Polarizzazione delle Opinioni nei Social Network

Berretti Michele, Fontana Aleksandar

21 dicembre 2022

Sommario

I Social Network sono diventati un mezzo onnipresente per la discussione di questioni morali e politiche, si crede infatti che questi abbiano notevolmente contribuito ad alterare il corso di numerosi eventi storici come la Primavera Araba e le presidenziali degli Stati Uniti. Basandoci su di un campione sufficientemente grande di interazioni su Twitter riguardo questioni tradizionalmente polarizzanti nella politica statunitense (Matrimoni omosessuali, controllo delle armi e cambiamenti climatici) introduciamo un modello matematico per la polarizzazione delle opinioni nei Social Network. Forniamo poi una definizione analitica di polarizzazione sfruttando il concetto di comunità in un grafo e impiegando la tecnica di discesa del gradiente confrontiamo i risultati numerici con quelli empirici per individuare i parametri del modello che meglio rappresentano i dati a nostra disposizione. Sviluppiamo infine vari modelli di machine learning con i dati a nostra disposizione.

1 Introduzione

Servizi e prodotti digitali sono sempre più protagonisti delle nostre vite quotidiane ([Lambiotte and Kosinski, 2014]), possiamo fare l'esempio di servizi di messaggistica, di e-commerce e web-banking. Questa crescente fusione col mondo digitale implica che il comportamento delle persone, il loro modo di comunicare, i loro spostamenti e persino il loro stato psicologico possono essere facilmente registrati e studiati ([Lazer et al., 2009]). Twitter, uno dei più grandi Social Network al mondo, è diventato parte integrante della vita lavorativa e sociale ([Miller, 2011],

[Weinberger, 2011], [Lazer et al., 2009]) di circa mezzo miliardo di utenti nel mondo. Il suo potenziale come mezzo di ricerca per le scienze sociali non deve essere trascurato in quanto fornisce un grande numero di strumenti e la possibilità di ottenere campioni ampi e diversi della nostra società. Recentemente in [Brady et al., 2017] è stato studiato, impiegando una community campione di circa 270mila utenti su Twitter e 1 milione di retweet, il fenomeno del ”moral contagion” ovvero come l'aggiunta di contenuti emotivi incrementasse la diffusione e la manipolazione di opinioni. Nella stessa pubblicazione hanno poi mostrato come la comunità da loro studiata presentasse un evidente effetto di polarizzazione. Ispirandoci al loro lavoro e basandoci sulla stessa community campione ci siamo proposti di studiare come poter rappresentare matematicamente questo effetto di polarizzazione nei Social Network.

Nell'articolo di [Brady et al., 2017], i comportamenti della comunità precedentemente descritta sono studiati da un punto di vista sociologico, classificano gli utenti in ”conservatori” o ”liberali” basandosi sul loro punto di vista riguardo tre argomenti polarizzanti della società statunitense (Matrimoni omosessuali, controllo delle armi e cambiamenti climatici) e mostrano infine dal punto di vista qualitativo una rete fortemente polarizzata. Quello che manca tuttavia è una definizione quantitativa di polarizzazione e una definizione di comunità che permetta di tradurre il loro lavoro in un modello matematico in grado di generalizzare il fenomeno e che può essere confrontato con dati reali. Mostriamo a tal proposito in Figura 1 una ricostruzione del grafo con i dati forniti e del grafo ottenuto dal modello.

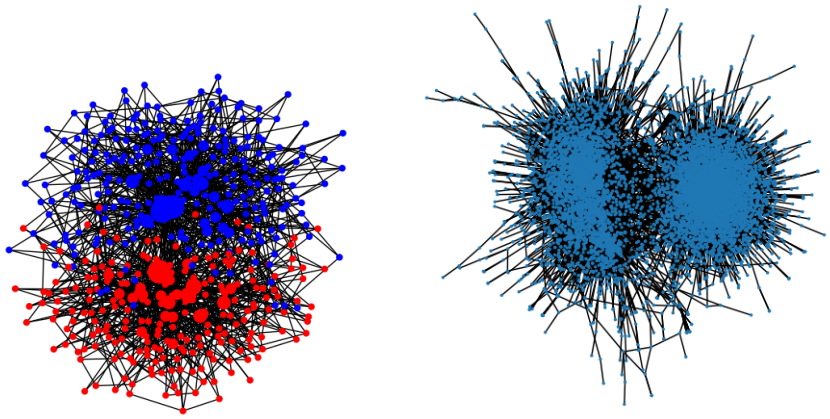

Figura 1: Sinistra: Grafo ottenuto dal modello. Destra: grafo ottenuto dai dati.

Lo scopo di questo lavoro è dunque quello di fornire un modello matematico in grado di descrivere quantitativamente la polarizzazione di una rete sociale in base ad un qualunque livello di coesione sociale. La validazione di tale modello viene poi effettuata trovando i parametri che meglio rappresentano i dati empirici forniti.

Sviluppiamo infine diversi modelli di Machine Learning con l'obbiettivo di assegnare un Orientation ad un qualunque utente estraneo al training set.

2 Analisi del Dataset

Prima di poter procedere con una descrizione del nostro modello è necessario analizzare nel dettaglio i dati su cui stiamo basando il nostro studio. Il dataset fornito da [Brady et al., 2017] consiste di circa 270mila utenti su Twitter e di poco più di 1 milione tra commenti a tweet e retweet in un periodo che va da Marzo ad Aprile del 2017, dunque senza particolari eventi di stress per la politica americana. Il dataset risulta diviso in tre csv contenenti tutti i tweet corrispondenti ai tre topic dell'articolo, poichè i csv forniti presentano numerose colonne inutili al nostro studio è stato necessario effettuare una pulizia dei dati prima dell'avvio del nostro studio.

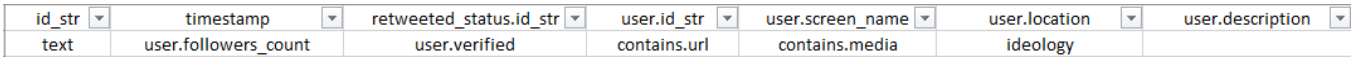

Figura 2: Titoli delle colonne dei dataset forniti da [Brady et al., 2017]

Infatti le uniche colonne utili per il progetto sono state:

- "id_str": un codice univoco che riconosce il singolo utente, particolarmente utile per poter comparare la la nostra edge list con quella fornitaci da [Brady et al., 2017];
- "user.id_str": cioè l'username dell'utente, fondamentale per poter ricavare dai testi le connessioni tra i vari nodi;
- "text": in cui è contenuto il testo del singolo tweet e il testo dei tweet che ha retwettato l'utente;
- "ideology": rappresenta il valore di ideology fornito dai creatori dell'articolo ed esso è univoco per ogni utente, non è quindi diviso in base al topic preso in considerazione.

