Università degli Studi di Modena e Reggio

Facoltà di **Scienze Fisiche Informatiche Matematiche**
Corso di Laurea Magistrale in **Informatica**

TESI DI LAUREA MAGISTRALE

Tecniche di text analytics e machine learning per la classificazione automatica di giochi da tavolo basata su manualistica, categorie e meccaniche di gioco

Candidato
Dattolo Simone
Matricola 115373

Relatore
Prof. Martoglia Riccardo

Anno Accademico 2020-2021

RINGRAZIAMENTI

Ringrazio il prof. Martoglia Riccardo,
la pazienza e disponibilità fatta persona,
i feedback, l'assistenza continua
e i consigli. Nulla di ciò
sarebbe stato possibile senza
la sua presenza. Matteo Pontiroli e
Luca Giovannoni per tutto il materiale
fornito prima ancora
di cominciare questo progetto.

PAROLE CHIAVE

Machine Learning

BGG

Text analytics

Boardgame

Rulebook

Meccaniche e categorie

Indice

Elenco delle figure

Introduzione

La quantità di informazioni generata ogni giorno dalle persone è estremamente elevata. Volendo trattare un qualsiasi problema attingendo da tali informazioni, è necessario usufruire di quest'ultime da fonti che le trattano e le gestiscono in maniera affidabile, chiara e trasparente. L'informazione cruda presa cosi senza un minimo di preprocessing può risultare sì utilizzabile dall'uomo, ma solo in maniera superficiale. Invece può esser inutile alla macchina ai fini di una qualsiasi indagine che si vuole svolgere. In questo contesto si inserisce l'interesse di approndire una branca di studi legata alla *Data Analytics*. Considerando la parola **analytics** [6], si intende il processo di scoperta e comunicazione di pattern grazie ai dati, attraverso la risoluzione di problemi, creazione di modelli di previsione per capire quale decisione prendere. Analytics può contare su una moltitudine di modelli computazionali e su altrettanti domini in cui si possono fare delle analisi: *web, risk, market analytics*. Tra tutte queste tipologie vi è la **Game Science Data Analytics**, [9],[6]. Questa branca scientifica è ancora ampiamente sconosciuta e decisamente poco esplorata nel contesto scientifico. Tuttavia abbraccia una moltitudine di discipline. Orientata nel dominio dell'analisi dei giochi, descrive come applicare l'analisi dei dati nel contesto della *game research*.

Partendo a monte ci si pone come fine quello di approfondire la *Game science data analytics*. L'obiettivo iniziale è quello di estrapolare informazioni e analizzarle, a partire da un grande numero di *boardgame* dal sito web più fornito in ambito di giochi da tavolo, di carte e di ruolo: BoargGameGeek [7]. Una volta ottenute vengono usate per aumentare la comprensione circa quali meccaniche/categorie di gioco inneschino il cosiddetto *CT, computational thinking* [3]. Denota l'idea di sviluppare una soluzione generica, andando a scomporla identificando variabili e pattern, ricavando così un algoritmo di risoluzione. Il computational thinking rappresenta in pratica una abilità cognitiva di applicare i fondamenti della *computer science*.

Da qui ci sono i presupposti per portare avanti un progetto di tesi il cui fine si

discosta dallo studiare il CT. Invece ci si è diretti sulla creazione e sullo studio di modelli di machine learning grazie alla text analytics, per classificare i boardgame per meccaniche di gioco/categoria che vanno a influenzare il CT. Tale classificazione avviene in un primo momento come uno studio qui riportato [2]. In aggiunta a quest'ultimo, viene utilizzata, al posto della descrizione dei boardgame, la manualistica di ogni gioco da tavolo possibile. Si conclude con l'uso di entrambi i campi testuali.

Si apre qui un altro problema, che è quello del reperimento in maniera efficiente dei rulebook. Ciò ha portato a dover apprendere ulteriori conoscenze. Si è appreso l'uso di Selenium per automatizzare un browser che scaricasse in piena autonomia i rulebook direttamente da BGG. Una volta ottenuti più manuali possibili, si è portato avanti lo studio andando a estrarre le informazioni utili da tali file. In aggiunta allo studio di modelli di machine learning, qui ne viene poi creato uno ulteriore per poter classificare, sempre tramite apprendimento supervised, ciò che è un rulebook da ciò che non lo è.

La presente tesi è strutturata in due macro parti: **la parte I** viene usata per riportare gli argomenti e le tecnologie studiate proprio per svolgere tale lavoro di ricerca. **La parte II** riporta l'implementazione, la traduzione dei problemi e il loro svolgimento, con il fine di arrivare a dei risultati da poter studiare, il tutto usando la teoria e le tecnologie studiate nella parte precedente. Il lavoro continua dividendo nel complesso la tesi in 7 capitoli, i cui primi 3 sono incentrati sulla spiegazione dei problemi trattati e le tecnologie a cui ci si è approcciati per risolverli. I successivi 4 sono dedicati alle soluzioni ai problemi proposti, alla creazione di modelli e alla raccolta dei risultati.

Il **capitolo 1** verte sull'introduzione ai problemi trattati in questo progetto di tesi, dal download della manualistica, al riconoscimento dei rulebook, per arrivare alla classificazione dei boardgame per meccaniche/categorie tramite text analytics e machine learning.

Il **capitolo 2** illustra le tecnologie studiate e utilizzate nel contesto del download della manualistica, ma anche per la creazione dei dati oggetti di studio in questo progetto.

Il **capitolo 3** mostra il lato teorico dietro al Machine Learning, il cuore degli studi di questo progetto: dal funzionamento, alla nascita, a come viene usato nei giorni nostri, e come è stato utilizzato per la classificazione e creazione di modelli sui boardgame.

Il **capitolo 4** è il primo interno alla parte II, mostra come è stato creato il dataset che poi ha accompagnato tutto il progetto di tesi, come è strutturato il sito

web da cui si sono ricavate tutte le informazioni necessarie. Si parte da come è formato un boardgame sul sito BGG, a come scaricarne una quantità massiccia, fino a estrarre da ognuno di essi le informazioni necessarie per costruire il dataset.

Il **capitolo 5** è interamente dedicato alla manualistica dei boardgame. Illustra quali sono le problematiche legate ai rulebook, dove reperirli e con quale garanzia si ottengono dei buoni risultati. Si prosegue con la preparazione e la creazione di una routine che possa scaricare i manuali in piena autonomia senza doverli scaricare a mano. Si mostra come BGG offra un margine di download possibile, in quanto per ogni boardgame dedica ad esso una sezione Files in cui è possibile reperire direttamente file legati al gioco stesso, il manuale in moltissimi casi.

Il **capitolo 6** è il culmine dello studio di questo progetto di tesi. Si occupa inizialmente di spiegare come sono stati etichettati i rulebook scaricati con la routine del capitolo precedente. I manuali così scaricati sono stati oggetto di parsing, e quindi estrazione di informazioni da essi, per poter esser aggiunte al dataset utilizzato per gli studi. Il capitolo conclude con i modelli di machine learning creati per riconoscere i documenti che sono manuali di gioco da ciò che non lo sono.

Il **capitolo 7** rappresenta un altro apice dello studio di questo progetto di tesi. Diviso in due grandi sezioni, in quest'ultimo capitolo vengono illustrate le classificazioni dei boardgame fatte. In un primo momento usando le sole meccaniche, avvalendosi in un primo momento del solo rulebook, poi della description e in conclusione di entrambi i campi testuali. In un secondo momenti i medesimi modelli di machine learning sono stati fatti andando a classificare i giochi per categoria, usando i campi testuali precedentemente citati.

Parte I

Stato dell'arte

Capitolo 1

Introduzione al problema

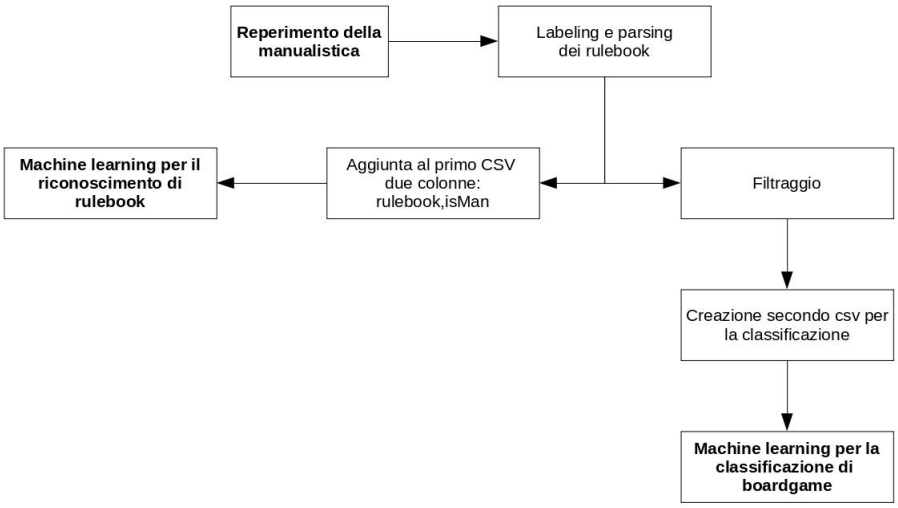
Una introduzione al problema offre una panoramica ben precisa di cosa si va a trattare. Essendo il problema di questo elaborato in realtà un insieme di problemi, 3 per la precisione, per ognuno di essi si va a illustrare cosa comporta risolverlo, quali tecnologie sono state usate per il singolo problema e cosa quest'ultimo ha prodotto. Ognuno di essi infatti merita attenzioni ben specifiche poiché sono problemi non banali. Di seguito di riporta l'elenco dei tre problemi:

1. *il reperimento della manualistica per ogni boardgame*
2. *un modello basato su machine learning che riconosca ciò che è un manuale di gioco da ciò che non lo è*
3. *la creazione di un modello basato su machine learning che possa classificare i boardgame per meccanica/categoria grazie all'uso dei rispettivi manuali, seguito dalla descrizione del boardgame*

Chiaramente per fronteggiare questi problemi che si sono rivelati complessi è stata necessaria una prima una fase di preparazione presente nella sezione 1.1. Invece si trova un approfondimento nel capitolo 2 circa le tecnologie specifiche utilizzate.

Dalla immagine 1.1, si può notare la linearità dei passaggi. Questo è dato dal fatto che il ranking da cui si va ad attingere per scaricare i boardgame è in continua evoluzione. Scaricare informazioni oggi comporta l'avere informazioni già vecchie appena dopo poco tempo. Indi per una indagine aggiornata per comodità si consiglia di ripartire da capo. Il **reperimento della manualistica** offre come output direttamente i manuali (in formato pdf/doc/docx). Dopodiché ognuno di essi deve esser etichettato come tale, poiché non c'è alcuna garanzia circa il fatto che siano effettivamente dei manuali a tutti gli effetti. Una volta etichettati, è possibile

Figura 1.1: Visualizzazione degli step

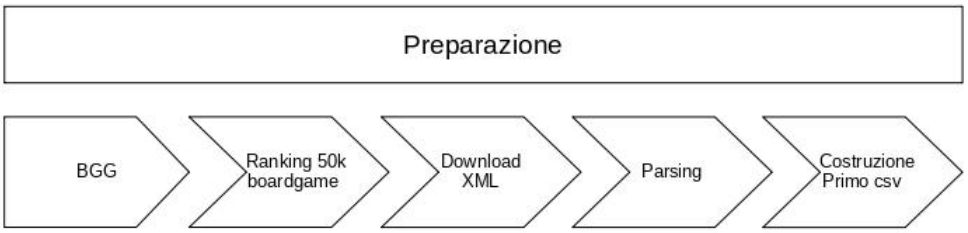


estrapolare da essi l'informazione, potendo svolgere poi il secondo problema, **machine learning per il riconoscimento di rulebook**. Per l'ultimo invece è necessaria una fase di filtraggio ulteriore dei manuali, e una volta estratta l'informazione da essi è possibile poi svolgere **machine learning per la classificazione dei boardgame basata su rulebook**.

1.1 Preparazione, creazione e download dei dati da BGG

Per parlare di ML bisogna avere dei dati su cui costruire poi i modelli necessari. Il sito BGG è un'enciclopedia per tutti coloro che sono appassionati di giochi da tavolo, di ruolo e di carte, varie ed eventuali. Al momento è il più grande sito web in questo ambito, dal gennaio del 2000 a questa parte, che possa fornire una mole interessante di informazioni, utilizzabili per costruire un dataset necessario per fare ML.

