Alma Mater Studiorum
Università di Bologna
Campus di Cesena

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
Corso di Laurea Triennale in Ingegneria e Scienze Informatiche

APPROCCI COMPUTAZIONALI AL CALCOLO COMBINATORIO E DELLA PROBABILITÀ

Elaborato in:
Matematica discreta e probabilità

Relatore:
Prof Fabrizio Caselli

Presentata da:
Michele Monti

Sessione IV
Anno Accademico 2020-2021

Per Simone,
amico e compagno di studi
nel chiudere questo capitolo ti porto con me

Indice

Elenco delle figure

Introduzione

La difficoltà nello studio del calcolo combinatorio e della probabilità è una condizione diffusa all'interno del corso di ingegneria e scienze informatiche. Parte del problema è sicuramente l'approccio concreto condiviso da molti studenti, il quale si scontra con l'aspetto teorico di questa materia.
Per questa forma mentis, lo studente è portato a dubitare di molte regole e teoremi (specie se contro intuitivi) anche dopo averne visto la dimostrazione. Ciò che questo tipo di studente trova più convincente è invece la prova pratica.

La realizzazione di questo progetto parte da quest'idea: fornire agli studenti dimostrazioni pratiche attraverso la simulazione di contesti reali, offrendo la possibilità di confrontare i risultati di dette simulazioni con quelli enunciati nei teoremi.
A tale scopo, una parte importante del lavoro è stata la realizzazione di grafici chiari ed esaustivi, che permettano di confrontare i risultati ottenuti con quelli attesi in maniera rapida ed intuitiva.
Ciò non di meno, la realizzazione di alcune delle simulazioni ha comportato delle sfide tecniche nel produrre e maneggiare grosse moli di dati, nell'utilizzo di dataset di dati reali e nell'aspetto presentazionale dei risultati.
Speriamo, attraverso la consultazione dell'elaborato ana-

lizzato di seguito, di semplificare lo studio ad alcuni studenti, aiutarli ad interiorizzare concetti basilari e non, fornendogli uno strumento per studiare più adatto a loro.

Struttura della tesi. La tesi comincia esponendo le tecnologie utilizzate. Segue la presentazione del progetto stesso che si articola nei seguenti argomenti: Variabili aleatorie discrete, Distribuzione uniforme e non uniforme, Densità binomiale, Densità ipergeometrica, Densità geometrica, Densità geometrica modificata, Densità di Poisson, Densità continua uniforme, Densità continua esponenziale e Densità continua normale. Ognuno dei suddetti argomenti sarà presentato partendo da una veloce panoramica della teoria matematica che sta alla base, seguita da una veloce spiegazione della simulazione corredata dai grafici prodotti (quando presenti). Infine si passa alle conclusioni, che riassumono il contributo di questa tesi.

Capitolo 1

Tecnologie utilizzate

Vista la finalità stessa della tesi (semplificare il processo di apprendimento agli studenti), abbiamo ritenuto di primaria importanza la leggibilità del codice. Per questo motivo la scelta è ricaduta sul Python, in quanto abbiamo ritenuto che la sua leggibilità avrebbe reso più facile allo studente capire la simulazione e, di conseguenza, fidarsi dei risultati.
Oltre a ciò, un altro punto a favore del Python è stata l'ampia scelta di librerie di cui si fa largo uso in questo progetto, sia per la gestione dei dati, sia per la realizzazione delle soluzioni grafiche.
Di seguito riportiamo una panoramica delle librerie utilizzate e della loro implicazione.

1.1 NumPy

Numpy è una libreria open source nata nel 2005 e molto popolare nella comunità di Python. Il suo impiego in questo progetto riguarda la realizzazione e la gestione della quasi totalità delle strutture dati che sono state necessarie per il funzionamento delle simulazioni e dei grafici che le corredano.
Viene inoltre fatto uso di alcuni metodi per il calcolo

di alcuni dei risultati e per la generazione dei risultati casuali usati nelle simulazioni con variabili discrete (es. lancio di un dado, lancio di una moneta, partita alla roulette, estrazione di una o più biglie da un'urna). Per quanto riguarda quest'ultimo utilizzo è importante specificare che Numpy fa utilizzo del Mersenne Twister come generatore di numeri pseudo-casuali (di questo generatore parleremo più avanti) [1].

1.2 Ipywidgets

Ipywidgets, conosciuto anche come jupyter-widgets, è una libreria che si propone di dare vita ai notebook come quello realizzato con questa tesi. La libreria offre metodi per dare all'utente il controllo dei dati, permettendogli di cambiarli facilmente. Attraverso l'uso di questa libreria si spera di rendere l'apprendimento un'esperienza immersiva.
Nel progetto qui descritto è fatto uso di due metodi di questa libreria:

- interact
- `interact_manual`

entrambi al fine di offrire la possibilità di modificare i dati in alcuni esercizi. Ove possibile, si è preferito usare interact, ma negli esercizi la cui simulazione comportava un tempo di computazione più lungo è stato necessario utilizzare interact_manual per evitare che la simulazione venisse eseguita su più istanze indesiderate con lunghi tempi di attesa per l'utente [2].

1.3 Matplotlib

Matplotlib è una libreria utile per realizzare grafici efficaci, offrendo una visualizzazione dei risultati veloce e molto adattabile al campo di utilizzo. Con l'ausilio di questa libreria abbiamo realizzato ogni sorta di grafico, dal grafico a torta, agli istogrammi, agli andamenti delle densità continue [3].

1.4 Math

La libreria Math, come suggerisce il nome, dà accesso a tutte le funzioni matematiche più importanti.
In questo progetto viene impiegata per avere buone approssimazioni di numeri irrazionali, come il numero di Nepero, e per calcolare rapidamente logaritmi e fattoriali [4].

1.5 Seaborn

Seaborn è una libreria per la visualizzazione di dati che si basa sul modello della più famosa matplotlib e pensata per funzionare bene in sinergia con le strutture dati tipiche di Pandas.
Questa libreria fornisce un'interfaccia ad alto livello per disegnare grafici accattivanti.
In quanto libreria di alto livello, è pensata per semplificare la realizzazione di grafici complessi, composti da più elementi, e per tale fine è strutturata in modo tale che chi ne fa uso possa concentrarsi più sull'assemblamento dei vari elementi che compongono il grafico, piuttosto che come disegnarli singolarmente [5].

1.6 Random

Com'è facile immaginare, in una simulazione di un problema di probabilità, la fonte della casualità gioca un ruolo fondamentale per la buona riuscita dell'esperimento. La libreria Random è tra le più usate per generare numeri pseudo-casuali, sia per distribuzioni uniformi, sia per distribuzioni normali, logaritmiche o di altro genere. Il funzionamento di base, comune alla maggior parte dei moduli di questa libreria, consiste nella generazione di un numero compreso nell'intervallo [0.0, 1.0) attraverso l'utilizzo del Mersenne Twister come generatore (seguiranno più dettagli su questo generatore) [6].

1.7 SciPy

SciPy è una libreria Open Source, sviluppata ed aggiornata pubblicamente su GitHub.
Questa libreria che nasce per estendere la più diffusa NumPy (citata sopra), aggiungendo strumenti per la computazione di liste, ed offrendo strutture dati specializzate come matrici sparse ed alberi n-dimensionali.
La libreria offre inoltre un'ampia gamma di algoritmi per l'ottimizzazione, l'integrazione e l'interpolazione di equazioni algebriche, differenziali, statistiche e molte altre classi di problemi.
Nonostante la tipologia di problemi che SciPy si propone di risolvere sia caratterizzata da un alto costo computazionale, i tempi di esecuzione dei metodi di questa libreria si mantengono bassi, grazie all'alto grado di ottimizzazione ottenuto dalla loro implementazione in linguaggi a basso livello, tra i quali troviamo il C, C++, e Fortran [7].