Per poter ricreare i collegamenti tra i nodi, utili per il capitolo **Analisi dei dati** abbiamo estratto tutti gli username presenti nel "text" e attentamente li posti in lower case, abbiamo preso in considerazione solo quelli di cui si aveva un valore di ideologia e poi abbiamo creato la lista di collegamenti dei nodi, abbiamo anche differenziato gli username taggati dagli utenti e gli username retweetati. Abbiamo tentato di ricostruire anche l'ideologia dei nodi più citati ma non presenti nel dataset iniziale, ma si è rivelato una soluzione poco affidabile perchè non riusciamo a stimare l'errore associato a questo metodo. Vediamo che a ciascun nodo è

attribuito uno score di Ideologia da parte degli autori di [Brady et al., 2017] che fornisce un valore stimato dell'orientamento politico del nodo considerato e della convinzione di tale pensiero. Valori negativi di Ideologia indicano un utente conservatore e valori positivi di Ideologia indicano un utente liberale, maggiore è il modulo di tale grandezza e maggiore è la convinzione di tale utente riguardo la sua idea politica. Con l'obbiettivo di fornire una struttura notevolmente semplificata del nostro dataset definiamo l'**Orientation** come la versione "discreta" dell'Ideologia utile per il modello a due stati che presenteremo nel prossimo capitolo. Ad ogni nodo con Ideologia inferiore a -0.5 attribuiamo un valore di Orientation pari a -1, ad ogni nodo con Ideologia superiore a 0.5 attribuiamo un Orientation pari a 1 e ai rimanenti nodi, neutrali, attribuiamo un valore di Orientation pari a 0. Possiamo dunque vedere la struttura del nostro dataset nell'istogramma a seguire:

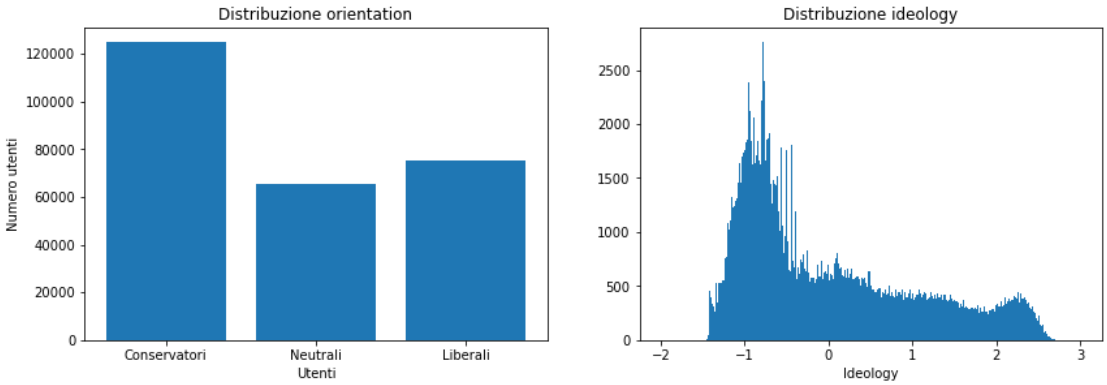

Figura 3: A sinistra possiamo vedere il numero di utenti liberali, conservatori e neutrali, a destra vediamo la distribuzione degli utenti basati sul loro valore di Ideology.

Per quanto riguarda invece lo studio di machine learning abbiamo deciso di unire tutti i tweet e relativi retweet annessi di un utente in un unico testo, proprio perchè il valore di ideology è unico ed associato all'utente e non al singolo messaggio.

3 Modello

Introduciamo adesso il modello che descrive la relazione tra la struttura della rete e le opinioni delle persone che la compongono. La prima grande semplificazione che effettuiamo è quella di attribuire agli utenti un'opinione decisa, come abbiamo già visto precedentemente introducendo l'Orientation. Definiamo dunque gli stati $\sigma_i = \pm 1$ per il nodo i a seconda che questo sia rispettivamente liberale o conservatore. Quest'approssimazione ci permette dunque di trattare il modello alla stregua di un modello di Ising ([Ising, 1925], [Huang, 1987]). I nodi entrano nella rete con uno stato σ_k scelto casualmente per ogni k e si connettono ai nodi già presenti con una variante della strategia del "Preferential Attachment" di Barabasi-Albert che tenga però conto dello stato dei nodi.

Il nodo all'ingresso esegue r collegamenti con nodi scelti casualmente tra i già presenti nella rete con una probabilità data da:

$$P_k(t) = \frac{e^{-\beta\sigma_0\sigma_k} K_k(t)}{\sum_j e^{-\beta\sigma_0\sigma_j} K_j(t)}, \tag{1}$$

dove $P_k(t)$ è la probabilità che all'iterazione t il nodo k con stato σ_k effettui un collegamento con il nodo appena introdotto di stato σ_0, $K_k(t)$ indica il grado del nodo k. Definiamo il parametro β, parallelo dell'inverso della temperatura nel modello di Ising, come un **indice di omofilia**. Il comportamento del sistema al variare di questo sarà approfondito nel prossimo paragrafo.

Reiterando questo processo N volte siamo in grado di costruire una rete di N nodi in grado di mostrare effetti di polarizzazione come quelli apprezzabili in Figura 3 dell'articolo di [Brady et al., 2017] che variano in intensità al variare del parametro β del modello.

3.1 Analisi del Modello

Nel capitolo precedente abbiamo introdotto l'espressione di $P_k(t)$, ovvero della probabilità di creazione di un link tra due nodi, e abbiamo visto come questa dipenda, oltre che dal grado dei nodi e dal loro stato, dal parametro β che rappresenta l'equivalente dell'inverso della temperatura in fisica. Ci proponiamo in questo paragrafo di studiare il comportamento del modello al variare sia di tale parametro mantenendo fissato il numero di link in ingresso, $r = 3$, che della dimensione della rete.

Prima di poter vedere però come varia il nostro modello al variare di β dobbiamo introdurre una grandezza in grado di quantificare la polarizzazione del siste-

ma. Sfruttando la bisezione in comunità di Kernighan-Lin ([Kernighan and Lin, 1970]) implementata nella libreria NetworkX per Python, quantifichiamo le due comunità del sistema come:

$$m_j = \frac{\sum_i \sigma_{i,j}}{|c_j|} \tag{2}$$

con $j = 1, 2$ che indica quale delle due comunità stiamo considerando e $|c_j|$ indica la dimensione della comunità j. $\sigma_{i,j}$ è dunque lo stato del nodo $i-$esimo appartenente alla comunità j.

La bisezione di Kernighan-Lin ([Kernighan and Lin, 1970]) è un algoritmo euristico per la ricerca di partizioni in un grafo. Dato un grafo non diretto e non pesato $\mathcal{G}$ con vertici V e collegamenti E, l'obbiettivo dell'algoritmo è quello di suddividere i nodi V in due partizioni tali che il numero di collegamenti tra le due sia minimo. È un algoritmo iterativo che migliora la qualità delle partizioni ad ogni step sfruttando un algoritmo "**greedy**" (effettua la scelta ottimale *locale* ad ogni iterazione) per accoppiare i vertici della partizione A con quelli della partizione B così che scambiandoli migliora la suddivisione. Dato un grafo con n nodi, ogni iterazione impiega un tempo $O(n^2 \log n)$.

Siamo dunque in grado di fornire una definizione quantitativa di **polarizzazione** come:

$$Pol = \sigma^2\{m_1, m_2\} \tag{3}$$

Mostriamo dunque in figura 4 come varia la polarizzazione al variare del parametro β per un diverso numero di nodi.

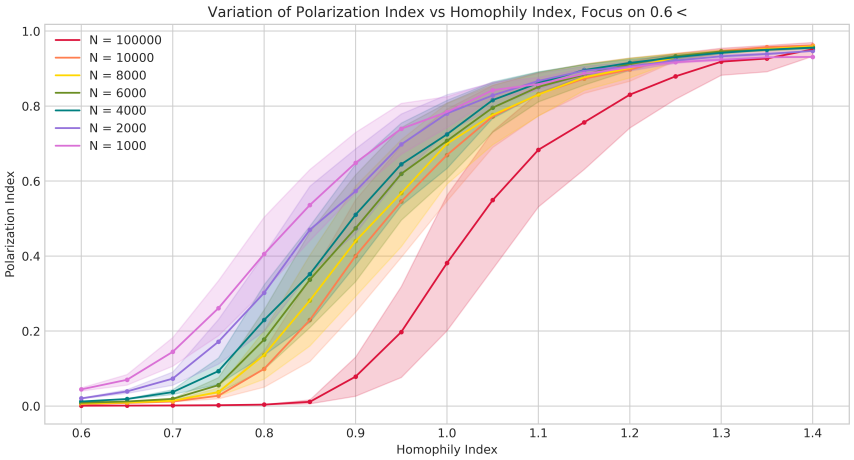

Figura 4: Variazione della polarizzazione in funzione di β per N = 1000, 2000, 4000, 6000, 8000, 10000 e 100000. L'intersezione delle curve potrebbe essere un indicazione di una transizione di fase del sistema. Notiamo come per β che tende a zero la polarizzazione si annulla e otteniamo un grafo di Barabasi-Albert. Valori ottenuti mediando 1000 grafi per ogni valore di β.