Figura 1.2: Iter per il download e la creazione dei dati



L'immagine 1.2 mostra le varie fasi circa la preparazione dei dati: si comincia col contattare il sito bgg, per poi visualizzare il ranking globale del sito inerenti ai boardgame. Si procede con il download dei boardgame direttamente dai database di BGG in formato XML [10]. Grazie a BGG, è possibile scaricarli in maniera semplice ed efficace utilizzando le API fornite direttamente dai gestori del sito stesso [11]. Le ultime fasi vengono svolte grazie all'uso di python [8] come unico strumento utilizzato per svolgere gran parte del lavoro. Il parsing dei boardgame e la costruzione del csv sono stati possibili grazie a librerie come Beautiful Soup [12], libreria python pensata per estrapolare informazioni da file strutturati come HTML e XML, e Pandas [13], tool molto potente per python utilizzato per effettuare analisi di dati e manipolazione di dataframe. Conclusa la preparazione dei dati, si è in possesso di una moltitudine di file XML, una volta parsati si è potuto costruire il dataset in formato csv, che viene utilizzato per il secondo e terzo problema.

1.2 Reperimento della manualistica

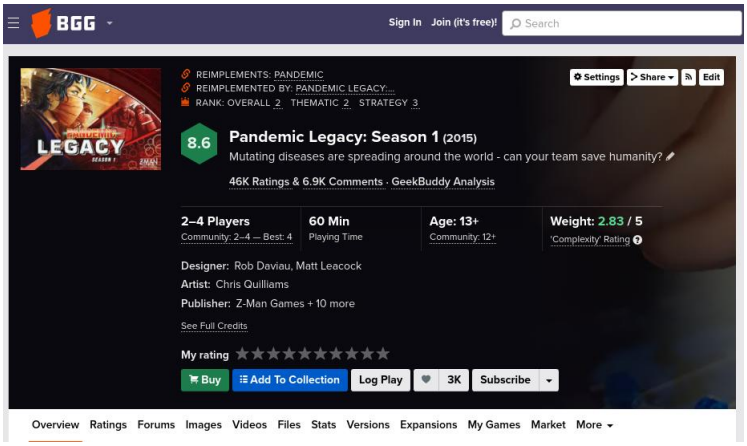
Boardgamegeek, come già citato, è un sito web enorme da cui attingere informazioni inerenti a tutta la branca dei boardgame. Quando si parla di manualistica, o *rulebook*, le cose si fanno più complesse. La piattaforma rappresenta un punto di incontro per tutta la community online appassionata di boardgame. Risulta possibile avere un account, rilasciare commenti, popolare il proprio profilo con i rispettivi giochi che si hanno, compilare una propria wishlist, rilasciare dei thumbs up e via dicendo.

La manualistica di ogni gioco purtroppo non segue alcuno standard. Ogni gioco possiede la propria pagina su boardgamegeek, con una sezione inerente ai file che gli utenti possono caricare, a favore della comunità, e che sono scaricabili dai soli utenti del sito. Si riporta in figura 1.3 un esempio di boardgame, *Pandemic: Legacy*, al secondo posto del ranking globale. Alla base di tale immagine è possibile notare come siano molteplici le schede che un boardgame possiede: grazie ad esse infatti si possono veder le recensioni del gioco, la posizione, vedere foto e video caricate dagli utenti, eventuali espansioni che il gioco possiede, varie ed eventuali.

La scheda *file* di tale boardgame, si può notare come ci siano dei documenti caricati dagli utenti del sito, come mostra l'immagine 1.4. Tra questi file purtroppo non sempre è disponibile un documento che possa essere un manuale, e soprattutto può non avere un formato adatto per essere utilizzato.

La complessità di questo primo problema sta nel scaricare una quantità elevata

Figura 1.3: Esempio di boardgame nel ranking



di manuali. Per effettuare il download, come già accennato, è necessario essere utenti registrati con una sessione sul browser attiva. Questo problema è stato risolto tramite una procedura di login e download automatizzata, usando Firefox e Selenium [14]. I documenti scaricati vengono effettivamente scelti secondo un criterio ben specifico. Risulta possibile infatti settare l'ordine dei file per i boardgame (thumbs up decrescenti, lingua inglese). Ciò permette così di avere un primo set formato da manualistica adatta per essere scaricata e poi trattata a dovere. Un esempio illustrato in figura 1.5.

Figura 1.4: Esempio di file nel boardgame

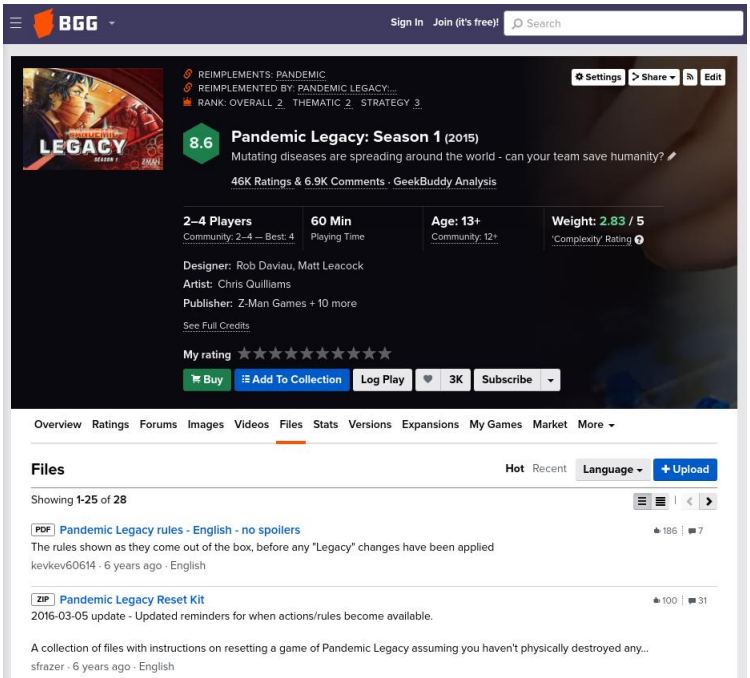
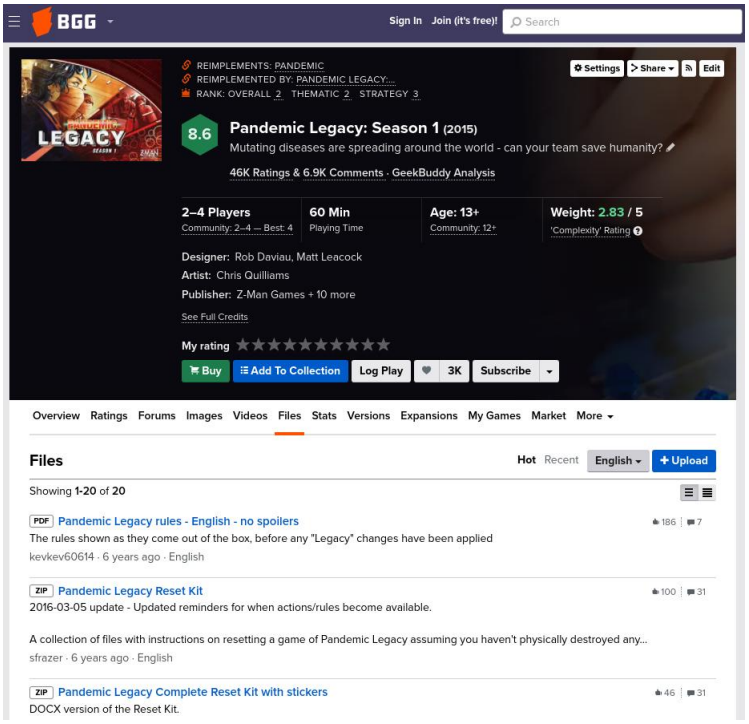


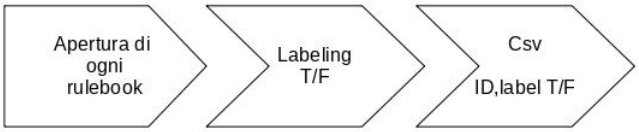
Figura 1.5: File ordinati per indice di gradimento, in inglese



1.3 Labeling, parsing e filtraggio dei rulebook

Una volta scaricato il rulebook per ogni boardgame, si procede con una fase di labeling. *I manuali scaricati, per verificare se fossero manuali o no, sono stati aperti uno ad uno proprio per essere controllati*, dopodiché si è potuto etichettare True/False a seconda che fossero o meno dei manuali. Si mostra in figura 1.6 l'ordine delle operazioni.

Figura 1.6: Iter seguito apertura-csv-labeling True/False



Grazie a pyPDF2 [15], si procede a estrapolare il testo dai file. Nel caso ci fossero delle scansioni, al posto di testo normale, viene in soccorso una libreria che si avvale di OCR [16]. I boardgame di cui non si ha il manuale sono stati scartati. Arrivati a tale punto *è possibile effettuare il riconoscimento dei manuali.* Non è ancora possibile svolgere machine learning per classificare i boardgame basandosi su rulebook, per alcuni motivi:

- effettuato il parsing dei file, per motivi non noti, il contenuto risulta illeggibile. Pur essendo magari un manuale poiché etichettato True in precedenza, non può esser usato per la classificazione dei boardgame.

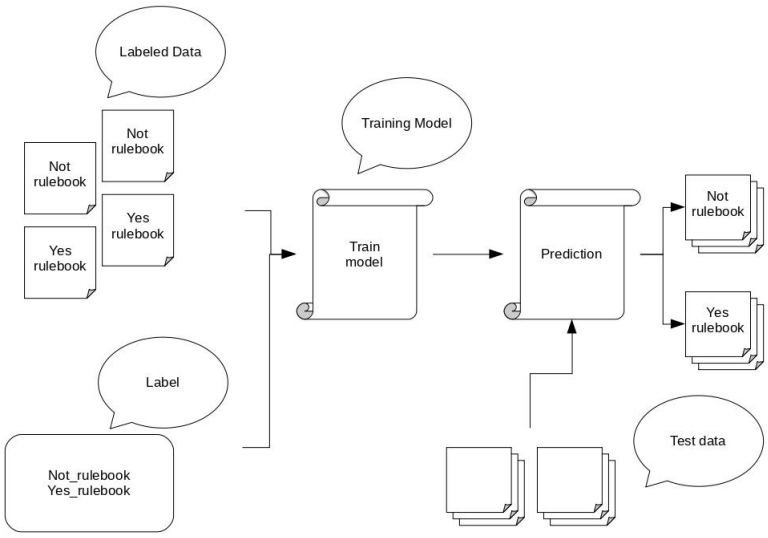
- il file risulta obiettivamente vuoto di testo, oppure in minima parte in inglese con un insieme di lingue, o viene mostrata una tabella, un cheatsheet, un riassunto dei punti salienti del gioco o addirittura la plancia da gioco, che in nessuno dei precedenti casi rappresenta un manuale a tutti gli effetti. Non è possibile usarlo per la classificazione dei boardgame perché non ritenuto un rulebook.

L'eliminazione di boardgame che hanno un rulebook a tutti gli effetti ma illeggibile, è stata effettuata in due maniere: *automatica tramite un algoritmo*, e *manuale.* Questo perché scrivere un algoritmo che filtrasse in maniera solitaria senza dove guardare a mano ogni contenuto dei rulebook se fosse decodificato bene o no, è stato ritenuto un passo importante per rendere questo passaggio meno pesante.

1.4 Riconoscimento di rulebook

Con i dati a disposizione finalmente risulta possibile svolgere del machine learning. Questo perché nella fase precedente i rulebook sono stati aperti uno ad uno, etichettati a dovere, estrapolati i rispettivi contenuti ed ora possono esser usati a tutti gli effetti. Si è creato così un classificatore che riconoscesse ciò che è un manuale da ciò che non lo è. Viene effettuata della Cross Validation per verificare la bontà dei modelli prodotti. La letteratura al giorno d'oggi non fornisce nulla circa di esperimenti e test fatti usando il manuale dei boardgame per trarre da essi informazioni e costruire dei classificatori.

Figura 1.7: Iter machine learning per il riconoscimento di rulebook



Nella figura 1.7 viene riportato il workflow che tipicamente viene seguito in un contesto di machine learning supervised. Sono necessari dei dati etichettati in precedenza, altrimenti non saremmo in un ambito supervisionato, in quanto le predizioni dei campioni vengono fatte costruendo un modello di training proprio dai dati supervisionati in precedenza dall'uomo. Una volta creato il modello dai dati di training, è possibile effettuare delle predizioni usando dei dati di test. Questi ultimi non devono essere utilizzati in fase di allenamento, perché banalmente se ciò dovesse accadere si andrebbe a invalidare le prediction in quanto si andrebbero a prevedere dei campioni che il modello conosce già a priori.

