtime della struttura non mi ha permesso di comprendere immediatamente la struttura globale.
- Risoluzione : Nonostante il tempo impiegato, un'attenta analisi di come i dati fossero elaborati e di come le classi venissero inizialmente configurate ha aiutato ad avere chiara l'esecuzione.

2. Utilizzo della sintassi usata per la creazioni di classe personalizzate.

Durante questo progetto mi sono resa conto dell'importanza di documentare il codice prodotto. Il progetto precedente utilizzava file .yaml come struttura di base per la creazione delle configurazioni delle classi. Questi file richiedevano regole di produzione la quale comprensione parziale avrebbe reso limitante la possibilità di sfruttare al meglio lo strumento.
- Risoluzione : Testing e simulazione sono stati step fondamentali al fine di assicurarmi che il progetto fosse consistente. Infatti ho impiegato una buona percentuale di tempo ad assicurarmi che i campi inseriti fossero corretti, confrontando e alterando l'output.

3. Traduzione in codice della Legge Elettorale.

Per una corretta e completa implementazione della legge elettorale ho dovuto approfondire il procedimento secondo cui opera tale legge. Sono quindi andata a consultare il testo originale tramite il sito della Camera. Alcune parti hanno reso

necessaria una complessa procedura di manipolazione dei dati. Si è resa necessaria particolare attenzione nella suddivisione dei seggi.

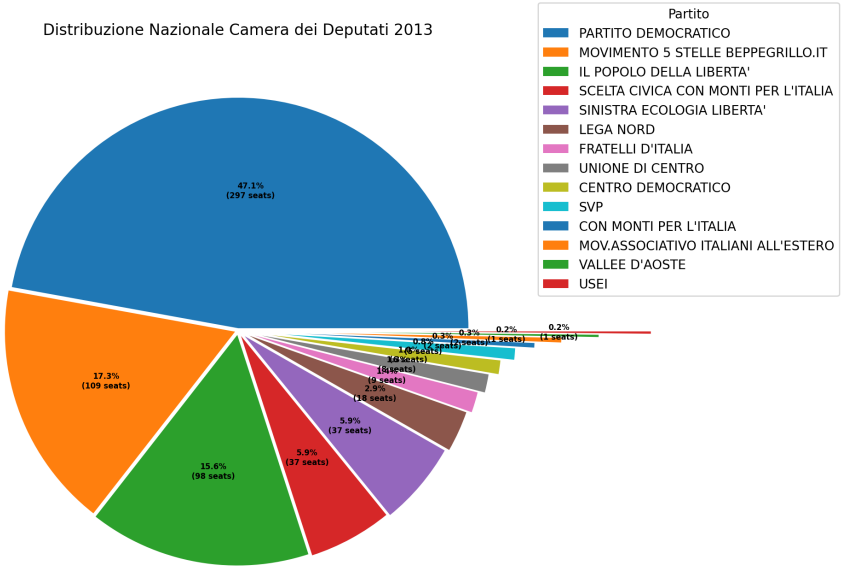
-Risoluzione: Per una più agevole procedura ed elaborazione ho dovuto configurare in maniera differente alcune classi cosi che le informazioni aggiuntive passassero ai livelli inferiori.

Risultati Ottenuti

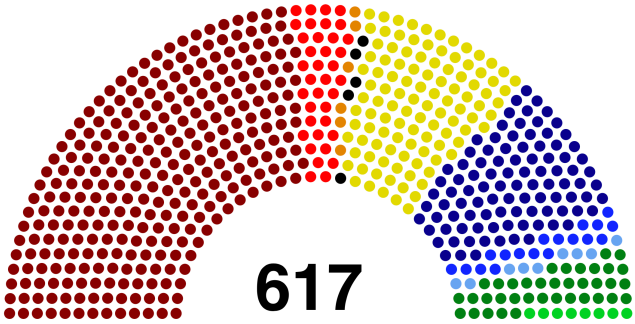
Output

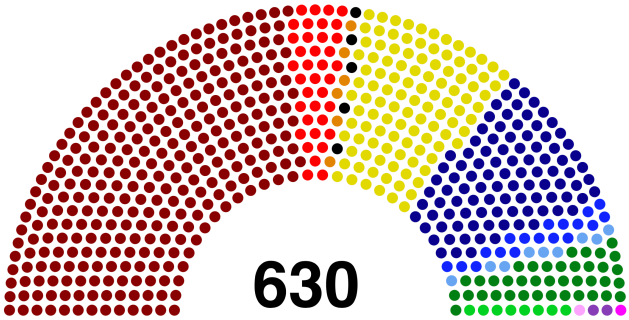
Il programma genera un duplice output, testuale e visivo. L'output testuale è formattato in modo da essere una lista di tuple d'informazioni.
Ogni tupla contiene:
- Nome della circoscrizione
- Nome della lane
- Nome del partito
- Numero dei seggi ricevuti nella circoscrizione della lane specificata

Per l'output visivo ho invece ritenuto opportuna una rappresentazione più semplice dei risultati. La mia scelta è quindi ricaduta su un pie chart in grado di rendere immediata la comprensione della distribuzione in percentuale dei seggi.



Le immagini di seguito riportate sono state generate per scopo puramente illustrativo della relazione.
Purtroppo la libreria utilizzata non era in grado di generare una visualizzazione di questo tipo.





Utilizzo e configurazione

Per il corretto funzionamento della simulazione è necessario avere una directory con una specifica configurazione di subdirectory e file.
La root deve contenere differenti folders:
- *Classes* : deve avere al suo interno appositi file di tipo .yaml o .py per la configurazione della struttura delle classi necessarie per la simulazione.
- *Data* : deve contenere al suo interno subfolders aventi lo stesso nome delle classi che attingono ai dati contenuti nell'omonima cartella.
- *Instances*: deve contenere al suo interno file .yaml contenenti i valori dei parametri delle varie istanze di ogni classe presente in Classes.

Dopo aver creato gli appositi file di configurazione delle classi è necessario formare i file atti ad istanziarli.
Per ogni singola classe deve corrispondere, all'interno di "Instances", un file che ha la struttura di un dizionario da cui vengono prelevate le chiavi usate come nome delle varie istanze.
Il valore associato alle chiavi è un dizionario avente per chiavi i nomi dei parametri della classe e per valore il valore da associare al parametro stesso.

Per il passaggio dei dati alle classi è necessario creare all'interno di ```Data``` una subdirectory avente lo stesso nome della classe destinazione. Al suo interno vengono successivamente salvati i file .csv reperiti dalle appropriate fonti .

Ogni file sopracitato ha il nome della funzione chiamante, come ad esempio ```Data/Nome_Classe/Nome_Funzione_Chiamante.csv```.
Nei file inerenti ai dati citati precedentemente la prima colonna è utilizzata per la divisione dei dati.

Una volta configurata la cartella è possibile eseguire la simulazione usando i seguenti comandi in una console di python :

```python
import src
src.run_simulation("/path/to/folder")
```
in alternativa, questo comando da terminale :
```shell
python -m src /path/to/folder
```
Nel caso si volesse implementare un'ulteriore Legge Elettorale, consiglio la creazione di un file dedicato nella cartella src/Commons contenente metodi specifici per l'elaborazione delle distribuzioni ed eventuali correzioni previste dalla Legge Elettorale d'interesse.

Sviluppi futuri

Idee

1. Tra gli sviluppi futuri più significativi c'è sicuramente l'implementazione di ulteriori Leggi Elettorali. In questo modo lo scenario di simulazione sarebbe più ampio e renderebbe possibile anche il confronto tra i risultati ottenuti da leggi elettorali differenti.