Proponiamo poi un ingrandimento di tale curva per valori di β compresi tra 0.7 e 0.8 effettuato con un considerevole aumento della statistica, passando dal mediare 1000 grafi al mediare 10000 grafi.

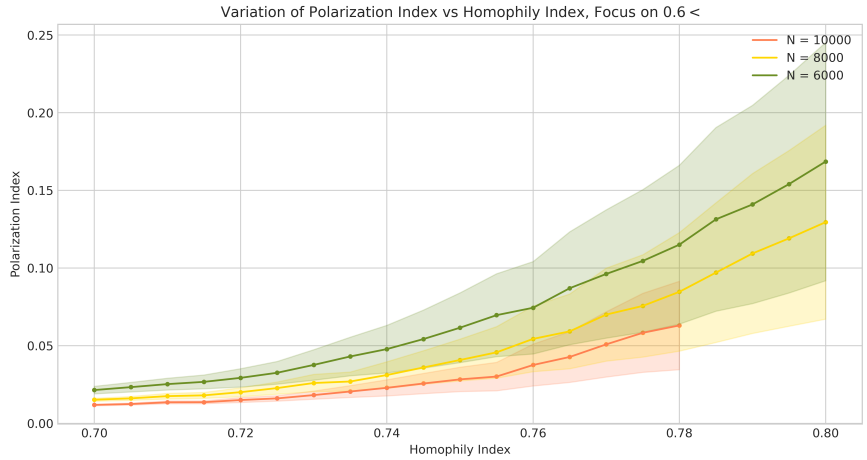

Figura 5: Variazione della polarizzazione in funzione di β per valori compresi tra 0.7 e 0.8. Valori ottenuti mediando 10000 grafi per ogni valore di β

L'intersezione delle curve di polarizzazione sembra occorrere per $1.05 \leq \beta \leq 1.30$ e potrebbe essere sintomo di una transizione di fase di ordine superiore al primo. A tal proposito effettuiamo un fit lineare della curva di polarizzazione:

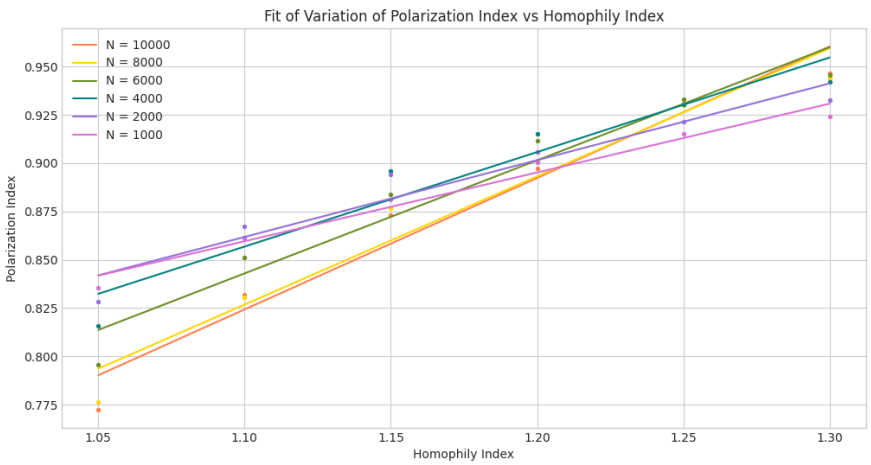

Figura 6: Fit Lineare variazione polarizzazione in funzione di beta per questo compreso tra 1.05 e 1.30.

Mostriamo infine (Figura 7) la variazione dei parametri del fit al variare del numero di nodi della rete. Il crescere del coefficiente angolare della curva di polarizzazione all'aumentare del numero di nodi significa che la polarizzazione avviene sempre più rapidamente all'aumentare del volume del sistema. Tuttavia questo fenomeno da solo non è sufficiente per giustificare una transizione di fase del sistema di qualunque ordine.

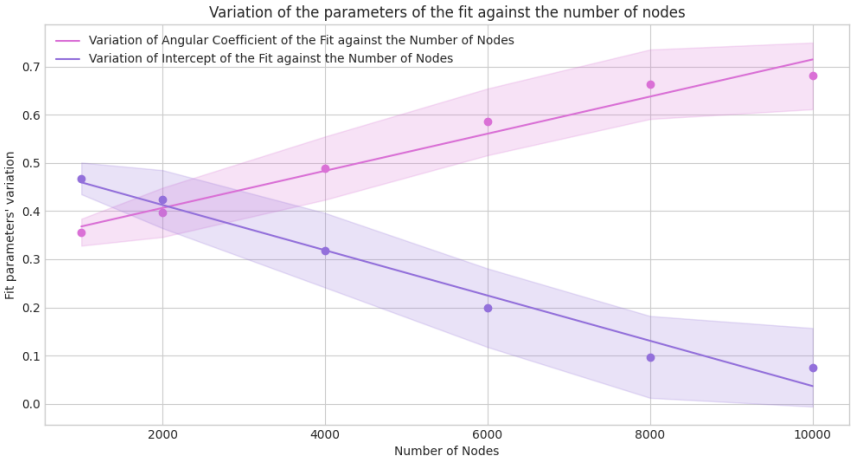

Figura 7: Variazione dei parametri del fit delle curve di transizione al variare del numero di nodi.

Per verificare la presenza o meno di una transizione di fase del primo ordine dobbiamo assicurarci che l'entropia del sistema, o analogamente la derivata prima della polarizzazione abbia una discontinuità al variare di beta. Tuttavia essendo, nel nostro caso, l'entropia basata sulla distribuzione di grado, questa non è in grado di fornire informazioni sulla transizione di fase per il nostro modello. Abbiamo comunque realizzato a scopo didattico l'entropia del nostro modello che andiamo qui a definire come:

$$\mathcal{H} = -\sum_{k=1}^{max\ degree} P(k) \log P(k) \tag{4}$$

dove $P(k)$ è la probabilità di avere un nodo di grado k.

Vediamo dunque l'andamento dell'entropia del sistema al variare dell'indice di omofilia:

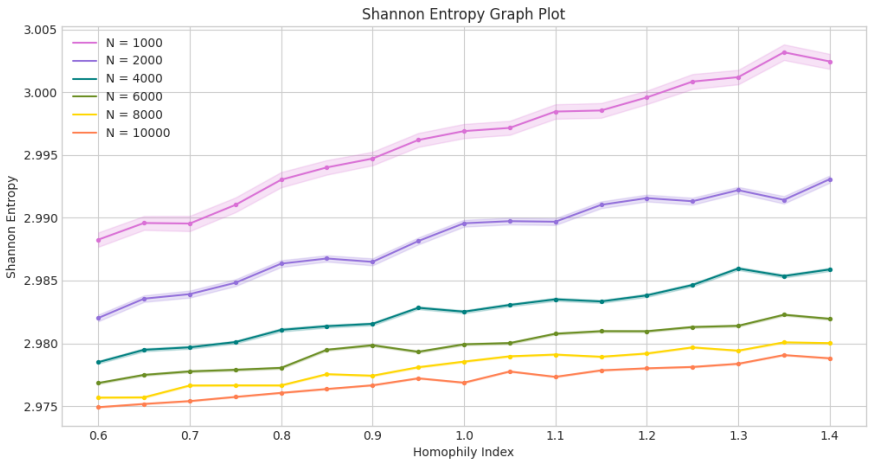

Figura 8: Variazione dell'entropia del sistema al variare del parametro beta. Il fatto che la funzione sia continua indica assenza di transizione di fase del primo ordine.