1.5 Classificazione di boardgame

Questo terzo problema è stato trattato in due modi diversi: la classificazione per i boardgame è stata effettuata in un primo momento per meccaniche, successivamente per categorie. Ciò è stato reso possibile poiché si è costruito il dataset al cui interno sono presenti queste due feature ricavate direttamente dagli XML dei boardgame, scaricati da BGG.

La text analytics ha permesso di utilizzare, come campo testuale, la description di ogni boardgame. Sono state seguite le orme del seguente studio [2], che ha sviluppato dei modelli per classificare i giochi per meccaniche e categorie, tramite l'uso della description dei giochi. Questo progetto di tesi ha esteso il seguente studio effettuando machine learning utilizzando la manualistica di ogni gioco in primis. Successivamente ha unito entrambi i campi testuali per replicare lo studio ed effettuare dei confronti in merito ai risultati ottenuti.

In figura 1.8 e 1.9 si riportano l'iter, equivalente, che si è seguito per la classificazione per categorie e meccaniche. L'input in fase di training sono i boardgame da intendere come la description del gioco da tavolo, in virtù del fatto che si è partiti seguendo le orme dello studio già precedentemente citato [2]. Questo campo testuale è stato poi cambiato utilizzando il solo manuale per ogni gioco, poi usando sia description che rulebook. In conclusione si hanno così tre tipi di studi per quest'ultimo problema: i risultati ottenuti con il solo uso della description, con il solo manuale, o entrambi. Tutto ciò prima effettuato sia per le categorie che meccaniche di gioco.

Figura 1.8: machine learning supervised per classificare i giochi per categoria

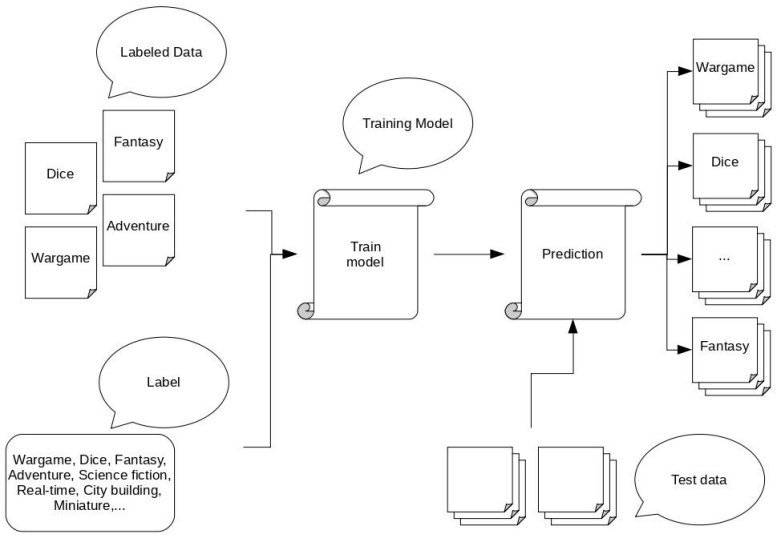
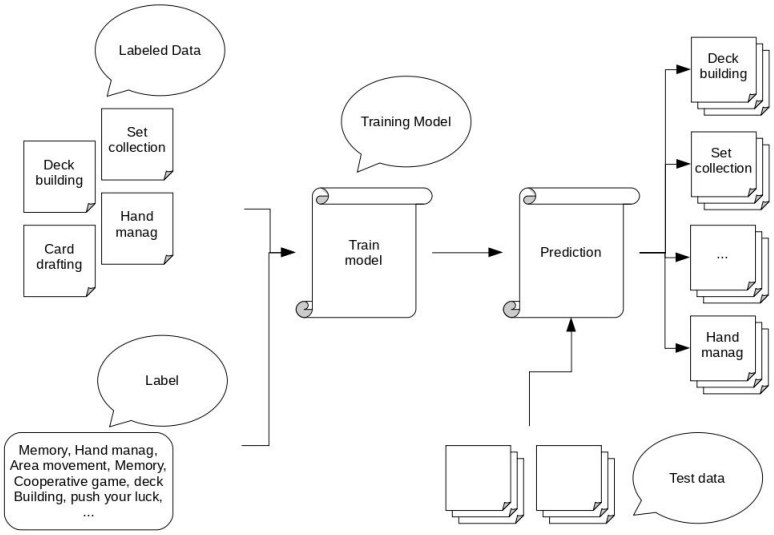


Figura 1.9: machine learning supervised per classificare i giochi per meccanica



Capitolo 2

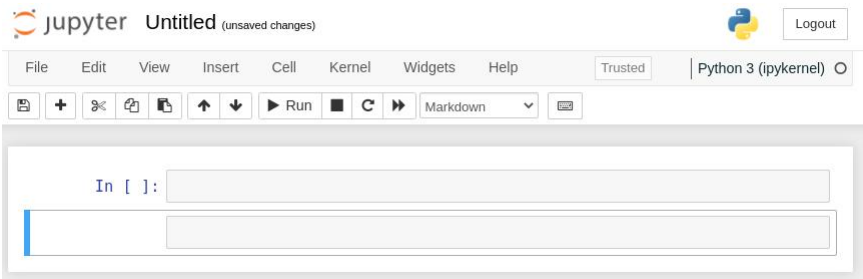
API per XML e Selenium

Nel capitolo 1 si è voluto dare una panoramica dei 3 grandi problemi affrontati. Si necessita però di un excursus teorico fornito da tale capitolo, su quali siano gli strumenti usati in fase di preparazione dei dati e come effettuare parsing di file XML, estrapolando da essi le informazioni. Si conclude illustrando invece il funzionamento di un tool rivelatosi estremamente utile per il download della manualistica in maniera completamente automatica.

2.1 Jupyter

Jupyter è un software liberamente scaricabile [18]. Utilizzato come servizio web interattivo, è sfruttato come strumento computazionale, usando anche differenti linguaggi di programmazione. Jupyter notebook è una applicazione web per la creazione e la condivisione di documenti scientifici. Come infrastruttura supporta una moltitudine di linguaggi, tra cui R, python e scala. Si può inoltre condividere i documenti tramite mail, dropbox e github. Il codice produce un output ricco e interattivo in molti formati, tra cui HTML e LaTeX.

Figura 2.1: Esempio jupyter



Si riporta in figura 2.1 un esempio di notebook Jupyter. Sono illustrate delle celle vuote. La prima è impostata per la scrittura di codice python, che permette all'interprete di elaborare ed eseguire. Così facendo è possibile eseguire qualsiasi cella. La seconda è utilizzabile per la scrittura di testo libero in Markup, un linguaggio simile ad RMarkdown. Permette la formattazione di testo, inserimento di video e foto e di chunk in LaTeX. In conclusione, è possibile convertire il file jupyter in un altro formato, come ad esempio HTML o un file pdf. Il progetto di tesi è stato portato avanti svolgendo elaborati grazie a Jupyter.

2.2 BeautifulSoup per il download di XML

La fase preparatoria dei dati include anch'essa degli strumenti utilizzati che devono essere riportati e illustrati. Questa fase, seppur non facente parte di alcuno dei 3 problemi, è risultata obbligatoria per ottenere i file XML, da cui sono state estratte le informazioni per la creazione dei dataset usati per effettuare machine learning.

Per il download degli XML è stata utilizzata la libreria python *request* in combinazione a *BeautifulSoup*. Una volta ottenuta la pagina web, quest'ultima è in grado di effettuarne il parsing, per vedere la struttura del file, riconoscere header, body e vari altri tag.

Per estrarre le informazioni vere e proprie all'interno dei file XML si è provveduto all'utilizzo di un altro pacchetto python, chiamato *ElementTree*. Essa è una libreria utilizzata per estrapolare qualsiasi tipologia di informazione da file di markup, come HTML e XML. Lavora coi migliori parser che provvedono a molteplici modi per navigare nell'albero dei file. Si riporta in 2.2 un esempio di XML. Si può notare come ogni campo, determinato da un *tag* di inizio e di fine, abbia una certa informazione, che può essere di qualsiasi tipo, un path ad un file generico, una stringa, un intero, e via dicendo.

Nella immagine 2.3 invece si illustra l'albero associato al file XML. Questa tipologia di file strutturato si presta molto bene per ElementTree. Tale libreria infatti può navigare dentro l'albero con metodi appositi che rendono molto semplice l'estrapolazione delle informazioni dai vari tag.

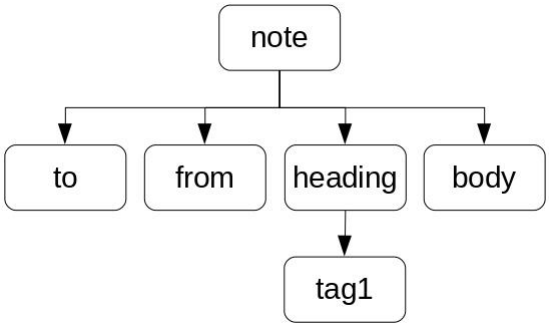
Una volta appreso ciò, si è potuto utilizzare nell'ambito di studi. Utilizzando la libreria sono stati estratti da ogni XML rappresentante ognuno dei vari boardgame informazioni come:

- *nome*, il nome del boardgame

Figura 2.2: Esempio di XML

```
<note>
    <to>Don</to>
    <from>Joe</from>
    <heading>Header part</heading>
    <body>
        Example body
            <tag1>
                example of tag1
            </tag1>
    </body>
</note>
```

Figura 2.3: Esempio di albero XML



- *ID*, un numero che identifica il boardgame
- *boardgamecategory*, le categorie di gioco cui il boardgame appartiene
- *boardgamemechanic*, le mecccaniche di gioco cui un boardgame si basa
- *description*, la descrizione del boardgame
- ...

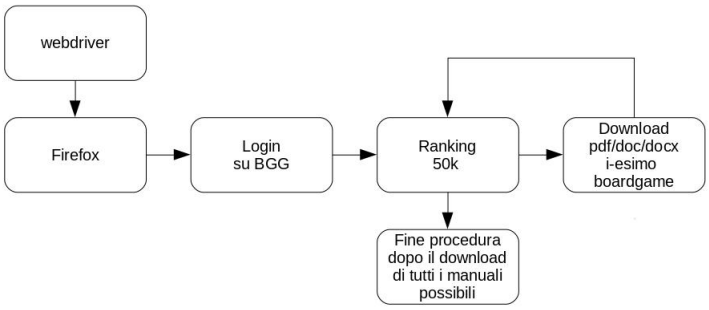
grazie a questi campi si è potuto costruire un csv utilizzabile anche subito per effettuare appunto machine learning per la classificazione dei boardgame, usando come label le categorie o meccaniche, avvalendosi della description o del rulebook (o di entrambi) per classificare i giochi.

2.3 Selenium

Per parlare di *Selenium* si comincia dando alcune nozioni di base, necessarie per capire in cosa Selenium è stato utile e cosa ha prodotto.

Si parla di **webdriver** in riferimento a un'interfaccia di controllo remota, che permette l'utilizzo e il controllo di un *user agent*. Per user agent si intende un qualsiasi applicativo che si collega ad un processo server generico. Tipico esempio di UA sono i browser web. Il webdriver consente, tramite protocolli appositi, di effettuare dei test che possano automatizzare un UA. Si inserisce qui Selenium, un framework che riesce ad automatizzare un browser. In questo progetto di tesi è stato utilizzato Firefox.

Figura 2.4: Esempio di funzionamento Selenium



Si riporta in 2.4 il funzionamento ad alto livello di Selenium. Una volta settati i parametri del webdriver, e connesso con il browser con cui si intende usarlo, si può far partire, automatizzando la procedura di download dei manuali. L'unico step manuale è l'inserimento da parte dell'utente delle credenziali per accedere al sito BGG, in quanto ritenuto poco sicuro leggerle da file o approcci affini. Una volta effettuato il login, Firefox è in grado di navigare in piena autonomia il sito, andando sulla scheda file di ogni boardgame, e scaricare il primo pdf/doc/docx, se presente.