1.8 Pandas

Pandas è una delle librerie più popolari nel campo dell'analisi dei dati.
Lo sviluppo di questa libreria inizia nel 2008 a cura della AQR Capital Management e dal 2009 è diventata Open Source. Pandas offre gli strumenti per la lettura, scrittura e modifica di alcune strutture dati dei formati tra i più diffusi. La libreria rende facile utilizzare dataset di grandi dimensioni in maniera efficiente.
Per ottenere le performance che offrono, i metodi di questa libreria sono ottimizzati con un'implementazione in C delle sezioni critiche e che potrebbero incidere negativamente sui tempi di esecuzione [8].

1.9 Mersenne Twister

Come accennato in precedenza, in questo elaborato si fa indirettamente uso del generatore di numeri pseudocasuali noto come Mersenne Twister.
Vista l'importanza della casualità in questa tesi, riteniamo opportuno soffermarci sul generatore che provvede a buona parte dei dati usati negli esercizi, specialmente nella parte riguardante le densità continue.
Il Mersenne Twister è un generatore di numeri pseudocasuali sviluppato nel 1997 da Makoto Matsumoto e da Takuji Nishimura.
Il suo nome deriva dalla lunghezza del periodo che viene scelto essere un numero primo di Mersenne. Un numero N è definito primo di Mersenne se soddisfa le due seguenti condizioni:

- è un numero primo (divisibile solo per sè stesso e per 1)

- N è uguale a $2^m - 1$ per un valore di m intero e positivo

La versione più diffusa del generatore è basata sul numero primo di Mersenne $2^{19937} - 1$, genera quindi un numero appartenente all'intervallo $[0.0, 1.0)$ con un periodo di $2^{19937} - 1$.
Il Mersenne Twister è il generatore di numeri pseudocasuali usato di default in diversi linguaggi tra i quali anche Python e Ruby, in diverse librerie Linux ma anche da Microsoft Excel e MATLAB [9].

1.10 Scelta della piattaforma

Vista la finalità del progetto, ovvero essere condiviso con gli studenti come strumento per lo studio, abbiamo scelto di realizzarlo su Colaboratory (che da questo momento chiameremo Colab).
Colab permette infatti di eseguire Python direttamente dal browser senza bisogno di alcuna configurazione, in questo modo basterà avere il link per poter consultare il materiale.

Colab permette inoltre di integrare il codice a celle di testo, in questo modo è stato semplice introdurre i vari argomenti con la dovuta spiegazione teorica, l'esposizione degli enunciati, le equazioni e quanto necessario ad una corretta e completa esposizione dell'argomento prima di procedere con le simulazioni e gli esercizi [10].

1.11 Spyder

Spyder è un ambiente di sviluppo open source e multi piattaforma per l'implementazione di codice Python.
Nello sviluppo dell'elaborato di tesi abbiamo fatto uso anche di Spyder come ambiente di sviluppo.
Sebbene Colab fornisca la possibilità di scrivere ed eseguire codice, gli strumenti di debugging che offre sono molto limitati. Per questo motivo, nello sviluppo di alcune delle simulazioni più complesse, ci siamo serviti di questo ulteriore aiuto [11].

1.12 GitHub

Per gestire le versioni del progetto ci siamo appoggiati a GitHub.
Questa è una piattaforma cloud-based per lo sviluppo ed il mantenimento del codice, oltre che per tenere uno storico delle modifiche.
Utilizzando GitHub abbiamo diviso lo sviluppo del progetto in due rami: da un lato testavamo le funzionalità, scrivevamo diverse versioni di uno stesso esercizio o simulazione e controllavamo cosa funzionava e cosa no; sul secondo ramo abbiamo costruito mano a mano il documento finale, aggiungendo di volta in volta la versione della simulazione che più ci aveva convinto tra quelle testate sul primo ramo [12].

Capitolo 2

Implementazione

Le simulazioni implementate nel progetto sono pensate per riprodurre, passo per passo, situazioni reali descritte dagli esercizi visti a lezione. Per fare questo è stato necessario ricorrere ad alcuni espedienti che analizzeremo in dettaglio in questa sezione della tesi.
Una volta ottenuto il risultato della situazione simulata, l'esperimento viene ripetuto un numero variabile di volte continuando a raccogliere i risultati. Alla fine viene fatto un conteggio dei singoli risultati, per verificare percentualmente quante volte si sono verificati e le percentuali ottenute vengono confrontate con quelle calcolate con la formula.

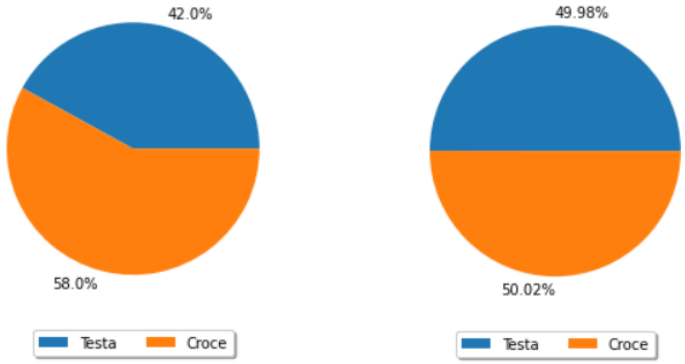
Scopo della simulazione è mostrare la somiglianza tra i risultati ottenuti e quelli attesi, ponendo l'accento sul fatto che ad un maggior numero di volte in cui viene eseguito l'esperimento, si associa immancabilmente un margine di errore più basso.
Si spera che lo studente, messo di fronte a questa evidenza, si convinca più facilmente di alcune delle regole meno intuitive e sia semplificato nel ricordarle.

2.1 Introduzione

L'elaborato di tesi si apre con una breve introduzione in più punti.
Siccome la finalità della tesi è rendere più intuitivi alcuni aspetti della teoria del calcolo combinatorio e probabilistico vedendoli applicati in delle simulazioni, appare chiara l'importanza di un codice comprensibile a tutti. Per questo il primo punto dell'introduzione dà una breve panoramica del Python, un linguaggio di per sè altamente leggibile, soffermandosi su alcune pratiche peculiari a Python. In questo modo si spera che anche gli studenti che non dovessero avere familiarità col linguaggio, non avranno una visione opaca del codice.

Seguono delle funzioni che sono state introdotte all'inizio del documento perchè comuni a più argomenti, ma che vedremo in dettaglio più avanti.
Infine, l'introduzione si chiude con uno sguardo all'interpretazione dei risultati. Viene fatto l'esempio banale del lancio di una moneta. A tutti è noto il risultato atteso di un singolo lancio: esattamente il 50% di probabilità che esca testa, ed il 50% che esca croce.

Per dimostrare la necessarietà di ripetere l'esperimento un congruo numero di volte, viene mostrata la distribuzione dei risultati simulando sul momento 100 lanci.



(a) 100 lanci (b) 1 000 000 di lanci

Figura 2.1: Distribuzione dei risultati di una serie di lanci di una moneta

Nonostante 100 possa sembrare un buon numero di lanci, dal punto di vista statistico è inadeguato ed i risultati che ne ricaviamo sono inconcludenti (in figura vediamo come in una serie di 100 lanci abbiamo ottenuto croce 58 volte). Per confronto viene ripetuto l'esperimento lanciando la moneta un milione di volte e vediamo che in questo caso i risultati ottenuti sono molto più vicini ad un 50 e 50.

2.2 Variabili aleatorie discrete

Conclusa l'introduzione, cominciamo definendo la prima classe di variabili che analizzeremo: le variabili aleatorie discrete.

"Una variabile aleatoria X si dice discreta se i valori che assume sono finiti oppure numerabili."