Andiamo dunque a verificare l'assenza di una transizione di fase del primo ordine calcolando la derivata della polarizzazione numericamente. Partendo dalla definizione di derivata come limite del rapporto incrementale:

$$f'(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon} \tag{5}$$

otteniamo per la derivata prima della polarizzazione:

$$Pol'(\beta) = \lim_{\epsilon \to 0} \frac{Pol(\beta+\epsilon) - Pol(\beta)}{\epsilon} \tag{6}$$

Andiamo dunque a mostrare l'andamento della derivata prima della polarizzazione al variare sia dell'indice di omofilia β che il volume del sistema:

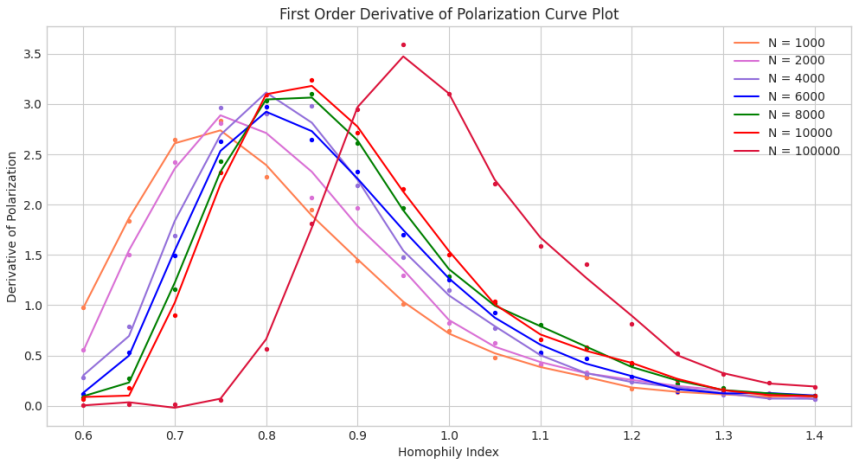

Figura 9: Andamento della derivata prima della polarizzazione al variare del parametro β.

Come ci aspettavamo la funzione risulta continua dimostrando ulteriormente l'assenza di una transizione di fase del primo ordine.

Come si può vedere dal grafico (i punti non coincidono con la curva) alla curva della derivata è stato applicato un filtro **Savitzky-Golay** con lo scopo di "smoothare" il comportamento di questa. Il filtro Savitzky-Golay è un filtro digitale solitamente impiegato per incrementare la precisione dei dati senza distorcerne la tendenza. Questo viene ottenuto tramite un processo detto di convoluzione, adattando successivi sottoinsiemi di valori adiacenti con un polinomio di basso ordine, nel nostro caso è stato considerato un polinomio di terzo grado e i sottoinsiemi considerati sono di 5 elementi. È bene precisare che tale filtro può essere impiegato senza ulteriori aggiustamenti solo nel caso in cui i punti del dataset siano ugualmente spaziati.

Andiamo dunque a verificare la possibile presenza di una transizione di fase del secondo ordine. A tale scopo dobbiamo controllare la continuità della derivata seconda della polarizzazione, ottenuta numericamente dal seguente limite della stima del rapporto incrementale:

$$Pol''(\beta) = \lim_{\epsilon \to 0} \frac{Pol(\beta + \epsilon) - 2Pol(\beta) + Pol(\beta - \epsilon)}{\epsilon^2}. \tag{7}$$

Analogamente al caso precedente viene applicato un filtro Savitzky-Golay per lo smoothing della curva. Mostriamo dunque l'andamento della derivata seconda

al variare dell'indice di omofilia in figura:

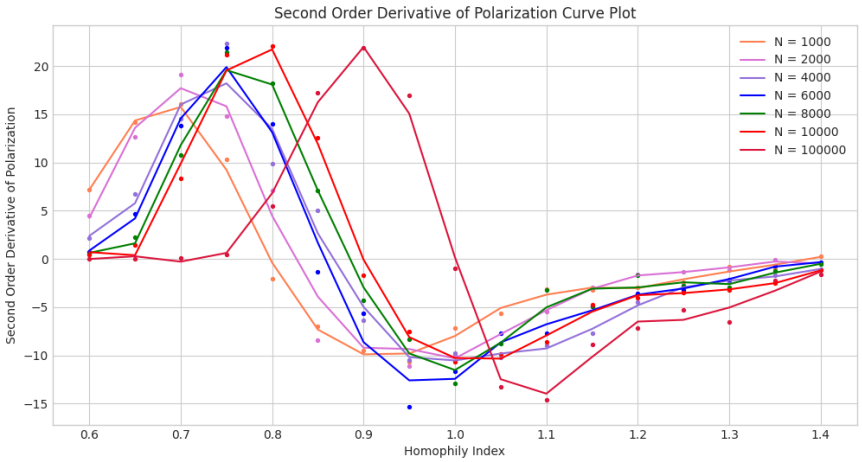

Figura 10: Andamento della derivata seconda della polarizzazione al variare di β.

Essendo anche la derivata seconda della polarizzazione una funzione continua possiamo escludere una transizione di fase del secondo ordine. L'assenza di transizioni di fase del primo e del secondo ordine è sufficiente per poter affermare che, per i volumi da noi considerati, il sistema non subisce alcuna transizione di fase di qualunque ordine e si sposta con continuità da uno stato non polarizzato ad uno polarizzato. Il risultato del resto non ci sorprende, ricordiamo infatti che il modello di Ising unidimensionale (su cui il nostro modello è largamente basato) non subisce anche esso alcuna transizione di fase ([Ising, 1925]).

Andiamo a vedere adesso come si comporta il sistema se invece di inizializzare il processo di costruzione del grafo con due soli nodi lo si inizia con un numero n di nodi iniziali che hanno fatto link in maniera del tutto casuale.

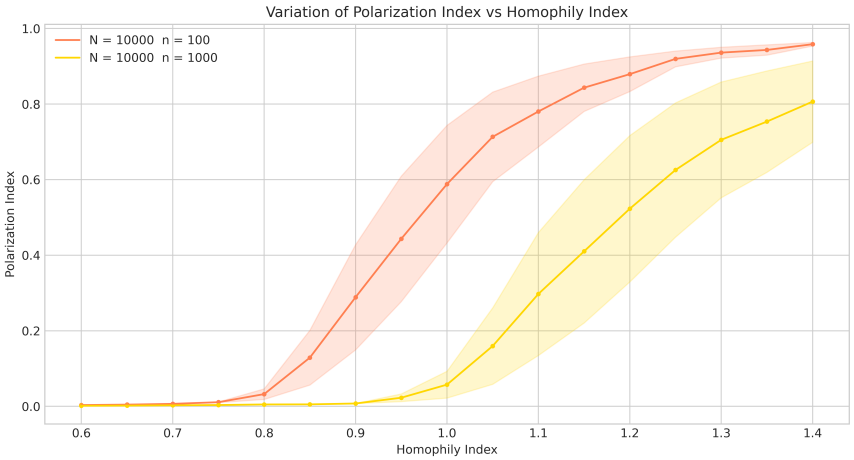

Figura 11: Andamento polarizzazione al variare di β per 10000 nodi e variando il numero di nodi iniziali.

Come possiamo vedere otteniamo un ritardo nell'effetto di polarizzazione all'aumentare del numero di nodi iniziali.