Capitolo 3

Machine Learning

Nel capitolo 2 si sono riportati gli strumenti appresi per scaricare le pagine dei boardgame direttamente da BGG, estrapolare le informazioni dai boardgame in formato XML, automatizzare il browser per poter scaricare in maniera automatica i file, e uno strumento per scrivere ed effettuare calcoli computazionali e redarre pdf da poter condividere. Una volta fatto ciò si procede con tale capitolo, per fare in modo di avere una infarinatura circa gli argomenti necessari per capire i due problemi rimanenti. Si parla in tale capitolo infatti di *Machine Learning*, che cos'è, quando è nato, come e dove si usa, caratteristiche e algoritmi fondamentali utilizzati nel corso degli studi.

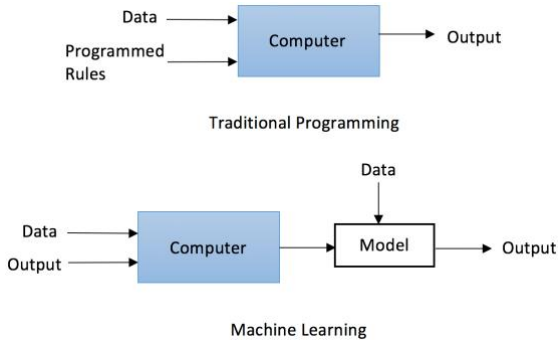
Machine Learning è un termine coniato negli anni '60, composto da due parole: *machine*, in riferimento a una macchina, computer, *learning*, si riferisce invece a una attività che si intende far apprendere o scoprire alla macchina, grazie a dei pattern, utile all'uomo. I motivi per cui ci si spinge a fare in modo che una macchina apprenda come un umano sono molteplici: le macchine possono realmente lavorare h24 sempre e comunque. Sono programmate dall'uomo, abbattono i costi, non sono soggette a stanchezza. Le macchine, guidate da questi ultimi, seguono algoritmi che aiutano a migliorare dei task che per l'uomo possono essere ripetitivi, logoranti e stancanti. Un esempio può esser la guida assistita: un veicolo capace di muoversi basandosi sull'ambiente, prendendo decisioni senza l'input umano.

Se si assume che l'uomo non patisca fatica, abbia abbastanza risorse per lavorare al massimo dell'efficienza, e si abbiano sufficienti lavoratori, il machine learning potrebbe comunque avere posto nella società? Le macchine lavorerebbero al loro stesso livello o anche meglio rispetto agli umani esperti in un determinato settore. Sulla base degli algoritmi che sono progettati e usati dalle macchine, pensati per imparare dalla verità, l'uomo comunque tende a fare degli errori. Le macchine tendono a minimizzare le

possibilità di prendere scelte sbagliate, usando un'intelligenza dettata proprio dagli esperti. Pertanto il concetto che sta emergendo viene chiamato **IA+intelligenza umana**, che combina i benefici del machine learning a quelli degli uomini. L'esempio principe può esser dettato da un'operazione chirurgica grazie all'utilizzo di una macchina, assieme a un dottore.

Il machine learning può esser paragonato a un set di regole che coinvolgono programmazione, esecuzione e lettura di un semplice set di regole e studio di determinati pattern? Se la risposta fosse sì, si potrebbe semplicemente costruire dei software, continuare a programmare ed estendere a nuove regole i software. La risposta invece è no. Questo perché effettuare continuamente l'update di regole e di pattern può diventare sempre più costoso a livello di risorse con l'andare del tempo. Il numero di pattern da ricavare per un'attività può essere enorme, non è praticamente ammissibile. Invece, al giorno d'oggi, è più conveniente lavorare con eventi che sono dinamici, che cambiano in diretta. Risulta più facile ed efficiente sviluppare algoritmi di apprendimento che comandino le macchine ad apprendere ed estrarre pattern da sé, usando una mole massiccia di dati. In figura 3.1 si riporta la differenza fra programmazione classica e machine learning.

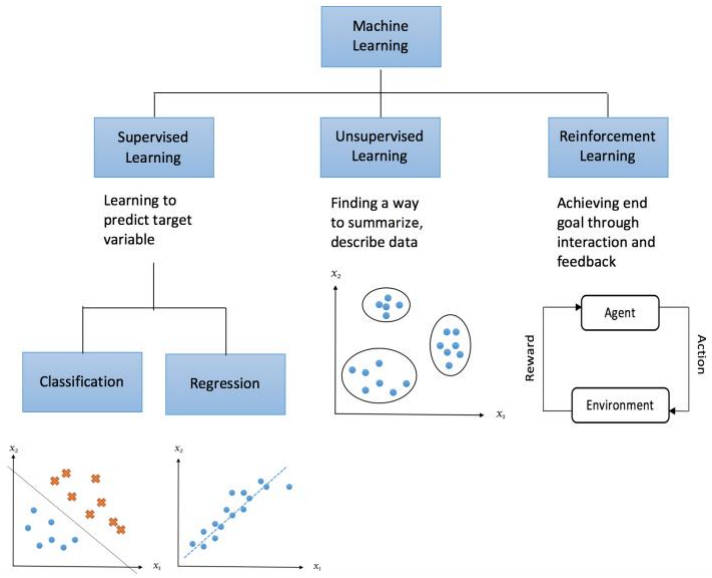
Figura 3.1: Differenza tra programmazione classica e machine learning, [1]



Un'altra motivazione che spinge ad utilizzare il machine learning è la crescita esponenziale dei dati. Il flusso di informazioni generate dalle persone, al giorno d'oggi, non è minimamente immaginabile. Il machine learning che imita l'intelligenza umana è un sottocampo dell'intelligenza artificiale. Il machine learning è decisamente molto vicino anche all'algebra lineare, teoria della probabilità, statistica e ottimizzazione matematica. Si costruiscono solitamente modelli basati sulla statistica o la teoria della probabilità, e si provano ad ottimizzare usando l'ottimizzazione matematica.

3.1 Tipologie di machine learning

Figura 3.2: Differenze fra le varie tipologie di machine learning [1]



Un sistema basato su machine learning viene nutrito da dei dati in input, testuali, numerici, immagini, suoni. Il sistema ha poi un output, un numero decimale, un intero, una stringa, che possono rappresentare una **categoria/classe**. Il focus principale del machine learning è quello di esplorare e costruire algoritmi che possano imparare dalla cronologia storica dei dati, effettuare così predizioni su nuovi dati di input. Per soluzioni *data-driven*, c'è la necessità di definire una funzione di valutazione, o *evaluate function*, chiamata anche *loss* o *cost function*. Essa è una misura di quanto bene il modello sta imparando. A seconda della natura dei dati, il machine learning può esser classificato in tre tipologie:

1. *unsupervised machine learning*: quando i dati che si usano per apprendere contengono informazioni senza alcuna descrizione associata. Il fine è quello di trovare la struttura al di sotto dei dati, per scoprire informazioni nascoste e descrivere i dati. Questo tipo di apprendimento su tali dati viene chiamato *unlabeled* data, perché appunto i dati non hanno una etichetta che li caratterizza.

2. *supervised machine learning*: quando i dati che si usano contengono informazioni con associata una descrizione, un target o un output desiderato. Il fine è trovare delle regole generali che possano mappare gli input agli output. Questo tipo di apprendimento su questi dati si chiama *labeled* data, poiché essi hanno

una etichetta che li classifica. Le regole di apprendimento quindi si usano per etichettare nuovi dati senza sapere il loro output. Le label vengono fornite spesso da alcuni sistemi intelligenzi o umani esperti nel settore. Questo tipo di machine learning è usato oggi per alcune applicazioni, come la *pattern e speech recognition, products or movie reccomendations.*

3. *reinforced machine learning*: i dati usati per apprendere sono collegati tramite retroazione al sistema. Ciò permette di adattare il sistema stesso a delle condizioni dinamiche, in ordine di raggiungere un certo scopo alla fine. Il sistema valuta le performance basate sulla risposta della retroazione, e reagisce di conseguenza.

3.2 Overfitting, underfitting

Il fine del machine learning è quello di essere capace di generalizzare. Ciò è possibile, per esempio nel machine learning supervisionato, se il modello reagisce bene di fronte a dei dati che non ha mai visto. Questo implica che il modello è in grado di riuscirci se durante la fase in cui è stato allenato è stato capace di generalizzare a sufficienza. Non sempre avviene ovviamente. Si parla infatti di due situazioni in cui si può cadere durante il machine learning: *overfitting* e *underfitting.* Il significato di questi due aspetti si spiega con la definizione di *bias e varianza.* Il bias è **il valore atteso della differenza fra la predizione e il valore vero**, calcolato come:

$$Bias[\hat{y}] = E[\hat{y} - y] \tag{3.1}$$

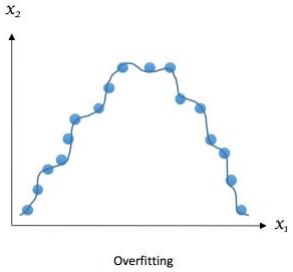
dove $\hat{y}$ rappresenta la predizione.
Quando avviene un fenomeno di memorizzazione, può di fatto causare overfitting. Il modello lavorerà bene solo sui dati usati in fase di allenamento, e le loro predizioni saranno estremamente precise. Questo fenomeno è detto *low bias.* Al contempo l'overfitting non aiuterà a generalizzare il modello sui dati, ricavando così dei pattern estremamente legati ai dati di training. Il modello avrà pertanto delle perfornace pessime su dati che non ha mai visto. Si parla di *high variance.* La varianza **è la dispersione delle predizioni alla loro media**. Ciò vuol dire:

$$Variance[\hat{y}] = E[\hat{y}^2] - E[y]^2 \tag{3.2}$$

Overfitting può verificarsi anche quando il modello è eccessivamente complesso, ci sono molti parametri e si vogliono descrivere le regole per apprendere usando poche osservazioni. Un esempio in figura 3.3

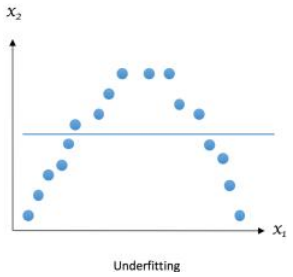
Si parla invece di underfitting quando il modello non riesce ad avere delle buone

Figura 3.3: Esempio di overfitting [1]



performance durante l'allenamento dei dati e non lavorerà bene di conseguenza in fase di test. Questa situazione può avvenire quando non ci sono abbastanza dati per allenare il modello, oppure si prova ad usare un modello non adatto ai dati che si hanno a disposizione. Si parla di *high bias*. Tuttavia la varianza del modello è bassa, come le performance, sia in fase di allenamento che in fase di test. In figura 3.4 un esempio. Banalmente la situazione in cui ci si vuole trovare risulta

Figura 3.4: Esempio di underfitting [1]

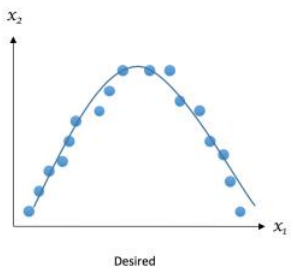


quella in figura 3.5. Si vuole evitare le situazioni elencate in precedenza. Alto bias si traduce in underfitting, mentre la varianza misura quanto la sensibilità del modello varia alle variazioni del dataset. Quindi si ha bisogno di evitare casi dove sia il bias sia la varianza siano alti. C'è bisogno pertanto di trovare un trade-off tra bias-varianza. Minimizzare l'errore totale di un modello richiede una grande attenzione fra bilanciamento di bias e varianza. Dato un set di training $x_1, \dots x_n$ e le rispettive label $y_1, \dots, y_n$ si vuole trovare una funzione di regressione $\hat{y}(x)$ che stimi la vera relazione di $y(x)$ il più correttamente possibile. Si misura l'errore di stima, chiamato **Mean Square Error (MSE)**:

$$MSE = E[(y(x) - \hat{y}(x))^2] = Bias[\hat{y}]^2 + Variance[\hat{y}] \tag{3.3}$$

Il Bias misura l'errore di stima, mentre la varianza descrive quanto la stima $\hat{y}$ si muove attorno alla rispettiva media. Più il modello di apprendimento $\hat{y}(x)$ risulta

Figura 3.5: Fit desiderato di un modello [1]



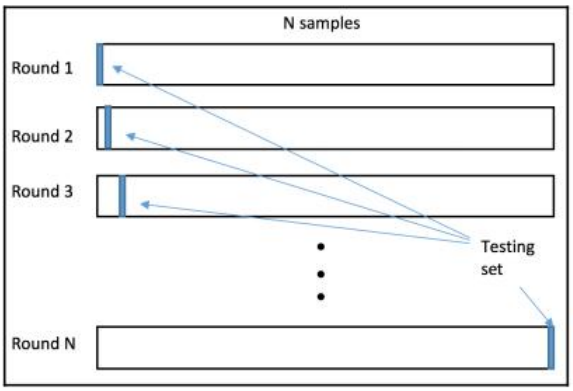
complesso con più campioni di training, più il bias sarà basso. Tuttavia questo andrà a creare più condizioni di possibilità circa l'effettuare il miglior fit del modello a causa dell'aumento dei dati. La varianza di conseguenza può lievitare.