Sono due gli esempi di variabili aleatorie discrete che riportiamo, entrambi basati sul gioco della roulette. Prima di introdurli, vediamo il primo espediente che abbiamo

usato per simulare una roulette. Abbiamo simulato la roulette generando un array di 37 elementi, come i numeri della roulette (da 0 a 36), ed abbiamo assegnato a 18 di questi elementi il caso di vittoria (negli esercizi proposti le puntate sono sempre fatte solo su un colore), con una distribuzione simile a quella di una vera roulette.

```
def mk_roulette():
    ruota = np.zeros(37, dtype=np.bool_)
    ruota[1::2] = True
    return ruota
```

Gli esercizi che incontreremo più avanti, anche se non si baseranno sul gioco della roulette, avranno un livello di astrazione paragonabile a questo.
Passando agli esempi delle variabili aleatorie discrete, il primo chiede, giocando 3 puntate da 10 euro sul rosso, qual è la probabilità di vincere più di X con $0 \leq X \leq 30$. Essendo le vincite possibili dei valori finiti, la variabile è discreta.

Simile è l'esempio successivo, nel quale ci interroghiamo su quale sia la vincita attesa dato il numero di partite alla roulette e la puntata. Ancora una volta simuliamo la roulette con l'astrazione vista sopra, giochiamo più volte le partite richieste dall'esercizio e raccogliamo i risultati. Essendo le vincite (o in questo caso le perdite) un valore finito e numerabile (in euro ed in centesimi), la variabile è discreta.

2.3 Distribuzione uniforme e non uniforme

Come ultima premessa, definiamo la differenza tra una distribuzione uniforme ed una non uniforme.
Sia $A = x_1, ..., x_n$; una variabile X che assume i valori

in A tutti con la stessa probabilità $\frac{1}{n}$ si dice variabile uniforme su A. Scriviamo in questo caso $X \sim U(A)$, e la densità è:

$$p_X(h) = \begin{cases} \frac{1}{n} \; se \; h \in \; (x1, x2, ..., xn) \\ 0 \; altrimenti \end{cases} \tag{2.1}$$

Come esempio di distribuzione uniforme abbiamo proposto i risultati del lancio di un dado, nell'esempio implementato lanciamo un dado 100 000 volte e raccogliamo i risultati per produrre il seguente grafico:

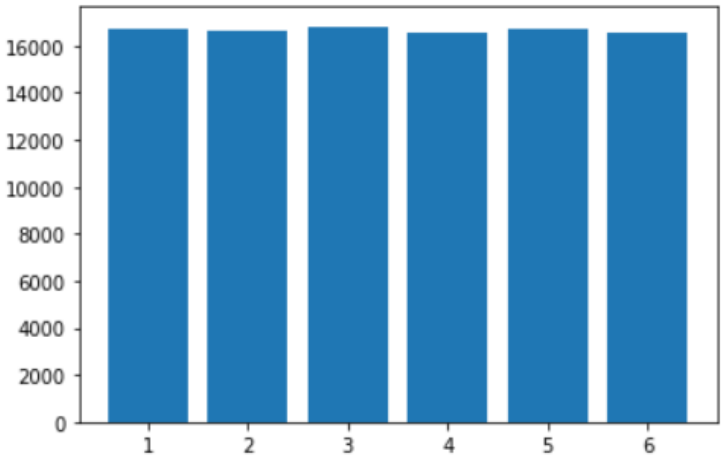


Figura 2.2: Risultati di 100 000 dadi

Per la distribuzione non uniforme abbiamo continuato a lavorare coi dadi. Questa volta l'esempio propone di lanciarne due, la variabile K assume un valore uguale al numero di risultati pari ottenuti.
La densità di questa variabile è:

$$p_Y(k) = \begin{cases} \frac{1}{4} \; k = 0, 2 \\ \frac{1}{2} \; k = 1 \\ 0 \; altrimenti \end{cases} \tag{2.2}$$

Essa viene confermata dall'esperimento, che ripete il lancio dei due dadi fino ad ottenere questa distribuzione:

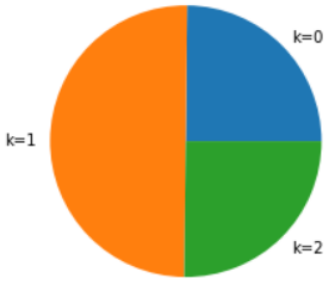


Figura 2.3: Distribuzione del numero di risultati pari dal lancio di 2 dadi

2.4 Densità binomiale

Molti problemi nell'ambito del calcolo della probabilità sono riducibili a schemi di successo-insuccesso.
La densità ideale per descriverli è la binomiale, ma iniziamo provando a capire cosa sono.
Parliamo di schema successo-insuccesso quando la variabile aleatoria che stiamo studiando può dare solo due possibili risultati. Il caso più intuitivo a cui ciò può essere applicato è quello di una moneta (la quale può dare solo testa o croce); in questo caso diremmo che quando si verifica uno dei due eventi si tratta di un successo, mentre quando si verifica l'evento opposto è un insuccesso.
Poniamo ora che le prove siano indipendenti tra di loro, ovvero che i risultati delle prove precedenti non cambino le condizioni delle prove successive; in questo caso parliamo di schema successo-insuccesso a prove indipendenti.
Data la variabile X data dal numero di successi su n tentativi, nel caso di prove indipendenti la probabilità che X assuma un certo valore dipende solo dal numero di successi desiderati, non dall'ordine in cui si presentano. La

probabilità quindi di avere k successi ed $n-k$ insuccessi in un ordine prefissato è data dalla formula $p^k(1-p)^{n-k}$, dove p corrisponde alla probabilità che un singolo evento risulti in un caso di successo. Siccome possiamo scegliere le k posizioni per i successi in $\binom{n}{k}$ modi, la densità della variabile X è definita come segue:

$$p_X(k) = \begin{cases} \binom{n}{k} p^k (1-p)^{n-k} & se\ k = 0, 1, 2, ..., n \\ 0 & altrimenti \end{cases} \tag{2.3}$$

X è una variabile binomiale e la formalizzeremo come $X \sim B(n, p)$.
Per l'esempio relativo alla densità binomiale troviamo nuovamente la roulette. Se supponiamo di giocare n partite facendo sempre la stessa puntata, la probabilità di fare una certa vincita può essere vista come la probabilità di ottenere i k successi che pagherebbero quella cifra. Se quindi, per esempio, ipotizzassimo di andare al casinò con 15 euro e di fare 3 puntate da 5 euro l'una, la probabilità di uscire dal casinò con 0, 10, 20 o 30 euro corrisponde alla probabilità di ottenere 0, 1, 2 o 3 successi.

L'esempio proposto fa esattamente questo, gioca le 3 puntate ed aggiunge ai risultati con quanti soldi si è usciti dal casinò, ripetendo l'esperimento molte volte. Applicando la formula sappiamo che la distribuzione della probabilità in questo momento è calcolata come:

$$P(X = 30) = (\tfrac{18}{37})^3 = 0,115$$

$$P(X = 20) = \binom{3}{1}(\tfrac{18}{37})^2 \tfrac{19}{37} = 0,365$$

$$P(X = 10) = \binom{3}{2}(\tfrac{18}{37})(\tfrac{19}{37})^2 = 0,385$$

$$P(X = 0) = (\tfrac{19}{37})^3 = 0,135$$

Andando a guardare i risultati di un gran numero di partite (simulate similmente agli esempi precedenti sulla roulette), il grafico prodotto è il seguente:

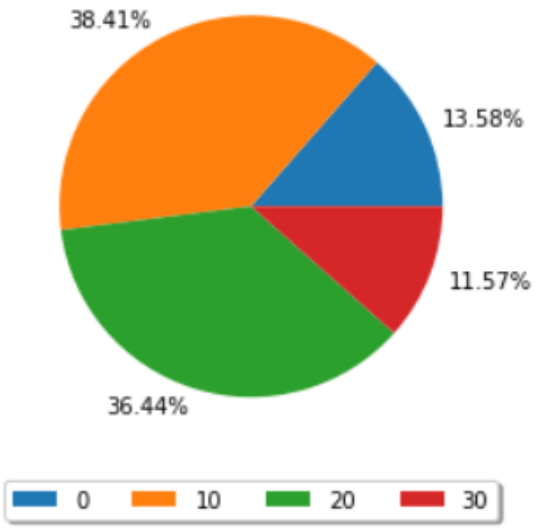


Figura 2.4: Media dei soldi rimasti dopo 3 puntate da 5 euro alla roulette (ripetuto 100 000 volte)

Altro elemento importante in questa tesi era l'interattività; in questo esempio si lascia all'utente la possibilità di intervenire sul numero di giocate e sulla puntata, producendo un grafico differente ma ugualmente curato.