Ci soffermiamo infine sull'analisi di una singola comunità ottenuta tramite la bisezione di Kernighan-Lin. Realizziamo un grafo di Barabasi-Albert con lo stesso numero di nodi di una delle due community nel limite per β molto grandi utilizzando la libreria NetworkX per Python. Riportiamo in figura 12 la distribuzione di grado dei due grafi:

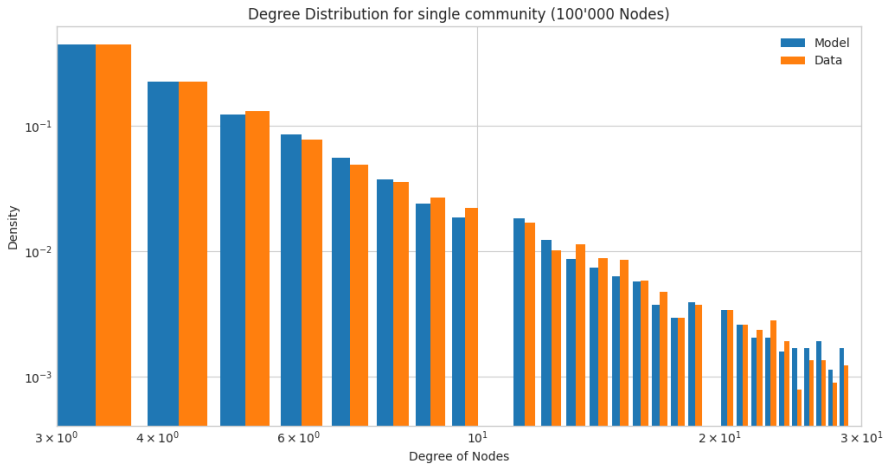

Figura 12: Distribuzione di grado di una singola comunità del modello e di un grafo di Barabasi-Albert con 100000 nodi. Possiamo notare come sia evidente la coincidenza tra le due distribuzioni.

Dunque andando ad osservare i comportamenti del nostro modello agli estremi ($\beta = 0$ e $\beta \to \infty$) otteniamo rispettivamente un grafo di Barabasi-Albert per $\beta = 0$ e due grafi di Barabasi-Albert diversi che si separano per $\beta \to \infty$.

3.2 Analisi dei Dati

Avendo studiato i dati empirici a noi disponibili nella precedente sezione, vediamo adesso come poter conciliare questi con il nostro modello e trovare dunque i parametri che meglio si adattano ad una rappresentazione dei dati. Per individuare tali parametri definiamo innanzitutto una funzione di costo $\mathcal{L}$ come:

$$\mathcal{L} = \left(Pol_{target} - Pol(\beta_{try})\right)^2 \tag{8}$$

con β_{try} il valore dell'indice di omofilia che sta provando in quel passo.

Essendo la polarizzazione funzione solo di β, il gradiente di questa coinciderà con la sua derivata in β. Procediamo dunque con il calcolo numerico della derivata partendo dalla definizione come limite del rapporto incrementale:

$$f'(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}, \tag{9}$$

abbiamo

$$\mathcal{L}'(\beta) = \lim_{\epsilon \to 0} \frac{(Pol_{target} - Pol(\beta + \epsilon))^2 - (Pol_{target} - Pol(\beta)^2}{\epsilon}. \tag{10}$$

Definiamo dunque il β del passo successiva come:

$$\beta^{(n+1)} = \beta^{(n)} - \alpha \mathcal{L}'(\beta^{(n)}). \tag{11}$$

Dal punto di vista operativo forniamo al codice un valore di β iniziale con cui realizzare un numero N di grafi per calcolarne il valore medio della polarizzazione. Quest'operazione di media sui valori di *Pol* è necessaria siccome per valori di beta sulla transizione di fase l'errore a questa associato è nell'ordine del $10 - 20\%$. Ne calcoliamo poi la derivata e il successivo valore di beta, reiteriamo infine fino a quando la variazione del valore di β ottenuto è inferiore alla precisione sul valore di β da noi richiesta.

Illustriamo dunque in figura 13 i risultati da noi ottenuti:

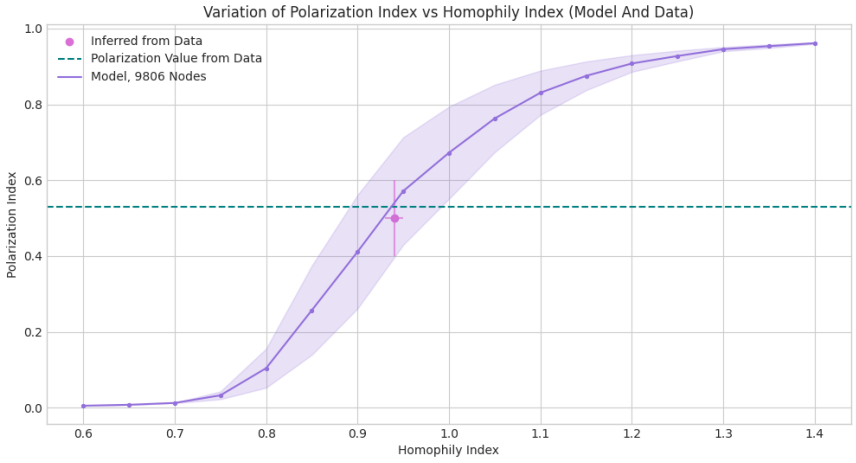

Figura 13: In viola: Variazione della polarizzazione al variare dell'indice di omofilia, simulazione MC con 1000 iterazioni. In Verde: Valore ottenuto direttamente dai dati della polarizzazione, ovviamente indipendente da β in quanto è quello ottenuto dai dati empirici. In rosa: Valore dell'indice di omofilia ottenuto tramite la discesa del gradiente e il corrispettivo valore di polarizzazione ottenuto mediando su 1000 iterazioni.

Possiamo dunque vedere un ottimo raccordo tra la polarizzazione dei dati e la polarizzazione ottenuta dal modello con i parametri ottenuti dall'ottimizzatore.

Vediamo infine il numero di collegamenti che vengono effettuati tra nodi con stessa Orientation e con Orientation opposta per: il nostro modello al variare del numero di nodi, i dati a nostra disposizione e il nostro modello con i valori dei parametri ottenuti dall'ottimizzatore. Anche in questo caso i dati hanno un buon raccordo con il nostro modello come possiamo vedere dal grafico mostrato in figura 14.

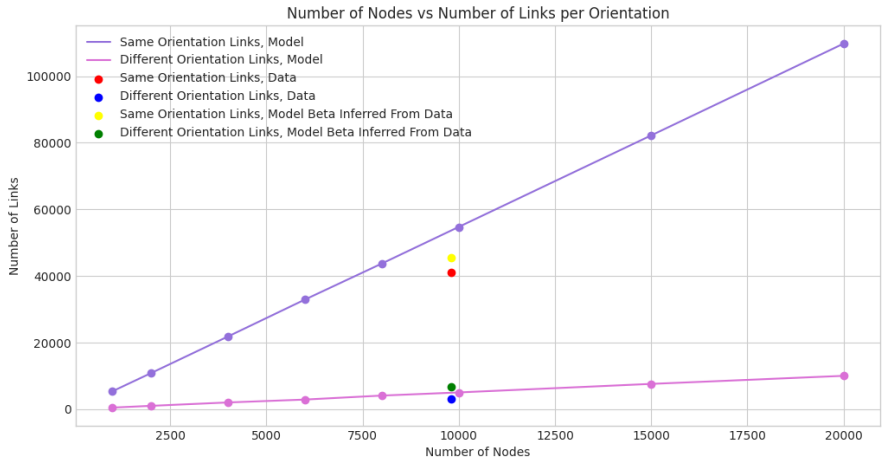

Figura 14: Numero di collegamenti tra nodi con stessa Orientation o Orientation diversa al variare del numero di nodi. In rosso il numero di collegamenti tra nodi con stessa Orientation per i dati sperimentali e in blu il numero di collegamenti tra nodi con diversa Orientation per i dati sperimentali. In giallo il numero di collegamenti tra nodi con stessa Orientation per il modello con i parametri ottenuti dall'ottimizzatore e in verde il numero di collegamenti tra nodi con diversa Orientation per il modello con i parametri ottenuti dall'ottimizzatore.