3.3 Cross-validation

La cross validation è un metodo utilizzato in ambito di machine learning per limitare casi di overfitting. Questo approccio consiste nel prendere l'insieme di training set, dividendolo in due subset. Uno viene usato per la fase di allenamento del modello, ovvero il training, il cui subset è superiore in termini di campioni posseduti. L'altro subset invece viene usato per la fase di testing. Vengono registrate le performance in questo modo. Questo procedimento può esser ripetuto in maniera iterativa. La strategia con cui si cambia il test set prende nomi diversi:

- LOO, Leave One Out. Il test set viene costruito usando un solo campione, e tutto il rimanente usato per il training. Ogni campione così singolarmente entra nel test set. Metodo che risulta molto dispendioso se si ha un dataset enorme. Esempio riportato in figura 3.6

Figura 3.6: Leave One Out Cross Validation [1]



- K-Fold. Il dataset viene suddiviso in K fold. Su K-1 fold viene basato il training, il fold rimasto fuori viene usato in fase di testing. In figura 3.7 si riporta un esempio.

Figura 3.7: KFoldCross Validation [1]

Round	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
1	Testing	Training	Training	Training	Training
2	Training	Testing	Training	Training	Training
3	Training	Training	Testing	Training	Training
4	Training	Training	Training	Testing	Training
5	Training	Training	Training	Training	Testing

3.4 Valutazione di un modello di machine learning

Esistono nel contesto dei modelli utilizzati per classificare, delle grandezze che è necessario conoscere e non vanno trascurate. Ciò vuol dire esser consapevoli e capire soprattutto gli indici più utilizzati per comprendere le performance dei classificatori che si producono. Si riportano pertanto alcuni indici di misura, tutti compresi tra 0 ed 1, che sono:

- *accuracy*: misura la percentuale di previsioni corrette. Un valore elevato indica una accuratezza predittiva maggiore.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{3.4}$$

- *precision*: misura la percentuale di positivi effettivi tra gli esempi previsti come positivi. Un valore elevato indica una accuratezza predittiva maggiore.

$$Precision = \frac{TP}{TP + FN} \tag{3.5}$$

- *recall*: misura la percentuale di positivi effettivi come positivi. Un valore elevato indica una accuratezza predittiva maggiore.

$$Recall = \frac{TP}{TP + FN} \tag{3.6}$$

- F_1: rappresenta la media armonica della precision e recall.

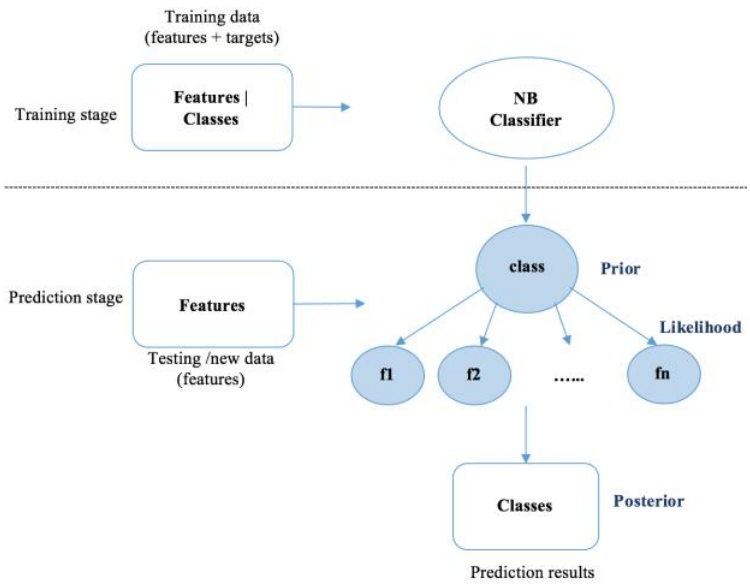
$$F_1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{3.7}$$

3.5 Algoritmi usati in ambito machine learning

Il machine learning trova la sua strada anche grazie agli algoritmi ideati per metterlo in pratica. In python sono presenti numerose librerie che utilizzano algoritmi qui citati, una di queste senza dubbio è sklearn, [17] tool semplice ed efficiente, è pensato per analisi predittiva dei dati. Riusabile, open source per uso commerciale, sotto licenza BSD, ed accessibile a tutti. Si basa su NumPy, SciPy e matpplotlib, librerie e tool di python anch'esse.

3.5.1 MNB

Figura 3.8: Multi NaiveBayes [1]



Dato un campione x con n feature, $x = (x_1, x_2, \ldots, x_n)$, il focus del Naive Bayes è di determinare le probabilità che questo campione appartenga ad ognuna delle K classi possibili $y_1, y_2, \ldots, y_k$ tale che $P(y_k|x)$ oppure $P(x_1, x_2, \ldots, x_k, k = 1, 2, \ldots, K)$. NB assume l'indipendenza condizionale fra le varie feature di una classe, questo è fondamentale. Esse, partendo dal presupposto che sono molto stringenti e non vere in situazioni reali, ha mostrato delle performance molto buone anche su problemi complessi dove questa indipendenza assolutamente era falsa. In questo modo si può applicare il teorema di Bayes:

$$P(y_k|x) = \frac{P(x|y_k)P(y_k)}{P(x)} \tag{3.8}$$

- $P(y_k)$ è il come le classi sono distribuite, senza nessuna conoscenza ulteriore delle feature. Essa è chiamata a *probabilità Bayesiana a priori* e viene data in un qualche modo o ricavata in maniera semplice dai campioni di training.

- $P(y_k|x)$, all'opposto di $P(y_k)$ è a *posteriori*, con una conoscenza extra dell'osservazione.

- $Px|y_k)$ è la distribuzione disgiunta di n feature, data dai campioni appartenenti alla classe y_k. Questa rappresenta come le feature con alcuni valori ricorrano, ed è chiamata in gergo *likelihood*. Realmente sarebbe molto complessa da calcolare con il crescere delle feature. Ciò è possibile risolverlo grazie ad una assunzione circa l'indipendenza delle feature. Può essere espressa di conseguenza come prodotto delle probabilità condizionali di ogni feature:

$$P(x|y_k) = P(x_1|y_k)P(x_2|y_k)\dots P(x_n|y_k) \tag{3.9}$$

  Ogni probabilità può esser ricavata dai campioni di training.

- $P(x)$ chiamata *evidenza*, dipende unicamente dalla distribuzione delle feature, che non appartengono specificamente ad una certa classe, e pertanto è costante. Posteriori è proporzionale alla priori e a likelihood, ovvero:

$$P(y_k|x) \propto P(x|y_k)P(y_k) = P(x_1|y_k)P(x_2|y_k)\dots P(x_n|y_k)P(y_k) \tag{3.10}$$

Si riporta in 3.8 il funzionamento dell'algoritmo.

3.5.2 SVC

Il classificatore *Support Vector* si prefissa d trovare un iperpiano ottimale che divide al meglio le osservazioni fra le classi diverse del problema che si sta modellando. Un *iperpiano* è un piano di *n-1* dimensione che separa in due lo spazio n-dimensionale delle feature delle osservazioni. Un iperpiano in uno spazio a due dimensioni divide lo spazio delle feature con una retta, e una superficie in tre dimensioni, e così via. L'iperpiano ottimale è scelto tale per cui la distanza tra i punti di ogni spazio più vicini sia massimizzata. A questi punti vicini viene dato il nome di *support vector*. Si riporta un esempio in figura 3.9.

Per trovare l'iperpiano adatto si rende necessario un approfondimento. Come si evince nell'immagine 3.10 solo l'iperpiano C è adatto.

A livello matematico, uno spazio a due dimensioni può esser espresso da un vettore w indicante il coefficiente angolare, mentre la b è l'intercetta. Un qualsiasi

Figura 3.9: Svm con una retta che divide lo spazio a due dimensioni [1]

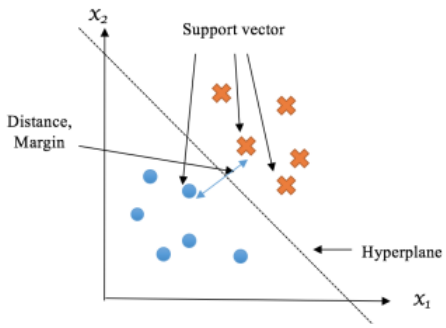
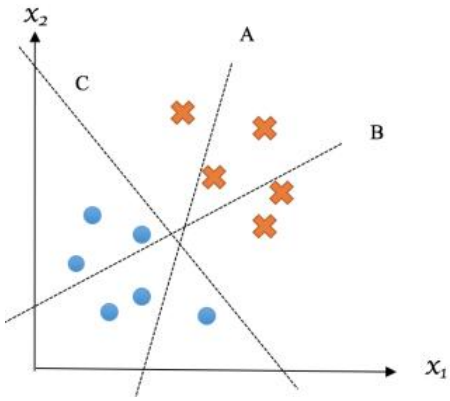


Figura 3.10: Svm possibili iperpiani per lo spazio a due dimensioni [1]

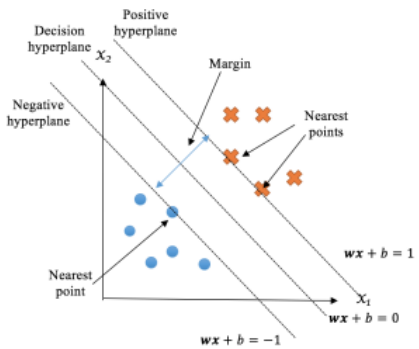


punto sull'iperpiano soddisfa l'equazione $wx + b = 0$. Un iperpiano è in grado di separare solo se si soddisfano due condizioni:

- per un qualsiasi punto x di una classe, soddisfa l'equazione $wx + b > 0$
- per un qualsiasi punto x di un'altra classe, soddisfa l'equazione $wx + b < 0$

Tuttavia possono non esser sufficienti per trovare l'iperpiano ottimale, poiché possono esistere molte soluzioni che verificano le due equazioni proposte. Se si ruota l'iperpiano infatti si possono trovare altre soluzioni che le verificano, ma non si ha un iperpiano ottimo. I punti, o punto, più vicini sul lato positivo vanno a creare un iperpiano parallelo all'*iperpiano decisionale*. Esso prende il nome di **iperpiano positivo**, in maniera analoga per i punti, o punto dal lato negativo prenderà il nome di **iperpiano negativo**. La distanza tra l'iperpiano positivo e negativo viene chiamata **margine**. Si parla di **iperpiano decisionale ottimale** quando il margine è massimizzato.

Figura 3.11: Gli iperpiani positivo, negativo e decisionale [1]



A livello matematico i due iperpiani positivo e negativo vengono descritti come segue:

$$wx^{(p)} + b = 1 \tag{3.11}$$

$$wx^{(n)} + b = -1 \tag{3.12}$$

dove $x^{(p)}$ è il campione oltre l'iperpiano positivo, $x^{(n)}$ è invece il campione oltre l'iperpiano negativo. La distanza fra un punto $x^{(p)}$ e l'iperpiano decisionale può esser calcolata come segue:

$$\frac{|wx^{(p)} + b|}{||w||} = \frac{1}{||w||} \tag{3.13}$$

In maniera equivalente per un punto $x^{(n)}$:

$$\frac{|wx^{(n)} + b|}{||w||} = \frac{1}{||w||} \tag{3.14}$$

Il margine diventa $\frac{2}{|w|}$. Si necessita di un'altra condizione, ovvero che nessun punto cada dentro i due iperpiani:

$$wx^{(i)} + b \geq 1 \; se \; y^{(i)} = 1 \tag{3.15}$$

$$wx^{(i)} + b \leq 1 \; se \; y^{(i)} = -1 \tag{3.16}$$

Si riporta che $(x^{(i)}, y^{(i)})$ è una osservazione, ciò può esser ulteriormente combinato come segue:

$$y^{(i)}(wx^{(i)} + b) \geq 1 \tag{3.17}$$

Per trovare l'iperpiano decisionale, si risolve il seguente problema di ottimizzazione:

- minimizzare $|w|$
- s.t $y^{(i)}(wx^{(i)}+b \geq 1)$, training set dato da $(y^{(1)}, x^{(1)}), (y^{(2)}, x^{(2)}) \ldots, (y^{(m)}, x^{(m)})$

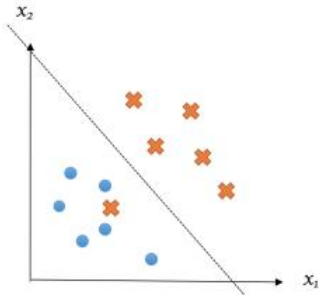
Il modello per apprendere e classificare un nuovo campione x', y' è dato da:

$$y' = \begin{cases} 1, \ se\, wx' + b > 0 \\ -1, \ se\, wx' + b < 0 \end{cases} \tag{3.18}$$

Inoltre, $|wx' + b|$ può esser raffigurata come la distanza dal punto cui si sta facendo la predizione, all'iperpiano decisionale ed è anche interpretato come la confidenza della predizione: più alto è il valore, più ulteriormente lontano è il punto predetto dall'iperpiano decisionale, quindi più certamente è sicura la predizione.