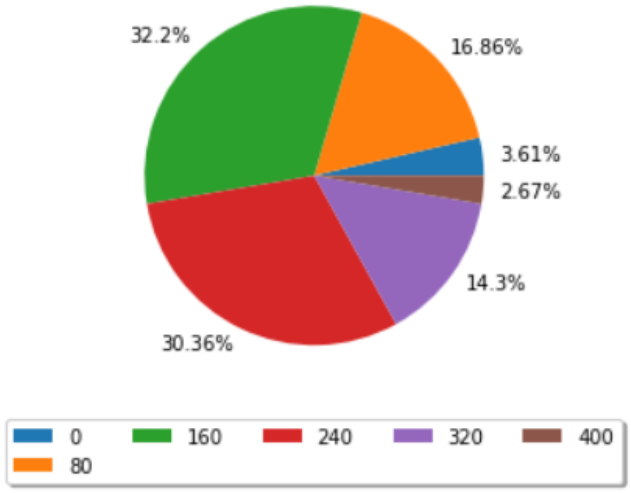


Figura 2.5: Media dei soldi rimasti dopo 5 puntate da 40 euro alla roulette (ripetuto 100 000 volte)

2.5 Densità ipergeometrica

Quando le prove di uno schema successo-insuccesso non sono indipendenti tra loro abbiamo una densità diversa. Prendiamo ad esempio il caso di un'urna contenente b palline bianche ed r palline rosse e dalla quale vogliamo estrarre n palline senza reinserimento. Ogni volta che estraiamo una pallina bianca lo consideriamo un caso di successo, viceversa ad ogni rossa estratta avremo un insuccesso. Data X la variabile che assume un valore pari al numero di successi, la probabilità di $X = k$ è dato da tutti i sottoinsiemi delle $b + r$ palline presenti nell'urna costituiti da k palline bianche ed $r - k$ palline rosse.
Siccome le k bianche possono essere scelte in $\binom{b}{k}$ modi e le $r - k$ rosse in $\binom{r}{n-k}$ modi, la densità di X è data da:

$$p_X(k) = \begin{cases} \frac{\binom{b}{k}\binom{r}{n-k}}{\binom{b+r}{n}} & se\ k = max(0, n-r), ..., min(n, b) \\ 0 & altrimenti \end{cases} \tag{2.4}$$

Chiameremo quindi la variabile X ipergeometrica e la scriveremo come $X \sim H(n; b, r)$.

La simulazione qui proposta vede un'urna contenente 8 palline bianche e 2 rosse. Se ci chiedessimo qual è la probabilità di estrarre al più una rossa effettuando 3 estrazioni, la variabile $X =$ "*numero di rosse estratte*" sarebbe una ipergeometrica $X \sim H(3; 2, 8)$ e si calcolerebbe quindi così:

$P(X \leq 1) = P(X = 0) + P(X = 1) = \frac{\binom{2}{0}\binom{8}{3}}{\binom{10}{3}} + \frac{\binom{2}{1}\binom{8}{2}}{\binom{10}{3}} = \frac{14}{15} = 0.933$

Similmente alla roulette, l'urna è simulata tramite un vettore lungo $b+r$ con b elementi con valore "False" ed r elementi con valore "True" distribuiti casualmente lungo il vettore.

```
def mk_urna(bianche, rosse):
    tot = bianche + rosse
    urna = np.zeros(tot, dtype=np.bool_)
    while(rosse > 0):
        rnd = np.random.randint(0, (tot))
        if (not urna[rnd]):
            urna[rnd] = True
            rosse = rosse - 1
    return urna
```

Per simulare le estrazioni senza rimpiazzo, eliminiamo 3 elementi casuali dal vettore contando poi quanti ele-

menti a "True" (o palline rosse) sono rimasti. Se ne è rimasta almeno una significa che ho estratto al più una rossa e conto la prova tra quelle "riuscite". Le prove riuscite diviso le prove totali, con un numero di prove sufficientemente grande, coincide con la probabilità calcolata precedentemente

```
for i in range(estrazioni):
        tmp = np.delete(tmp,
        np.random.randint(0, ((bianche + rosse)-i)))
    if(tmp.sum() > 0):
        results = results + 1
```

Allo studente è lasciata la possibilità di intervenire sul numero di estrazioni.

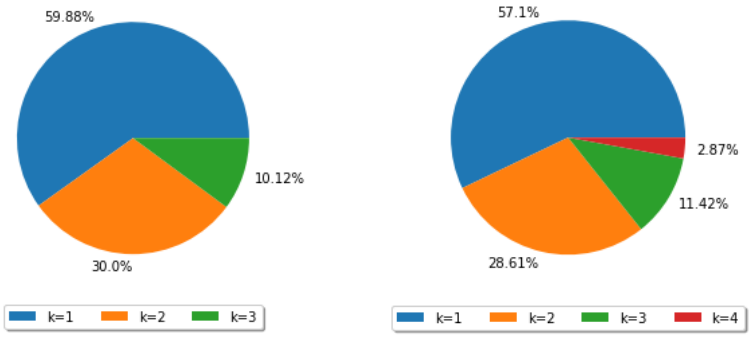
2.6 Densità geometrica e geometrica modificata

Proseguendo con lo scenario di successo-insuccesso, potremmo essere interessati non al numero di successi ottenuti sui tentativi eseguiti, ma al numero di tentativi che mi devo aspettare prima di ottenere un successo.
Per l'esempio proposto continuiamo a lavorare con l'estrazione di palline da un'urna: in questo caso avremo 2 palline bianche e 3 rosse e la variabile aleatoria X data dal numero di estrazioni per trovare la pallina rossa. La densità di X è facile da calcolare ed è data da

$$p_X(k) = \begin{cases} \frac{6}{10} \ se \ k = 1 \\ \frac{3}{10} \ se \ k = 2 \\ \frac{1}{10} \ se \ k = 3 \\ 0 \quad altrimenti \end{cases} \tag{2.5}$$

La simulazione ha un funzionamento del tutto simile a quella proposta per la densità ipergeometrica, ma questa volta si permette di modificare il numero di palline

rosse e quello di palline bianche. I grafici prodotti dalla simulazione sono di questo tipo:



(a) 2 bianche e 3 rosse (b) 3 bianche e 4 rosse

Figura 2.6: Grafici prodotti con diverse quantità di palline

Se però le palline venissero rimpiazzate, allora parleremmo di densità geometrica modificata. La variabile X diventa quindi numerabile e la densità è:

$$p_X(k) = \begin{cases} p(1-p)^{k-1} & se\ k = 1, 2, 3, ... \\ 0 & altrimenti \end{cases} \tag{2.6}$$

Ripetendo l'esperimento precedente con la novità del rimpiazzo si ha:

$$p_X(k) = \begin{cases} (\frac{2}{5})^{k-1}\frac{3}{5} & se\ k = 1, 2, 3, ... \\ 0 & altrimenti \end{cases} \tag{2.7}$$

Anche la simulazione rimane la stessa, ma non viene più scartato l'elemento estratto dal vettore di volta in volta.
Per dare una migliore prospettiva allo studente, in figura abbiamo deciso di affiancare il grafico prodotto dai risultati ottenuti tramite la simulazione, a quello definito dalla formula. Ancora una volta lo studente può modificare il numero delle palline di ogni colore.