Si può notare come il modello tenda a sovrastimare il numero di collegamenti tra nodi di orientation diversa. Per risolvere questo problema si può pensare di generalizzare il modello nel seguente modo:

$$P_k(t) = \frac{e^{-\beta\sigma_0\sigma_k} K_k^s(t)}{\sum_j e^{-\beta\sigma_0\sigma_j} K_j^s(t)}, \tag{12}$$

Dove s rappresenta la potenza con cui il Barabasi-Albert influisce sul modello, valori maggiori di 1 di questo parametro fanno sì che i nodi con il maggior numero di collegamenti siano sempre più favoriti, ed è proprio questo il caso che vogliamo

studiare. Prima di determinare quali siano i parametri che meglio esprimono effettuiamo preliminarmente un ulteriore passaggio; poichè il nostro obiettivo è quello di migliorare l'accordo dei collegamenti tra nodi con stessa ideologia e ideologia diversa, poniamo che r_{max} vari ad ogni step seconda una distribuzione di probabilità, più precisamente dapprima utilizziamo una distribuzione normale ($\mu = 2.97$, $\sigma^2 = 1$) e una distribuzione di Poisson ($\lambda = 2.28$). Ricordiamo che r_{max} rappresenta il numero massimo di collegamenti che un nodo può fare appena entrato nel grafo, quindi variando questo valore si può determinare a priori il numero di link tra nodi.

Tabella studio collegamenti nodi con ideologia simile e diversa

	Polarizzazione	**Collegamenti tra nodi con stessa Orientation**	**Collegamenti tra nodi con Orientation diversa**
Dati Sperimentali	0.53±0.01	41118	3078
Modello Base, Distribuzione Normale	0.5±0.2	40300±100	3740±60
Modello Base, Distribuzione Poisson	0.5±0.2	40000±300	4000±100
Modello Generalizzato, Distribuzione Normale	0.5±0.2	41000±500	3300±500
Modello Generalizzato, Distribuzione Poisson	0.5±0.2	41300±500	3100±400

Figura 15: Tabella con valori polarizzazione, il numero di collegamenti tra nodi con ideologia simile e diversa al variare degli scenari considerati per la nostra analisi.

Vediamo in 15 come, utilizzando la formula generale, l'accordo tra i dati sperimentali e del modello sia ottimo, inoltre notiamo che la scelta tra la distribuzione normale e di Poisson non sembra invece influire molto sui risultati finali.

Come possiamo vedere in figura (1), considerando sia retweet che commenti ed escludendo i nodi neutrali, ovvero con un valore di ideologia compreso tra -0.5 e 0.5 otteniamo un effetto di polarizzazione apprezzabile. Considerando invece

Tabella Polarizzazione e Numero di Isole

	Numero Isole	Polarizzazione
Reply+Retweet Con Neutri	286	0.62 ± 0.03
Reply+Retweet Senza Neutri	605	0.56 ± 0.02
Solo Reply Con Neutri	135	0.004 ± 0.003
Solo Reply Senza Neutri	452	0.001 ± 0.002

Figura 16: Tabella con valori polarizzazione e numero di isole al variare degli scenari considerati per la nostra analisi.

anche i nodi neutrali il valore di polarizzazione diminuisce. Inserendo nodi neutrali stiamo aumentando il numero di nodi che appartengono alle due community senza però influire sulla somma degli stati di queste (il contributo del nodo neutro è 0) e ciò potrebbe far intendere una diminuizione della polarizzazione. Tuttavia come possiamo vedere dalla diminuizione del numero di isole, considerando i nodi neutrali aumenta il numero di nodi con Orientation presenti nel componente principale aumentando dunque la polarizzazione.

Abbiamo successivamente riconsiderato entrambi i casi escludendo i retweet e considerando solo i commenti ad un determinato tweet. In tal caso la polarizzazione tende ad annullarsi dimostrando come nella sezione commenti di un determinato tweet siano presenti all'incirca in ugual numero commenti positivi e negativi riguardo l'idea politica contenuta nel tweet commentato.

Vediamo poi in tabella come, nei quattro casi precedenti, vari il numero di isole, ovvero il numero di piccole comunità di nodi che creano collegamenti tra di loro senza poi comunicare con il resto dei componenti del grafo.

4 Applicazione di tecniche di Machine Learning

Una volta studiato nel dettaglio il nostro modello possiamo lasciarci alle spalle la questione più prettamente matematica del nostro studio e dedicarci unicamente ai dati empirici a nostra disposizione. L'analisi dei sentimenti è un particolare processo di classificazione di testi fine a determinare se un qualunque scritto sia

positivo, negativo o neutrale. Impostando correttamente un insieme di etichette per ogni testo nel ”training set” si può allenare un modello affinché faccia delle predizioni corrette su nuovi elaborati. Quello che ci proponiamo di fare in questa sezione è trainare un modello rifacendoci alle tecniche dell'analisi dei sentimenti tramite una partizione dei dati a nostra disposizione e sfruttare poi tale algoritmo per classificare qualunque utente, nuovo al modello, in liberale o conservatore.

Prima di procedere alla sezione di ”learning” è tuttavia necessario effettuare delle operazioni di base sui nostri dati in modo da presentare al codice di apprendimento i contenuti nel modo a lui più interpretabile possibile. Inizia dunque una lunga fase di ”**pre-processing**” dei nostri dati.

4.1 Pre-processing dei dati

Avendo largamente discusso la struttura dei dati a nostra disposizione nel paragrafo precedente, in questa sezione ci limiteremo a dire che questi sono stati organizzati in modo tale da avere un training set costituito da 5000 utenti liberali e altrettanti conservatori in una tabella, in figura 17, costituita da 3 colonne e N righe dove N è il numero di nodi (utenti, nel nostro caso 10mila).

	text	orientation
0	Markets*Weddings*Festivals	-1
1	IMO Libertarians, Republicans & RWNJS share th...	-1
2	ECOCARE Maldives is an independent non-profit-...	-1
3	#KoreanAmerican #charleston #engineer #hapa #K...	-1
4	News about Politics/Activism. #Liberal #Activi...	-1
...	...	...
9995	The current American political and geopolitica...	1
9996	Gods Golden Music	1
9997	Writer. Writer. Writer. Writer. Writer. Writer...	1
9998	Constitutional Conservative, U.S. Army Vet, 2A...	1
9999	Smoking a cigar & drinking a Presidente Smokin...	1

10000 rows × 2 columns

Figura 17: Tabella rappresentativa del training set. La prima colonna indica l'etichetta del nodo, la seconda contiene tutto il testo prodotto da tale nodo e nella terza è contenuto il suo punteggio di Orientation.

Ad ogni nodo è stato attribuito un'etichetta di "Orientation" a seconda del loro punteggio di "Ideology"; nel caso in cui quest'ultima fosse sopra 0.5 allora al nodo è stata attribuita un Orientation di +1 (liberale) e nel caso in cui l'Ideology fosse inferiore a −0.5 (conservatore). I nodi neutrali non sono stati presi in considerazione così da semplificare notevolmente il problema. Riportiamo dunque il codice scritto:

```
orientation_p = []
text_p = []

orientation_n = []
text_n = []

i=0
j=0

while len(orientation_p) < 5000:
  if ideology[i]>0.5:
    orientation_p.append(1)
    text_p.append(raw[i])
  i+=1

while len(orientation_n) < 5000:
    if ideology[j]<-0.5:
      orientation_n.append(-1)
      text_n.append(raw[j])
    j+=1


orientation_n.extend(orientation_p)
text_n.extend(text_p)
orientation = orientation_n
text = text_n
```

Andando a vedere i dati ci accorgiamo che possiamo trovare numerose tag HTML siccome i dati sono stati importati direttamente da Twitter. Per rimuovere queste tag non volute utilizziamo la funzione *html.parser* della libreria *BeautifulSoup*. Il codice che effettua questa pulizia è qui riportato:

```
def strip_html(text):
    soup = BeautifulSoup(text, "html.parser")
    return soup.get_text()

dataset["text"] = dataset["text"].apply(strip_html)
```

Prima di passare direttamente alla costruzione del modello dobbiamo effettuare delle ulteriori operazioni di pulizia dei dati. Sappiamo che un essere umano per poter classificare sentimenti abbia bisogno di articoli, congiunzioni, punteggiatura e così via. Tuttavia per le macchine non è questo il caso e per classificare i sentimenti correttamente è necessario rimuoverli per non generare confusione. Per effettuare questa operazione sfruttiamo la libreria *NLTK* (”Natural Language Processing ToolKit). Questo è il codice che implementa tale procedimento:

```
def punc_clean(text):
  import string as st
  a = [w for w in text if w not in st.punctuation]
  return ''.join(a)
dataset["text"] = dataset["text"].apply(punc_clean)

def remove_stopword(text):
  stopword = nltk.corpus.stopwords.words("english")
  stopword.remove("not")
  a=[w for w in nltk.word_tokenize(text) if w not in stopword]
  return ''.join(a)

dataset["text"] = dataset["text"].apply(remove_stopword)

dataset.to_csv("cleaned.csv")
```

Consideriamo importante far notare come tra le ”stopwords” abbiamo escluso il ”not” siccome aggiunge un contributo significativo per la comprensione del sentimento nel testo.