Si presenta ora una casistica particolare in figura 3.12, che però può esser decisamente frequente: La maggior parte dei problemi reali non sono necessariamente

Figura 3.12: Presenza di un outlier [1]



lineari. Questo implica che un iperpiano, più semplicemente una retta o una superficie possono non bastare per avere una divisione netta fra le classi. Bisogna introdurre un misura legata alla mal classificazione, data proprio dagli outlier. Il fine è quello di minimizzare questo errore. Si parla quindi di *hinge loss* per un campione $x^{(i)}$, l'errore legato alla mal classificazione:

$$\xi^{(i)} = \begin{cases} 1 - y^{(i)}(wx^{(i)} + b), \ se\, mal\, classificato \\ 0, \ altrimenti \end{cases} \tag{3.19}$$

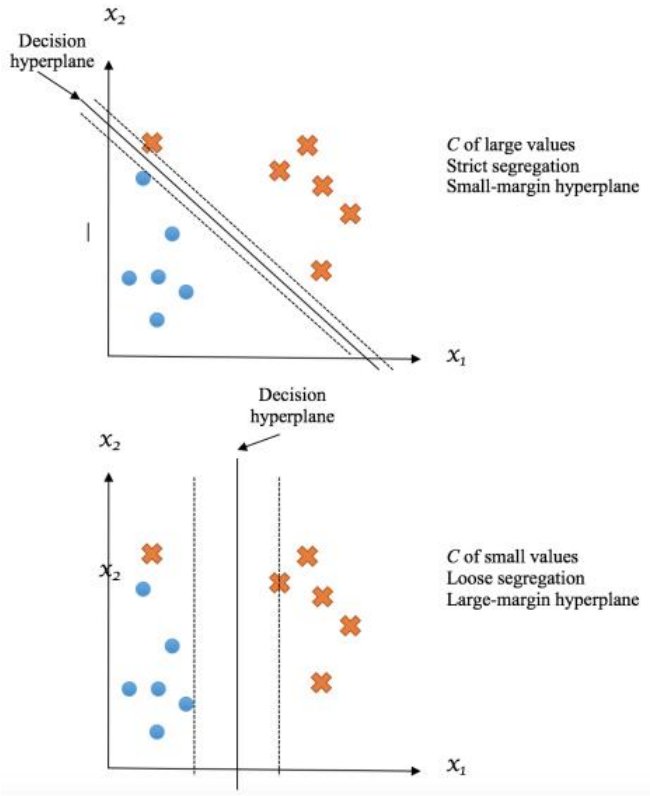
Dovendo ridurre $||w||$ l'obiettivo finale è quello di minimizzare la seguente funzione obiettivo, asandosi sul training set $(y^{(1)}, x^{(1)}), (y^{(2)}, x^{(2)}) \ldots, (y^{(m)}, x^{(m)})$ di m campioni:

$$||w|| + C\frac{\sum_{i=1}^{m} \xi^{(i)}}{m} \tag{3.20}$$

C è un iperparametro che controlla due termini: con un valore alto, la penalità diventa alta. Ciò implica che la regola per suddividere i campioni diventa più stringente e il modello tende a far overfit, qualche errore è permesso durante la fase di training. **C** elevato vuol dire bias ridotto, ma il modello soffre di varianza elevata.

Se **C** invece è sufficientemente piccolo, l'influenza dell'errore di mal classificazione diventa più basso. Il modello permette più dati mal classificati rispetto a un **C**

Figura 3.13: Trade off varianza-bias nel SVM[1]



elevato. Perciò, la separazione dei dati diviene meno stringente. Tale modello ha una varianza ridotta, ma bias elevato.

Si presenta quindi un trade-off nella scelta del parametro C, riportato in figura 3.13

3.5.3 SGD

Il metodo di Discesa del Gradiente Stocastico, SGD è un metodo di ottimizzazione usato in ambito machine learning, ma anche in un contesto di deep learning. Per apprendere tale metodo però è necessario riportare alcune definizioni utili per comprendere appieno il SGD.

Si parla di **metodi del primo ordine**, in riferimento a una funzione scalare $f : \Omega \to \mathbb{R}$ considerando il problema di ottimizzazione generale non vincolata:

$$argmin_{w \in \Omega} f(w) \tag{3.21}$$

I metodi iterativi per la risoluzione di tale problema si prefissano di generare una sequenza di soluzioni $w_0, w_1 \ldots$. I metodi del primo ordine sono per definizione

vincolati. Generano pertanto sequenze di w considerando solo il valore della funzione e del gradiente in punti differenti di Ω. Così, i metodi del primo ordine sono appropriati solo quando f è almeno differenziabile.

Si parla di *Discesa del Gradiente (GD)* in riferimento al metodo più semplice ed intuitivo del primo ordine per trovare i minimi di funzione. Partendo da un punto w e procedendo verso la direzione $-\nabla_w f$, chiamata anche *antigradiente*, è la direzione più veloce per la discesa col fine di trovare il minimo di f. Il metodo GD parte da un punto casuale scelto $w_0 \in \Omega$ e genera i successivi punti applicando un aggiornamento di questo tipo:

$$w_{t+1} = w_t - \alpha_t \nabla_{w_t} f \tag{3.22}$$

dove α_t è un parametro chiamato *steplength*,∇_{w_t} è il gradiente nel punto w_t. L'equazione precedente si può scrivere in maniera più compatta come:

$$G_{f,\alpha}(w) : \Omega \to \Omega \tag{3.23}$$

In conclusione a questo excursus teorico, il GD è un approccio greedy che cerca di minimizzare la funzione utilizzando come direzione quella dell'antigradiente, considerando che il gradiente, per natura, punta verso curve di livello superiori, in automatico l'antigradiente punta verso curve di livello inferiori, indi verso minimi locali/globali. Senza ulteriori assunzioni, il GD può risultare comunque costoso a livello computazionale. Si ricorda inoltre che il gradiente non è altro che la definizione generalizzata del concetto di derivata. In quanto tale, essa viene utilizzata in molti contesti di ottimizzazione, come il deblurring di immagini e la ricerca di minimi in funzioni.

SGD può essere utilizzato ovviamente anche in ambito di machine learning supervisionato. Si consideri una distribuzione ignota D su uno spazio definito da $Z = X\text{x}Y$ dove X è lo spazio delle feature in input, Y è lo spazio delle label associate all'input, ovvero le risposte corrette. Il fine del SGD nel machine learning è quello di costruire un modello che selezionato fra una moltitudine di modelli, riesca ad approssimare la distribuzione delle label date dalle feature. Idealmente si vorrebbe trovare il valore del parametro del modello w, minimizzando il *population risk*:

$$R(w) = \mathbb{E}_{z\tilde{}D} l(w; x) \tag{3.24}$$

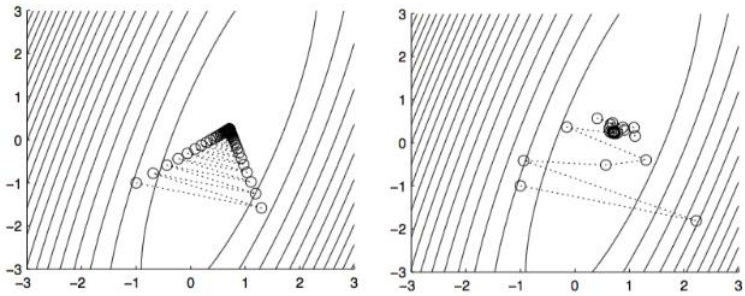
dove $l(w; z)$ è la loss function del modello su z, con parametro w. La scelta della loss function è cruciale per ottenere dei risultati buoni. Se la distribuzione dei dati D non è nota come spesso accade, si assume che un dato insieme $S = (z_1, \ldots, z_n)$,

sia quello definito, al posto del *population risk*, l'*empirical risk*, definito come:

$$R_S(w) = \frac{1}{n}\sum_{i=1}^{n} l(w; z_i) \tag{3.25}$$

Se si prova ad apprendere, minimizzare R_S con il GD si trova che il gradiente dell'empirical risk dipende da tutti gli n campioni. E fintanto che *più dati si hanno meglio è* ci ritrova a correre contro un muro. La chiave che sta al di fuori e che lega le osservazioni è proprio la somma. Pertanto il gradiente è un vettore che risulta come somma di n vettori gradienti, uno per esempio. Risulta ragionevole credere che si può ottenere una buona approssimazione del gradiente su un qualsiasi punto, prendendo un qualsiasi subset casuale di un certo numero b di campioni, andando ad aggiungere i vettori gradiente e scalando il risultato. Questo processo stocastico per la stima del gradiente rilancia il GD col nome di *Stochastic Gradient Descend*. Senza entrare troppo nel dettaglio, l'antigradiente non è l'unica soluzione circa la direzione di discesa per minimizzare la funzione in questione, come si riporta in 3.14. Si evince dall'immagine come, partendo entrambe dal punto (-1,-1) la direzione di destra arriva a convergenza molto prima rispetto all'andamento a sinistra, detto anche a zig-zag.

Figura 3.14: Esempio di differenti direzioni di discesa [5]



3.5.4 Random Forest

Random Forest [4] può essere usato per effettuare multiclassification, oppure in ambito di regressione. Il Random Forest viene applicato per alcuni motivi di seguito riportati: è possibile usarlo per la multiclassificazione e per la regressione; veloce da allenare e per effettuare delle predizioni; possiede solo 1 o 2 iperparametri ed è facile da parallelizzare.

Il Random Forest è un algoritmo basato su un insieme di alberi, cui ognuno di essi dipende da un insieme di variabili casuali. Più genericamente si parla di un vettore

di dimensione p, $X = (X_1, X_2 \ldots, X_p)^T$ rapresentante un input di una variabile. Con la variabile Y si rappresenta la risposta, ovvero l'output. Si assume che i dati seguano una distribuzione ignota $P_{XY}(X, Y)$. L'obiettivo è quello di trovare una funzione $f(X)$ che predica Y. La funzione di predizione viene determinata da una *loss function*, chiamata $L(Y, f(X))$; banalmente si cerca di minimizzare questo valore:

$$E_{XY}(L(Y, f(X))) \tag{3.26}$$

$L(Y, f(X))$ è una misura che indica di quanto $f(X)$ è vicina ad Y. Penalizza i valori di $f(X)$ che sono lontani da Y. Per la loss function solitamente si può scegliere la *squared error loss* per problemi di regressione:

$$L(Y, f(X)) = (Y - f(X))^2 \tag{3.27}$$

oppure nel caso di problemi di classificazione la 0/1

$$L(Y, f(X)) = I(Y \neq f(X)) = \begin{cases} 0 \, if \, Y = f(X) \\ 1 \, altrimenti \end{cases} \tag{3.28}$$

Concentrandosi sul lato classificazione, se l'insieme dei possibili valori di Y si chiama Υ, minimizzando $E_{XY}(L(Y, f(X)))$ per la loss function sulla classificazione si ottiene:

$$f(X) = argmax_{y \in \Upsilon} P(Y = y | X = x) \tag{3.29}$$

ovvero il Teorema di Bayes.