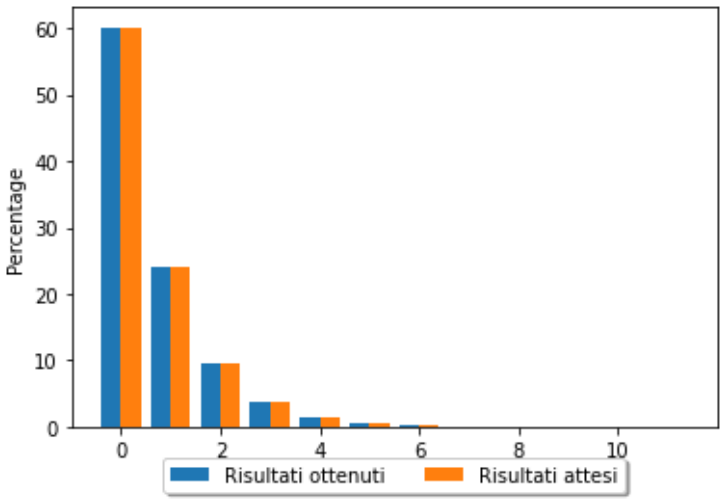


Figura 2.7: Grafici a confronto per le estrazioni da un'urna contenente 2 palline bianche e 3 rosse fino all'estrazione di una rossa

Abbiamo parlato della densità geometrica modificata, ma in cosa differisce dalla densità geometrica standard? Se la modificata modella la distribuzione di una variabile in cui ci si chiede la probabilità che il primo successo avvenga al k-esimo tentativo, la geometrica standard riguarda la probabilità di ottenere k insuccessi prima di un successo. In altre parole è una densità geometrica modificata a cui viene applicato uno slittamento di 1. La densità risultante è la seguente:

$$p(k) = \begin{cases} (\frac{2}{5})^k \frac{3}{5} & se\ k = 0, 1, 2, 3, ... \\ 0 & altrimenti \end{cases} \tag{2.8}$$

Per l'ultima simulazione di questo argomento abbiamo deciso di riproporre un esempio coi dadi. In questo caso lanciamo simultaneamente 2 dadi, uno rosso ed uno blu, ripetendo il lancio fino ad ottenere due 6. Ci chiediamo quindi quale sia la probabilità di ottenere un 6 col dado rosso prima di ottenerlo col dado blu.
Il funzionamento di questa simulazione somiglia agli esempi coi dadi già incontrati precedentemente e, come nel-

l'ultimo esercizio, mettiamo a confronto il grafico ottenuto dai risultati dei lanci simulati col grafico prodotto dalla teoria:

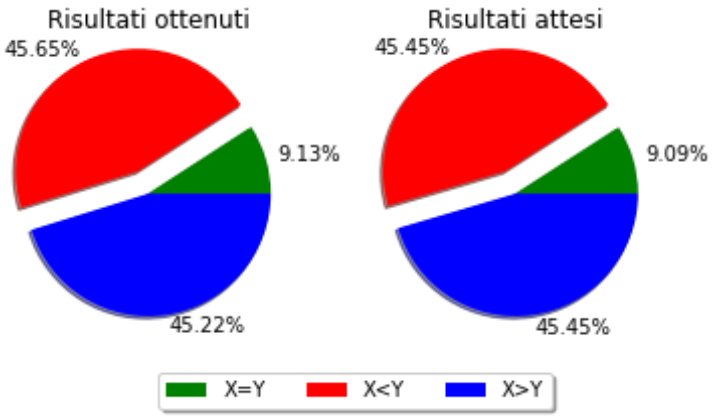


Figura 2.8: X è pari al primo lancio in cui il dado rosso ha dato 6, Y al primo lancio in cui il blu ha dato 6

2.7 Densità di Poisson

L'ultima densità che affrontiamo riguardante le variabili discrete è quella di Poisson.
Se ci trovassimo davanti una variabile binomiale $X \sim B(n, p)$ con n (numero di tentativi) molto grande ed il parametro p fosse molto piccolo, sviluppare il calcolo della binomiale sarebbe un processo molto lungo.
Supponiamo ad esempio che in una città nascano 10 000 bambini ogni anno e che un bambino ogni 2 000 sia affetto da una malattia rara. Se ci domandassimo la probabilità che quest'anno nascano più di 5 bambini affetti da questa malattia, potremmo pensare di risolverlo con una variabile binomiale. In questo caso la variabile X data dal numero di bambini malati nati quest'anno sarebbe $X = B(10^4, \frac{1}{2000})$, ma effettuare il calcolo $P(X > 5)$, pur avendo una formula esatta per farlo, sarebbe complesso. Per questo tipo di problema introduciamo la densità di

Poisson, definita come segue:

$$p(k) = \begin{cases} e^{-\lambda}\frac{\lambda^k}{k!} \; se \; k = 0, 1, 2, ... \\ 0 \qquad\quad altrimenti \end{cases} \tag{2.9}$$

con $\lambda > 0$.

Nonostante questa sia un'approssimazione, i risultati che produce sono molto simili a quelli che otterremmo dal lungo e tedioso sviluppo della formula della binomiale, la quale offrirebbe una soluzione esatta.

Per convincerci di questo introduciamo una simulazione. Il funzionamento è semplice: per ogni bambino nato quest'anno estraiamo se il bimbo in questione sarà malato (l'estrazione avrà esito positivo con probabilità di 1 su 2000). Una volta ripetuto il procedimento per tutti i bambini contiamo quanti risultano malati. Se l'esperimento fosse il lancio di una moneta questo sarebbe l'equivalente di lanciarla una volta. Diventa quindi necessario ripetere questo lavoro un congruo numero di volte.

Siccome il costo computazionale di questa operazione è molto alto, abbiamo deciso di renderlo più interattivo: la singola esecuzione del programma simula quindi solo 100 anni di nascite (per ridurre il tempo di esecuzione); rieseguendo il programma i nuovi risultati si aggiungono a quelli vecchi ottenendo una simulazione su un maggior numero di anni ad ogni esecuzione.

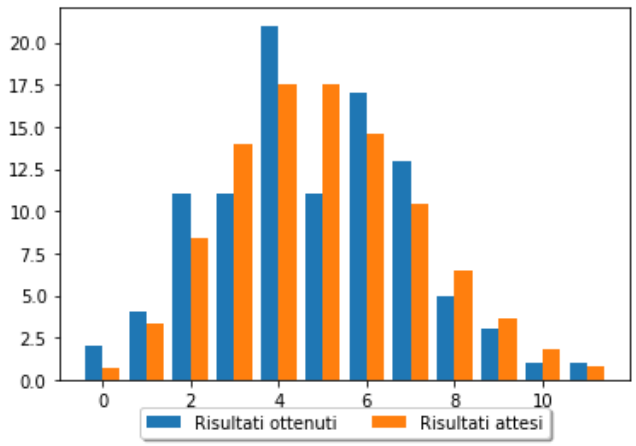


Figura 2.9: Poisson su 100 anni

Lo studente può così osservare che al crescere del numero di anni simulati, il grafico dei risultati effettivamente ottenuti, va via via avvicinandosi a quello prodotto dall'applicazione della formula di Poisson.