Siamo dunque ad un passo dalla costruzione del nostro modello, l’ultimo step prima di procedere è infatti quello di assegnare ad ogni parola un ”sentiment score”. Per fare ciò impieghiamo la libreria SKlearn e il modulo ”TfidVectorizer”. L’uso di TfidVectorizer è equivalente all’uso di CountVectorizer seguito da TfidfTransformer. Lo scopo del CountVectorizer è quello di trasformare un dato testo in un vettore basandosi sulla frequenza (count) con cui appaiono tutte le parole nel testo. CountVectorizer crea una matrice dove ogni colonna della matrice rappresenta una parola e ogni campione del testo è una riga della matrice. Il valore di ogni cella dunque non è altro che il numero di volte che in quella frase la parola appare. Il TfidfTransformer trasforma questa matrice in rappresentazione tfidf normalizzata.

$tf - idf$ sta per ”term-frequency inverse document-frequency”. L’obbiettivo di usare $tf - idf$ invece delle frequenze raw delle occorrenze di un token in un dato documento è quello di ridurre l’impatto di token che appaiono con frequenze

elevate nel documento e che sono dunque empiricamente meno informativi di altre parole del testo.

La formula per calcolare la $tf - idf$ di un termine t in un documento d è:

$$tf - idf(t,d) = tf(t,d) * idf(t) \tag{13}$$

con

$$idf(t) = \log \frac{n}{(df(t)+1)} + 1 \tag{14}$$

dove n è il numero totale di documenti nell'insieme e $df(t)$ è la frequenza dei documenti che contengono t. L'aggiunta del termine $+1$ fa si che non vengano completamente ignorati i termini con $idf(t) = 0$.

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectr = TfidfVectorizer(ngram_range=(1,2), min_df=1)
vectr.fit(dataset["text"])
vect_X = vectr.transform(dataset["text"])
```

Possiamo procedere finalmente alla fase di creazione del modello. Siccome si tratta di attribuire un punteggio di $+1$ ad un nodo considerato liberale e -1 ad un nodo considerato conservatore è chiaro che sarà necessario impiegare un algoritmo di classificazione binaria. Presenteremo dunque nelle successive sezioni i risultati di tale modello sfruttando diversi algoritmi di classificazione.

Il test set impiegato sarà uguale per tutti e ottenuto mediante il seguente codice:

```
orientation_test = []
text_test = []

#RANGE IMPONE DIMENSIONE TRAINING SET. 10K MIGLIORE.
for i in range(150000,222000):
  if ideology[i] >=0.5:
    orientation_test.append(1)
    text_test.append(raw[i])
  elif ideology[i] <= -0.5:
    orientation_test.append(-1)
    text_test.append(raw[i])
```

4.2 Regressione Lineare

La **Logistic Regression** è un tipo di **Modello Lineare Generalizzato** che utilizza un approccio non Bayesiano alla classificazione binaria. Infatti invece di ipotizza-

re una distribuzione a priori e calcolare la distribuzione dei parametri, si suppone immediatamente che la probabilità di appartenere ad una classe dato l'elemento $\vec{x_i}$ abbia la seguente forma:

$$p(C_k|\vec{x_i}) = \sigma(\vec{w}^T \vec{x_i}). \tag{15}$$

Abbiamo dunque per l'implementazione in Python di tale modello:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
clf = model.fit(vect_X,dataset["orientation"])
```

Nel test set il modello con questo algoritmo di classificazione predice correttamente l'Orientation del 70.65% dei nodi.

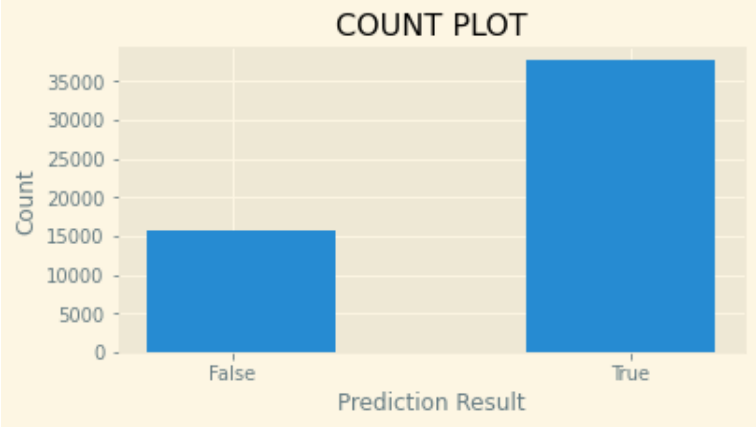

Figura 18: True/False prediction histogram per Linear Regression

4.3 Support Vector Machine

Support Vector Machine è una classe di algoritmi di classificazione più flessibili in quanto possono effettuare classificazione lineare sfruttando funzioni non lineari come funzioni base. Con la nostra implementazione impieghiamo un classificatore lineare per "fittare" un iperpiano che separa i dati in due classi.

```
from sklearn import svm
SVM = svm.LinearSVC()
SVM.fit(vect_X,dataset["orientation"])
```

Nel test set il modello con questo algoritmo di classificazione predice correttamente l'Orientation del 70.63% dei nodi.

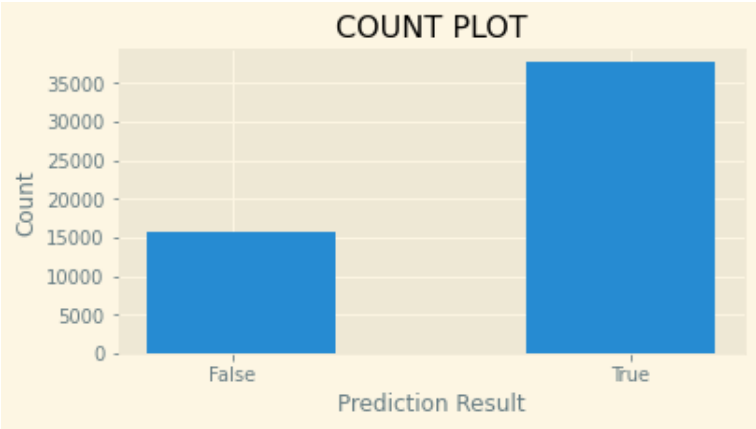

Figura 19: True/False prediction histogram per Support Vector Machine

4.4 Random Forests

Random Forests sono un insieme di modelli di apprendimento che ”fittano” molteplici ”Alberi Decisionali” in sottoinsiemi del dataset impiegato per il training e si effettua poi la media dei risultati.

```
from sklearn.ensemble import RandomForestClassifier
RF = RandomForestClassifier(n_estimators=100, max_depth=2,
    random_state=0)
RF.fit(vect_X,dataset["orientation"])
```

Nel test set il modello con questo algoritmo di classificazione predice correttamente l’Orientation del 73.60% dei nodi.