La f *ensemble costruct* è un insieme di *base learner* $h_1(x), \ldots, h_J(x)$. Essi sono una combinazione di quello che è il predittore $f(x)$, chiamato *ensemble predictor*. Nel contesto della classificazione $f(x)$ è la classe predetta più votata:

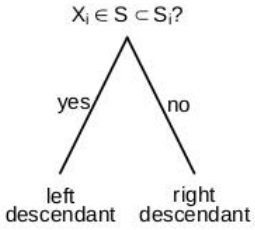
$$f(x) = argmax_{y \in \Upsilon} \sum_{j=1}^{J} I(y = h_j(x)) \tag{3.30}$$

Nel Random Forest il j-esimo base learner è un albero denotato come $h_j(X, \Theta_j))$, dove Θ_j è un insieme di variabili casuali. Ogni Θ_j è indipendente l'una dall'altra. Si precisa come la definizione di Random Forest è estremamente generale. Ci sono molte implementazioni in determinati modi.

Di seguito viene fatta una *introduzione agli alberi di classificazione*. Gli alberi del Random Forest sono basati su partizionamenti binari ricorsivi. Essi ripartiscono lo spazio dei predittori utilizzando una sequenza di split binari. La *radice* dell'albero comprende l'intero spazio dei predittori. I nodi finali si chiamano in gergo *terminal node* e formano la partizione finale dello spazio dei predittori. Ogni nodi all'interno

dell'albero possiede due figli, a destra e sinistra, coerenti con i valori dei predittori delle variabili. Un predittore per una variabile categorica X_i prende possibili valori da un insieme di categorie finite $S_i = s_{i,1}, \ldots, s_{i,m}$. Un nodo manda un subset di queste categorie $S \subset S_i$ al nodo di sinistra, il rimanente delle categorie a quello di destra, come si evince nell'immagine 3.15. In un contesto di classificazione, dove ci

Figura 3.15: Esempio di split nel Random Forest[4]



sono K classi, uno split tipico è dato dall'*indice di Gini*:

$$Q = \sum_{k \neq k'}^{K} \hat{p_k} \hat{p_{k'}} \tag{3.31}$$

dove $\hat{p_k}$ è la proporzione delle osservazioni della k-esima classe nel nodo:

$$\hat{p_k} = \frac{1}{n} \sum_{i=1}^{n} I(y_i = k) \tag{3.32}$$

Suddetta quantità restituisce una misura di quanto il nodo è, nell'ambito della classificazione, puro (ovvero un buon fit della classe), o impuro (pessimo fit della classe). Un nodo genera due discendenti, un subset delle categorie va al nodo di sinistra, il rimanente a destra. Definito comunque il criterio di split per i nodi figli, che si chiamano Q_L, Q_R e la loro grandezza come n_L, n_R lo split che si sceglie minimizza $Q_{split} = n_L Q_L + n_R Q_R$. Per una variabile categorica, i vari Q sono calcolati di modo che generino tutti i possibili modi di scelta di un subset di categorie da far andare poi nei nodi figli. In maniera ricorsiva, una volta scelto lo split, i dati vengono divisi in due partizioni, dopodiché il modello viene trattato nella stessa maniera rispetto a come si è partiti dal nodo radice. Si continua fino a che non viene incontrato un criterio di stop.

Il classificatore Random Forest genera quindi molti alberi decisionali. Ogni albero prende un campione da classificare, e in output risponde con una classe. La classe che prende più voti sarà la predizione del modello. Se gli alberi non sono correlati fra loro, in quanto entità separate gli uni dagli altri, si possono avere delle prestazioni incredibilmente alte. Se ci sono alberi che classificano male i rispettivi

campioni, ma ce ne sono di più che li classificano bene, andando a votazione alla fine, si sceglie l'albero con la predizione giusta. Ciò ovviamente non funziona se gli alberi forniscono predizioni errate, e andando a votare alla fine, si sceglie la classe sbagliata per la predizione.

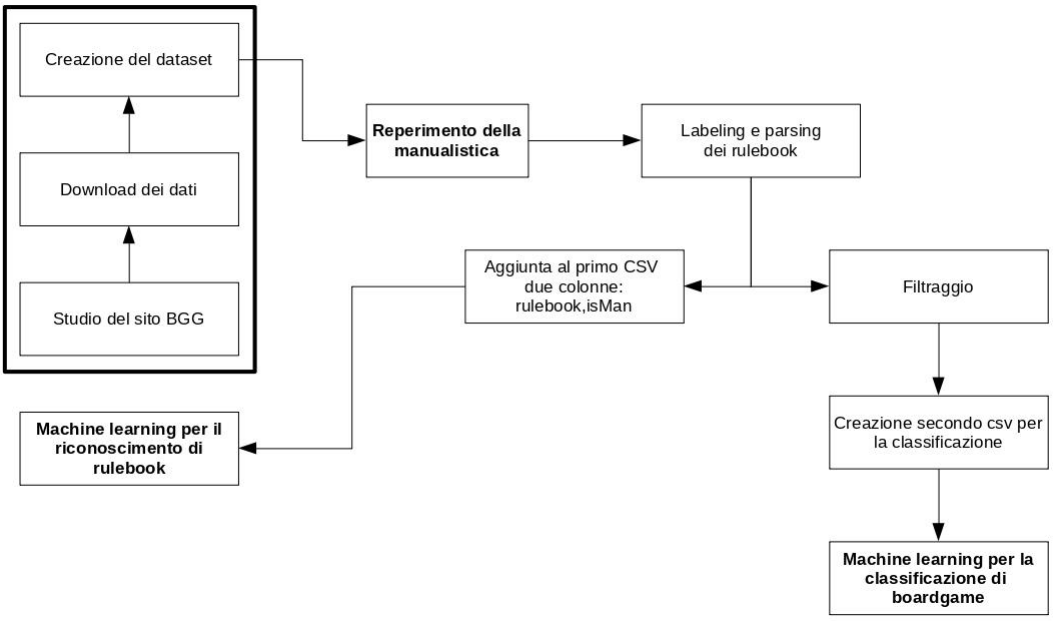
Parte II

Download dati e rulebook, creazione del dataset ed uso della Text analytics e Machine learning

Capitolo 4

Creazione del dataset

Figura 4.1: Fase studio del sito, download dati e creazione csv



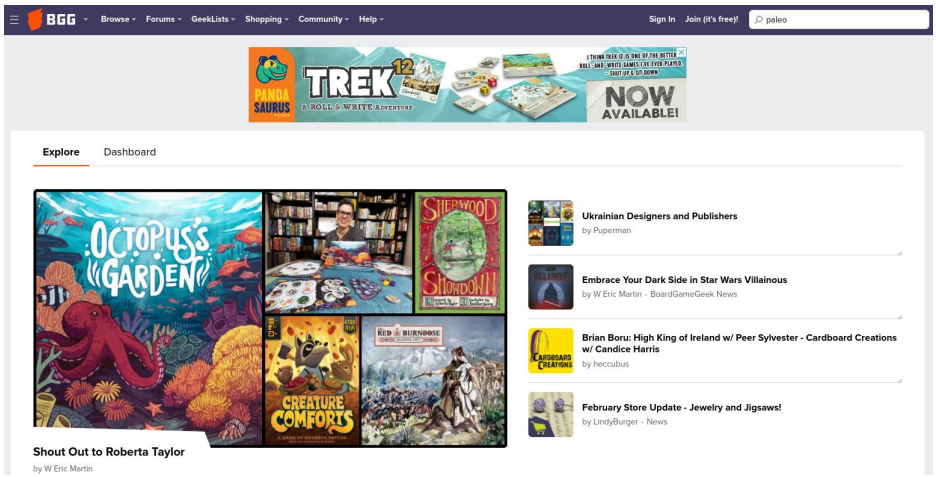
La prima parte del progetto di tesi è finalizzata prevalentemente per fare in modo che si abbia un background solido inerente alla teoria, alle tecnologie apprese che sono state utilizzate, alla natura dei problemi trattati. In particolar modo il precedente capitolo 3 è stato necessario per poter illustrare la teoria dietro il Machine Learning.

In questo capitolo iniziale della seconda parte, viene mostrato l'intero processo che ha portato ai risultati ottenuti dai dati, scaricati direttamente da BGG. Rappresenta la fase di download dei dati con la successiva creazione del dataset. Ciò verrà spiegato

lungo tutto questo capitolo, partendo dalla struttura del sito BGG nel dettaglio, quale è la struttura dati che si cela dietro ogni gioco su BGG, fino alla spiegazione dell'algoritmo per il download massiccio dei giochi nel ranking.

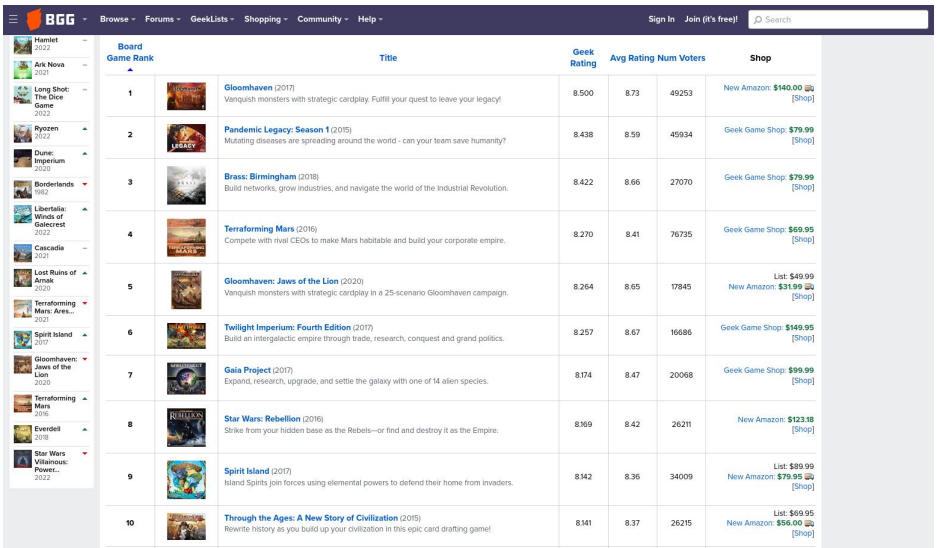
L'immagine 4.1 vuole mostrare il punto di partenza, interno al riquadro. Il sito BGG è estremamente vasto. Esso è diviso in 3 siti, citati per completezza, ma che non sono stati usati, e sono: RPGGeek, VideoGameGeek, Geek Events. La barra dei menù in alto nel sito possiede alcune opzioni importanti: l'iscrizione al sito e il login, assieme a una barra di ricerca. Si presenta come nell'immagine 4.2

Figura 4.2: Struttura della prima pagina del ranking BGG



Si mostra ora il ranking globale di cui dispone il sito, nell'immagine 4.3

Figura 4.3: Struttura main page BGG

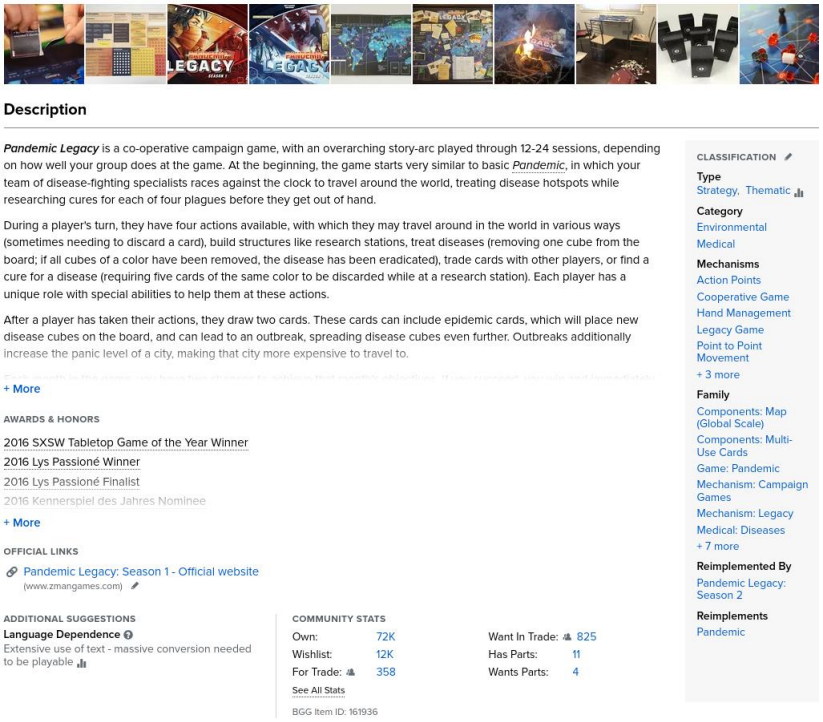


Per ovvie ragioni di spazio e di eccessiva lunghezza, sono stati riportati nell'immagine i primi 10 giochi del ranking. Ogni pagina del ranking possiede 100 giochi, per un totale di più di 50k giochi. Si ricorda che il ranking è in continua evoluzione. Pertanto nuovi boardgame possono essere inseriti in qualsiasi momento, che siano originali o fanmade, e vecchi boardgame possono essere rimossi. Andando più nello specifico dentro ogni gioco, è possibile avere più informazioni, come riporta l'immagine 1.3, nel capitolo 1. Per ogni gioco sono presenti i file caricati dagli utenti, le recensioni e le espansioni, foto e video inerenti al boardgame.