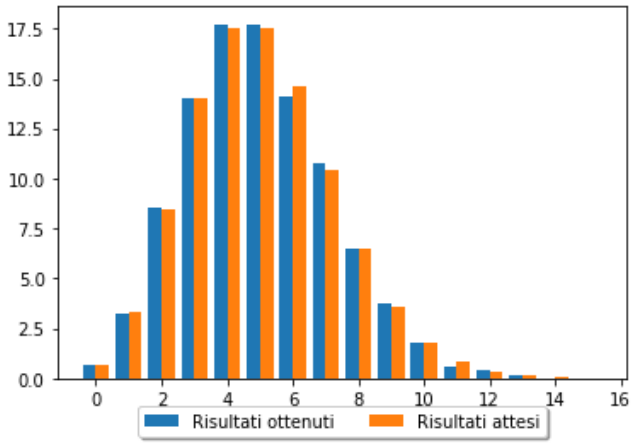


Figura 2.10: Poisson su 10 000 anni

2.8 Densità continua uniforme

Conclusa la parte sulle variabili discrete, si apre il capitolo sulle densità continue. Queste sono strettamente

non numerabili, come i valori all'interno di un intervallo di tempo. La prima densità che abbiamo affrontato è anche la più semplice tra le continue: la densità continua uniforme.
Una variabile si dice continua uniforme quando può assumere valori solo in un certo intervallo limitato ed in modo uniforme. L'esempio proposto parla di un autobus che passa da una certa fermata ogni 10 minuti, la variabile T data dal tempo di attesa è del tipo continua uniforme.
Per l'implementazione della simulazione, produrre i risultati risulta banale, ma per metterli a grafico abbiamo valutato 2 metodi. Il primo consisteva nell'utilizzo della libreria Seaborn (già descritta nel capitolo precedente) ed in particolare del metodo kdeplot(). Il metodo in questione fa utilizzo della stima kernel di densità, una tecnica non parametrica usata per stimare riconoscere i pattern e per stimare la densità in uno spazio numerico.

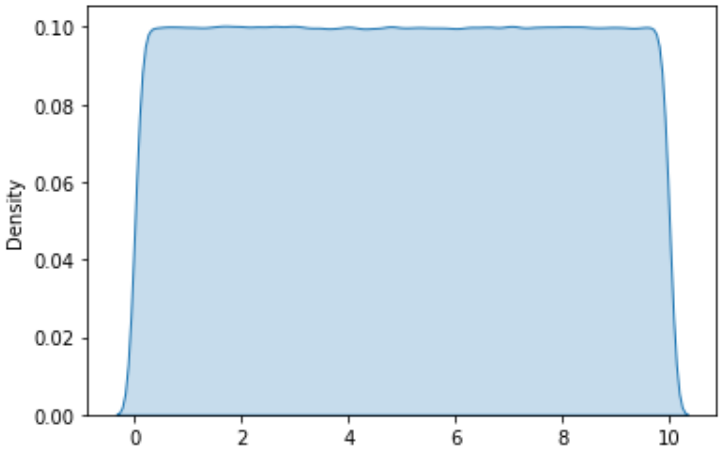


Figura 2.11: Densità continua uniforme stimata col metodo della stima kernel di densità

Per quanto elegante, questo metodo presentava un difetto evidente: partendo dal set di risultati calcolati, stimava possibile (anche se improbabile) l'esistenza di risultati esterni all'intervallo.
Per questo motivo abbiamo optato per lo sviluppo di un metodo dedicato, che stima la probabilità punto per punto contando i risultati in un intervallo di $+\varepsilon$ o $-\varepsilon$ rispetto al punto analizzato.

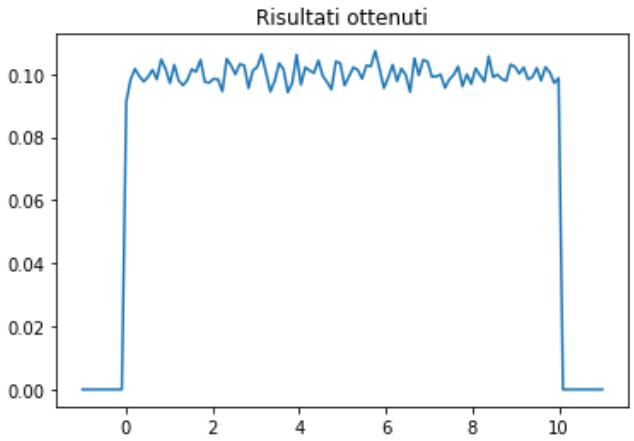
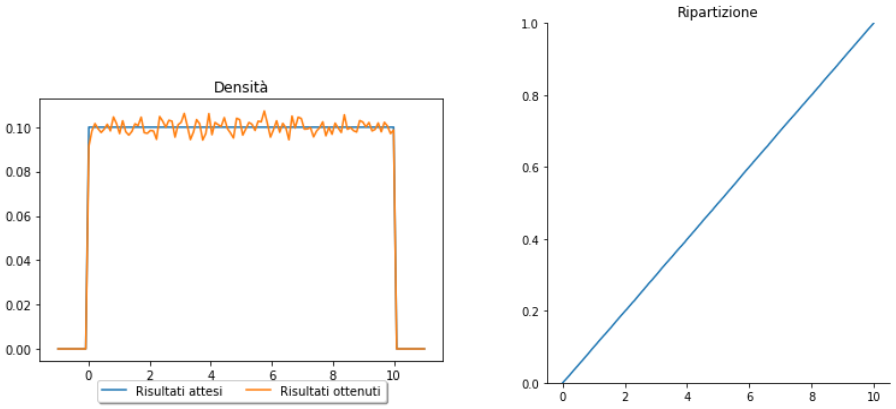


Figura 2.12: Densità continua uniforme stimata punto per punto

Tornando sul problema del bus, abbiamo ritenuto interessante mostrare allo studente i grafici delle densità (quello atteso e quello prodotto dalla simulazione) messi a confronto.
A questo abbiamo aggiunto il grafico della funzione di ripartizione, calcolato a partire dai risultati col metodo displot() della libreria Seaborn.



(a) Funzione della densità (b) Funzione di ripartizione

Figura 2.13: Densità continua uniforme - esempio del bus

Possiamo quindi farci 2 domande su questa variabile T:

- Mediamente quanto dovrò aspettare?
- Qual è la probabilità che il bus arrivi in un certo intervallo di tempo?

Per rispondere alla prima domanda facciamo una media tra tutti i tempi di attesa che abbiamo simulato e vediamo che il risultato coincide con quello calcolato con la formula $\frac{b+a}{2}$.
Per la seconda invece abbiamo proposto un esercizio in cui si lascia allo studente la possibilità di modificare gli estremi dell'intervallo. Sappiamo che la probabilità è data dall'ampiezza dell'intervallo scelto divisa per il tempo totale e contando i risultati ottenuti in quell'intervallo vediamo che la loro percentuale coincide con quella attesa.
Il grafico prodotto in questo esercizio evidenzia la zona del grafico sottesa dalla curva dei risultati che ci interessa.

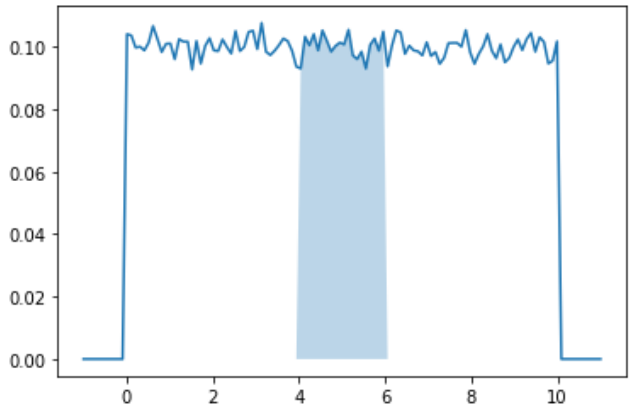


Figura 2.14: Densità continua uniforme nell'intervallo [4, 6]

2.9 Densità continua esponenziale

La seconda densità continua che abbiamo trattato è stata la esponenziale. Definita dalla densità

$$f(s) = \begin{cases} \lambda e^{-\lambda s} & se\ s \geq 0 \\ 0 & altrimenti \end{cases} \tag{2.10}$$

questo tipo di variabile è ideale per descrivere l'andamento di vari eventi naturali come il tempo di attesa per l'eruzione di un vulcano o la morte di un albero.
La sua funzione di ripartizione è data da

$$F_X(t) = \begin{cases} 1 - e^{-\lambda t} & se\ t \geq 0 \\ 0 & se\ t \leq 0 \end{cases} \tag{2.11}$$