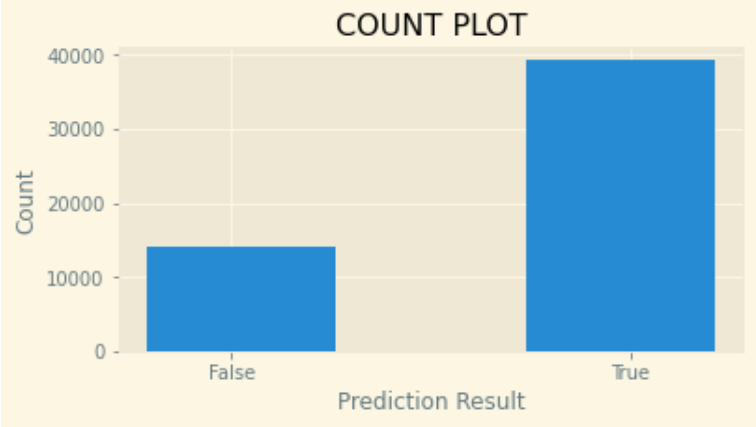

Figura 20: True/False prediction histogram per Random Forest

4.5 Neural Networks

Le **Neural Networks** sono un algoritmo di machine learning che include il fitting di numerosi *Hidden Layers* impiegati per rappresentare i neuroni e che sono collegati tra di loro con delle *Synaptic Activation Functions*. Questo modello consiste essenzialmente in una modellizzazione estremamente semplice del cervello. Per mantenere consistenza con gli altri modelli impiegati in questo studio abbiamo nuovamente impiegato la libreria *sklearn* anche se per questo tipo di modelli ce ne sono di più indicate come *TensorFlow*, *PyTorch* e *Keras*.

```
1 from sklearn.neural_network import MLPClassifier
2 NN = MLPClassifier(solver="lbfgs", alpha=1e-5,
     hidden_layer_sizes=(5,2), random_state=1)
3 NN.fit(vect_X,dataset["orientation"])
```

Nel nostro caso abbiamo usato 2 hidden layers, il primo con cinque neuroni ed il secondo con due.

Nel test set il modello con questo algoritmo di classificazione predice correttamente l'Orientation del 73.59% dei nodi.

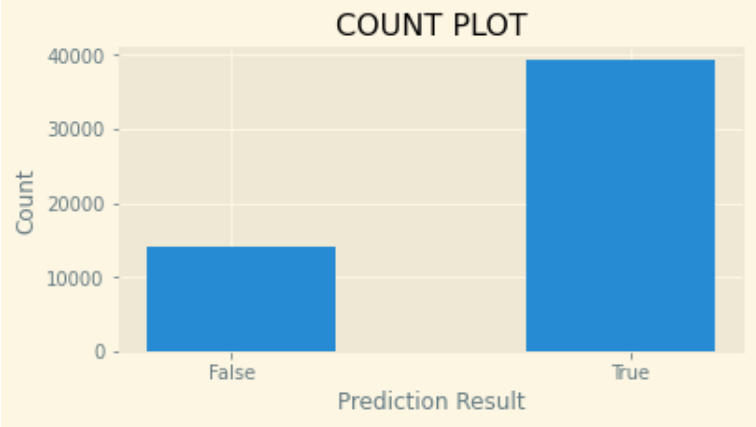

Figura 21: True/False prediction histogram per Neural Networks

5 Conclusione

Ricapitolando, dopo aver effettuato un'analisi dettagliata dei dati da noi impiegati, abbiamo elaborato e studiato a fondo un modello matematico che, come abbiamo potuto vedere, è in grado di emulare con sufficiente precisione uno scenario di polarizzazione come quello che abbiamo potuto osservare nei dati, inoltre si è notato come l'aggiunta di un solo parametro abbia permesso di migliorare la corrispondenza con caratteristiche secondarie dei dati. Abbiamo poi illustrato diverse applicazioni di machine learning a questi stessi dati mostrando come, con un sufficiente insieme di training, sia possibile sviluppare un modello in grado di predire l'"Orientation" di un utente "originale" con una buona percentuale di precisione.

Un ulteriore spunto per continuare l'analisi del nostro modello sarebbe quello di sostituire il modello di Ising alla base con un modello di Heisenberg, considerando dunque gli stati di spin, o nel nostro caso l'"Orientation" dell'utente, come un vettore e non come uno scalare. Studi preliminari di quest'implementazione sono stati effettuati e come possibile vedere in figura. I risultati non sono però quelli sperati. Si pensa di poter arrivare ad una versione con risultati accettabili incrementando il numero di dimensioni su cui considerare i nostri vettori di "Orientation" ma tale implementazione richiederebbe l'allocazione di risorse dal punto di vista computazionale a noi non accessibili.

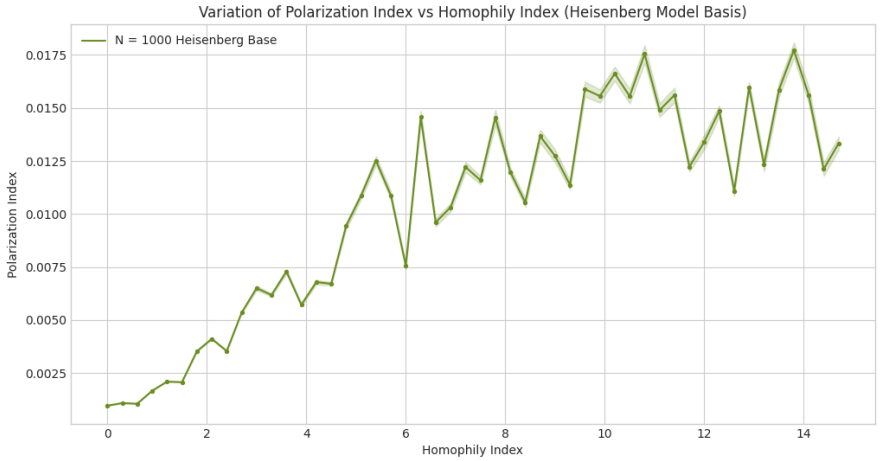

Figura 22: Andamento polarizzazione per 10mila nodi con Modello di Heisenberg come base.

Altrettanto promettente sarebbe lo studio degli stessi utenti Twitter in diversi periodi dell'anno in cui avvenivano nel mondo eventi con grande impatto dal punto di vista mediatico (Presidenziali 2020, Covid Outbreak, Guerra in Ucraina) per vedere come poter implementare l'azione di un campo esterno nel modello di Ising nel nostro modello. Tuttavia questo tipo di analisi richiederebbe accesso a dei dati su Twitter a noi non accessibili.

Riferimenti bibliografici

[Brady et al., 2017] Brady, W., Wills, J., Jost, J., Tucker, J., and Van Bavel, J. (2017). Emotion shapes the diffusion of moralized content in social networks. *Proceedings of the National Academy of Sciences*, 114:201618923.

[Huang, 1987] Huang, K. (1987). *Statistical mechanics*. Wiley, 2nd ed edition.

[Ising, 1925] Ising, E. (1925). Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift fur Physik*, 31(1):253–258.

[Kernighan and Lin, 1970] Kernighan, B. W. and Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307.

[Lambiotte and Kosinski, 2014] Lambiotte, R. and Kosinski, M. (2014). Tracking the digital footprints of personality. *Proceedings of the IEEE*, 102(12):1934–1939.

[Lazer et al., 2009] Lazer, D., Pentland, A., Adamic, L., Aral, S., Barabasi, A.-L., Brewer, D., Christakis, N., Contractor, N., Fowler, J., Gutmann, M., Jebara, T., King, G., Macy, M., Roy, D., and Van Alstyne, M. (2009). Social science. computational social science. *Science (New York, N.Y.)*, 323:721–3.

[Miller, 2011] Miller, G. (2011). Social scientists wade into the tweet stream. *Science (New York, N.Y.)*, 333:1814–5.

[Weinberger, 2011] Weinberger, S. (2011). Web of war. *Nature*, 471:566–8.