L'esempio proposto per questa variabile descrive il decadimento di una particella radioattiva. Partendo dal presupposto che il decadimento di questa particella segua una legge esponenziale di parametro $\lambda = 1$ in giorni, ci chiediamo la probabilità che decada entro 3 ore.
Per simulare questa variabile siamo ricorsi ad un espediente matematico: abbiamo applicato un cambio di variabile. Partendo da X, una variabile continua uniforme in $[0, 1]$, abbiamo ricavato $Y \sim Exp(\lambda)$ da

$$Y = -\frac{1}{\lambda} log(X) \tag{2.12}$$

```
#precisione=numero di volte che voglio ripetere il test
for i in range(precisione):
    test.append((-1/lamb)*math.log(random.uniform(0, 1)))
```

Iterando questo cambio di variabile abbiamo ottenuto il set di risultati su cui condurre la simulazione. Mettendo a grafico i risultati ottenuti con questo metodo ed la funzione di ripartizione risultante abbiamo ottenuto il grafico che segue:

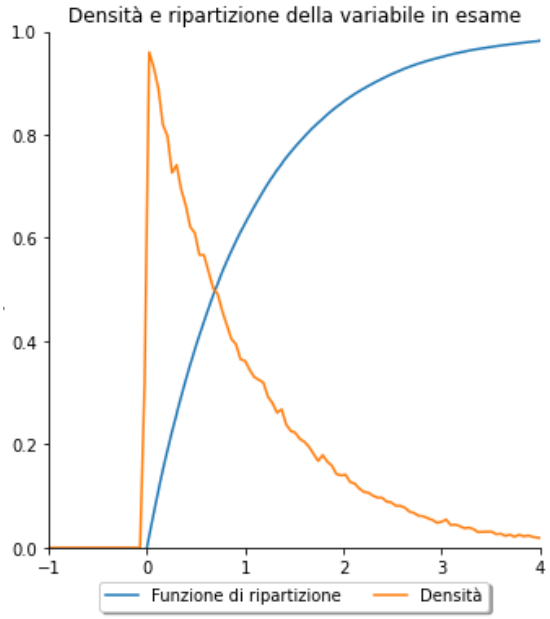
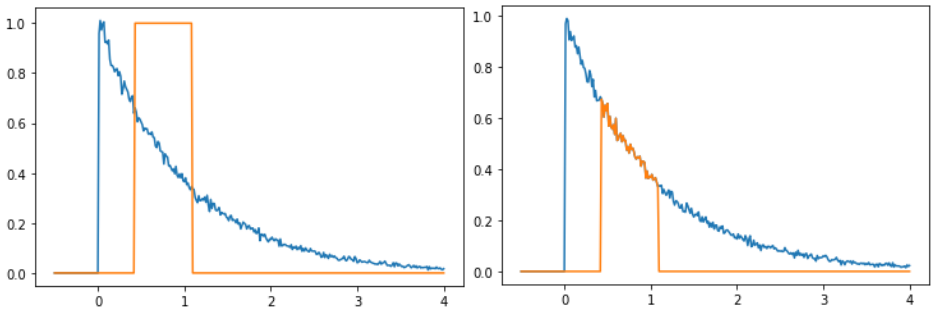


Figura 2.15: Densità continua esponenziale

Per quanto riguarda l'esempio della particella radioattiva invece abbiamo sviluppato l'argomento in tre esercizi.
Nel primo ci limitiamo a confrontare la percentuale dei risultati che hanno avuto un tempo di attesa inferiore a

quello richiesto, lasciando allo studente la possibilità di modificare il tempo di attesa.
Nel secondo lasciamo più libertà allo studente il quale può modificare anche il parametro λ. Produciamo quindi il grafico risultante e ripetiamo il primo esercizio controllando quante volte la particella è decaduta prima dell'intervallo selezionato (continuando ad affiancare il risultato ottenuto con quello calcolato con la formula)
A cambiare le carte in tavola è il terzo esercizio. Nonostante si torni alla particella radioattiva ed al parametro $\lambda = 1$, l'esercizio è più interessante poichè lavora sulla probabilità che la particella decada in un intervallo di tempo del tutto personalizzabile (anzichè prima di un certo momento).
Per produrre il grafico di questo esercizio è stato necessario ricorrere ad un espediente. Abbiamo cominciato costruendo una curva che assumesse valore 0 all'esterno dell'intervallo ed un valore maggiore della variabile esponenziale all'interno.
Abbiamo quindi calcolato la funzione data dal minimo tra la variabile esponenziale e la nuova curva.



(a) Passo 1 (b) Passo 2

Figura 2.16: Densità esponenziale nell'intervallo selezionato

Colorando l'area sottesa da questa nuova curva ed andando poi a nascondere i passaggi intermedi, abbiamo

ottenuto il grafico finale.

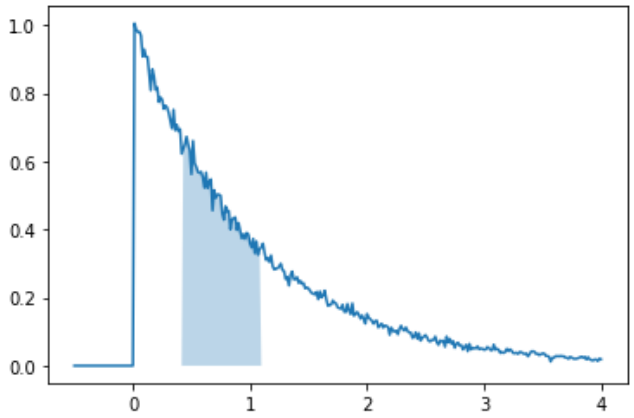


Figura 2.17: Risultato finale

2.10 Densità continua normale

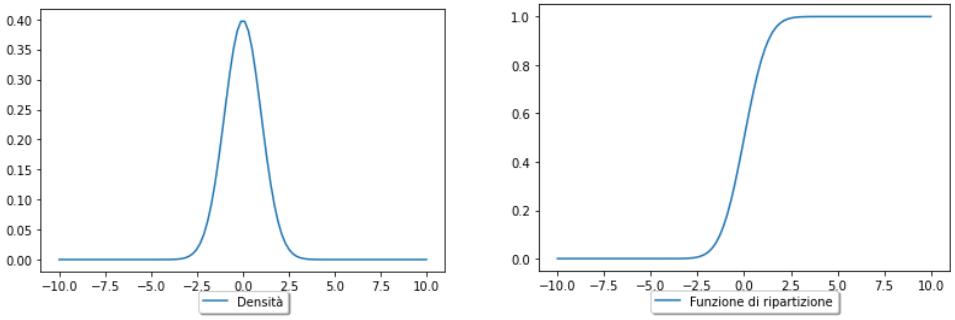
L'ultimo argomento trattato nel progetto è stata la densità continua normale (o gaussiana). Questa probabilmente rappresenta la variabile continua più importante. Si basa sull'equazione dimostrata da Gauss

$$\int_{-\infty}^{+\infty} e^{-\frac{x^2}{2}}\, dx = \sqrt{2\pi} \tag{2.13}$$

da cui consegue che

$$f(s) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \tag{2.14}$$

è una densità. Questa è la densità normale standard, formalizzata come X $\sim$ N(0, 1), dove il primo parametro corrisponde alla media, ovvero il valore medio della distribuzione, mentre il secondo è la deviazione standard. Sono quindi proposti i grafici delle funzioni di densità e di ripartizione ai quali lo studente può modificare media e deviazione per vedere come questo si ripercuote sul grafico prodotto.



(a) Funzione della densità (b) Funzione di ripartizione

Figura 2.18: Densità continua normale

Per il primo esercizio su questa variabile abbiamo descritto X una variabile continua normale standard e ci siamo interrogati sulla probabilità che assumesse un valore minore di 1. La generazione dei dati risulta banale, mentre per disegnare il grafico abbiamo utilizzato lo stesso metodo già sfruttato per le altre variabili continue.

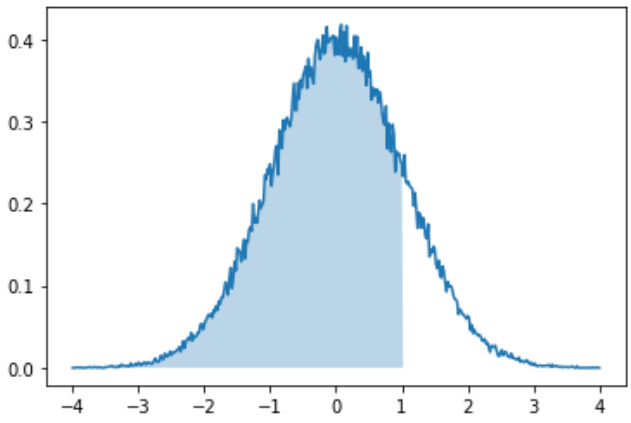


Figura 2.19: Gaussiana standard - $X < 1$

Anche in questo caso lo studente può interagire per modificare il valore massimo che può assumere X.
Per l'ultimo esperimento invece abbiamo voluto proporre qualcosa di diverso. Siccome le variabili normali si distinguono per la varietà di eventi reali che possono modellare, abbiamo voluto basare l'ultimo esercizio proprio su dei

dati reali. Abbiamo quindi utilizzato i dati raccolti da un team di ricerca di Hong Kong che aveva monitorato le altezze di un gruppo di 100 000 adulti.
Poi abbiamo ipotizzato il caso in cui un funzionario dell'esercito dovesse comprare le divise per 500 nuove reclute, domandandoci quanti capi di ogni taglia avrebbe avuto bisogno, sapendo che l'altezza di quel campione di ricerca era ben approssimata da una variabile normale $N(172, 5)$ e che le taglie erano le seguenti:

Taglia	Altezza
S	$h < 164$
M	$164 < h < 172$
L	$172 < h < 180$
XL	$h > 180$

Abbiamo quindi cominciato mettendo a confronto il grafico ottenuto dai dati raccolti su quel campione (sempre calcolandolo punto per punto in un intervallo di + o - ε) con quello prodotto da una variabile normale avente i parametri forniti.
Per produrre quest'ultima ci siamo serviti di una funzione della libreria Scipy che fornisce il valore esatto di una variabile normale coi parametri selezionati nel valore richiesto.

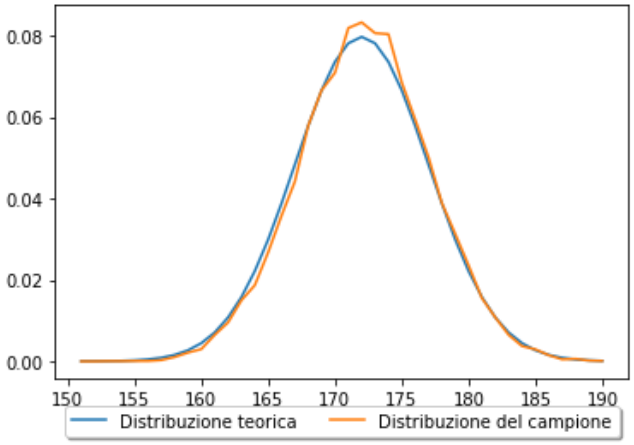


Figura 2.20: Grafico sui dati reali a confronto con $N(172, 5)$

Una volta mostrato il grado di approssimazione, abbiamo proseguito con la risoluzione dell'esercizio chiedendoci la percentuale di reclute che avrebbero avuto bisogno di una taglia L. I risultati di questo esercizio presentano un margine di errore maggiore rispetto a quelli precedenti: questa è una conseguenza diretta del lavorare su dati reali, ma l'approssimazione è comunque buona ed il grafico, prodotto con la stessa tecnica usata per la variabile esponenziale, si presenta così:

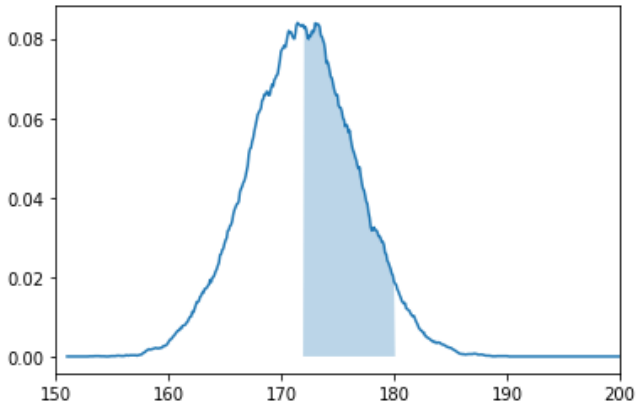


Figura 2.21: Reclute con altezza compresa tra 172 e 180 centimetri

Conclusioni e lavori futuri

Il risultato prodotto in questa tesi ha raggiunto e superato gli obiettivi preposti in fase di analisi.
Le simulazioni, incluse quelle più complesse e che lavorano su moli di dati più importanti, mantengono una buona reattività, minimizzando i tempi di attesa per osservare il risultato. Inoltre l'aspetto ottenuto nei grafici non ha disatteso le aspettative; al contrario, grazie allo studio di librerie più avanzate, è risultato più intuitivo e chiaro dei primi prototipi.

Molti argomenti del programma del corso non sono stati trattati in questa tesi, potrebbe essere interessante estendere questo lavoro per coprire anche quelle parti del programma.

Nel complesso siamo soddisfatti dei risultati raggiunti anche se per avere un riscontro effettivo della buona riuscita del progetto dovremo aspettare di presentarlo agli studenti dei prossimi anni. Tutto questo è stato fatto per loro.

Ringraziamenti

Ringrazio i miei genitori per il supporto, per avermi incoraggiato e spronato nei momenti difficili e per avermi permesso di portare avanti i miei studi. Ringrazio nonno Vito e nonna Rosalba per aver contribuito a garantirmi questa possibilità così importante e preziosa. Ringrazio tutti i miei compagni di corso, per aver condiviso con me questa esperienza ed in particolare Sara per il suo aiuto incondizionato, per la pazienza dimostrata e per aver creduto in me ogni giorno. Ringrazio i professori del corso per avermi guidato nell'apprendimento di tutte le materie che formano questo corso. Infine ringrazio il Dott. Fabrizio Caselli per aver creduto in questo progetto e per avermi accompagnato nella realizzazione di questa tesi.

Bibliografia

[1] Homepage di Numpy https://numpy.org/

[2] Homepage di Ipywidgets https://ipywidgets.readthedocs.io/en/latest/

[3] Homepage di matplotlib https://matplotlib.org/

[4] Homepage di math https://docs.python.org/3/library/math.html

[5] Homepage di seaborn https://seaborn.pydata.org/

[6] Homepage di random https://docs.python.org/3/library/random.html

[7] Homepage di scipy https://scipy.org/

[8] Homepage di pandas https://pandas.pydata.org/

[9] What is Mersenne Twister (MT)? http://www.math.sci.hiroshima-u.ac.jp/m-mat/MT/ewhat-is-mt.html

[10] Welcome in Colaboratory
`https://colab.research.google.com/?utm_source=scs-index`

[11] Homepage di Spyder
`https://www.spyder-ide.org/`

[12] What is GitHub?
`https://kinsta.com/knowledgebase/what-is-github/`

[13] Per la teoria matematica si è fatto riferimento al materiale del Dottor Fabrizio Caselli
`https://virtuale.unibo.it/pluginfile.php/990074/mod_resource/content/1/lezioniNew.pdf`