4.1 Struttura di un boardgame su BGG

Una panoramica del gioco in sè viene data nella scheda overview, come riporta l'immagine 4.4.

Figura 4.4: Dati nella scheda Overview di pandemic legacy su BGG



Dall'immagine si evince come sia denso di informazioni ogni singolo boardgame. Si riportano alcune delle informazioni estratte dagli XML in quanto ritenute le più importanti, e su cui si ha lavorato lungo il progetto, che sono:

- id

- nome del gioco
- descrizione
- categorie di gioco
- meccaniche di gioco
- ...

Ne sono state estratte delle altre, ma che non sono utilizzate ai fini della text analytics e machine learning, come per esempio, il numero minimo/massimo di giocatori, la durata media di una partita, la dipendenza dalla lingua, l'anno di pubblicazione, gli artisti e le espansioni. Alcuni di tali dati sono più adatti per una analisi statistica descrittiva per ogni singolo gioco, invece che per uno studio basato su text analytics.

Le immagini riportate offrono una panoramica ad alto livello. Una volta utilizzata la procedura per il download dei dati dai boardgame, quello che si ottiene è un insieme di file XML, la cui struttura viene riportata, in maniera semplificata, nell'immagine 4.5, in riferimento sempre al boardgame Pandemic.

Si può notare, come espresso nell'apposito capitolo 2 teorico, come ogni gioco abbia questa struttura a livello di XML. Ogni file di questo tipo è stato possibile ottenerlo proprio grazie alle API che i gestori di BGG offrono. Le informazioni di interesse sono all'interno dei tag, cui è stato possibile estrarre grazie a delle librerie python.

4.2 Download massiccio dei boardgame da BGG

Illustrata a livello di XML nella 4.1 la forma di un singolo boardgame, viene riportata adesso la procedura, a livello di algoritmo ad alto livello, utilizzata per scaricare in massa gli XML dei boardgame interni al ranking.

Nell'immagine 4.6 viene riportato un flow-chart semplificato di quello che è l'algoritmo implementato in python per scaricare i boardgame. Se si vogliono scaricare solo alcune pagine del ranking, è possibile di fatto selezionare l'intervallo di interesse. Una volta creata la directory cui si vanno a salvare i file XML scaricati, si procede andando a contattare il sito BGG, per poi iniziare a scorrere le pagine, una ad una, e per ognuna di esse all'interno del ranking si va a scaricare il documento XML di ogni gioco all'interno della pagina in questione. Una volta scaricato e salvato nell'apposita directory ogni XML della rispettiva pagina BGG, si procede con la pagina del ranking successiva, fino all'esaurimento dell'intervallo.

Figura 4.5: XML del boardgame Pandemic Legacy


```xml
<?xml version="1.0" ?>
<items termsofuse="https://boardgamegeek.com/xmlapi/termsofuse">
    <item id="161936" type="boardgame">
        <name sortindex="1" type="primary" value="Pandemic Legacy: Season 1"/>
        <name sortindex="1" type="alternate" value="Pandemic Legacy:  Saison 1"/>
        <name sortindex="1" type="alternate" value="Pandemic Legacy: 1. Évad"/>
        <name sortindex="1" type="alternate" value="Pandemic Legacy: 1a. Temporada"/>
        <name sortindex="1" type="alternate" value="Pandemic Legacy: Rok 1"/>
        <name sortindex="1" type="alternate" value="Pandemic Legacy: Seizoen 1"/>
        <name sortindex="1" type="alternate" value="Pandemic Legacy: Sezon 1"/>
        <name sortindex="1" type="alternate" value="Пандемия: Наследие"/>
        <name sortindex="1" type="alternate" value="パンデミック：レガシー シーズン1"/>
        <name sortindex="1" type="alternate" value="瘟疫危機：承傳"/>
        <name sortindex="1" type="alternate" value="□□□ □□□: □□ 1"/>
        <description>Pandemic Legacy is a co-operative campaign game, with an overarching story-arc played through
12-24 sessions, depending on how well your group does at the game. At the beginning, the game starts very similar
to basic Pandemic, in which your team of disease-fighting specialists races against the clock to travel around the
world, treating disease hotspots while researching cures for each of four plagues before they get out of
hand.&amp;#10;&amp;#10;During a player's turn, they have four actions available, with which they may travel around
in the world in various ways (sometimes needing to discard a card), build structures like research stations, treat
diseases (removing one cube from the board; if all cubes of a color have been removed, the disease has been
eradicated), trade cards with other players, or find a cure for a disease (requiring five cards of the same color
to be discarded while at a research station). Each player has a unique role with special abilities to help them at
these actions.&amp;#10;&amp;#10;After a player has taken their actions, they draw two cards. These cards can
include epidemic cards, which will place new disease cubes on the board, and can lead to an outbreak, spreading
disease cubes even further. Outbreaks additionally increase the panic level of a city, making that city more
expensive to travel to.&amp;#10;&amp;#10;Each month in the game, you have two chances to achieve that month's
objectives. If you succeed, you win and immediately move on to the next month. If you fail, you have a second
chance, with more funding for beneficial event cards.&amp;#10;&amp;#10;During the campaign, new rules and
components will be introduced. These will sometimes require you to permanently alter the components of the game;
this includes writing on cards, ripping up cards, and placing permanent stickers on components. Your characters can
gain new skills, or detrimental effects. A character can even be lost entirely, at which point it's no longer
available for play.&amp;#10;&amp;#10;Part of the Pandemic series&amp;#10;&amp;#10;</description>
        <yearpublished value="2015"/>
        <minplayers value="2"/>
        <maxplayers value="4"/>
        <link id="1084" type="boardgamecategory" value="Environmental"/>
        <link id="2145" type="boardgamecategory" value="Medical"/>
        <link id="2001" type="boardgamemechanic" value="Action Points"/>
        <link id="2023" type="boardgamemechanic" value="Cooperative Game"/>
        <link id="2040" type="boardgamemechanic" value="Hand Management"/>
        <link id="2824" type="boardgamemechanic" value="Legacy Game"/>
        <link id="2078" type="boardgamemechanic" value="Point to Point Movement"/>
        <link id="2004" type="boardgamemechanic" value="Set Collection"/>
        <link id="2008" type="boardgamemechanic" value="Trading"/>
        <link id="2015" type="boardgamemechanic" value="Variable Player Powers"/>
         ...
    </item>
</items>
```


Si riporta nell'immagine 4.7 l'elenco di 10 file XML dentro la directory. Per ovvi motivi di comodità non è stato possibile riportare l'intero contenuto della cartella. Alla fine di questa fase, si ha una moltitudine di XML, cui sarà possibile effettuare il parsing, per poter così creare il dataset finale.

4.3 Costruzione del dataset

Una volta conclusa la fase di download, si possiedono ora una moltitudine di file XML strutturati come in figura 4.5, si procede con il parsing di ogni file.

L'immagine 4.8 illustra il lato algoritmico della procedura implementata in python, per estrarre le informazioni dai tag all'interno di ogni XML dei giochi. Una volta verificata la directory che contiene i file XML, per ognuno di essi si cominicia a navigare l'albero, alla ricerca dei tag di interesse che possano contenere le informazioni desiderate. Di seguito se ne riportano alcuni fra i tanti ricavati:

- *type*, il tipo di boardgame, che può essere il gioco base o un'espansione

Figura 4.6: Algoritmo download XML da BGG

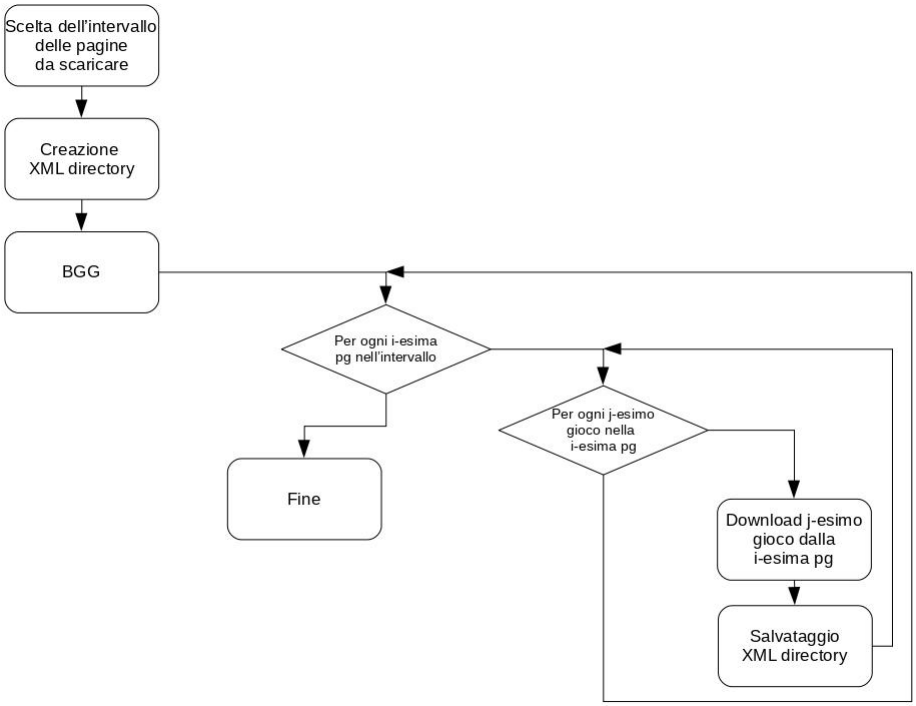
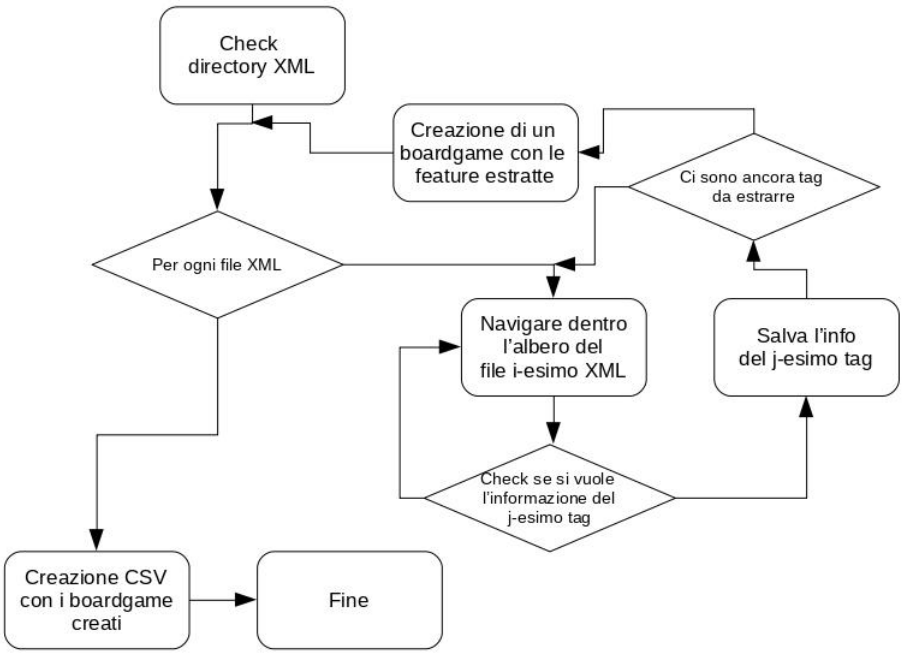


Figura 4.7: 10 file XML dentro la directory

```
-rw-r--r-- 1 nds nds  23285 ott  8 22:01 game_rank_10000_id_2986.xml
-rw-r--r-- 1 nds nds  25190 ott  8 22:01 game_rank_10001_id_166888.xml
-rw-r--r-- 1 nds nds  18533 ott  8 22:01 game_rank_10002_id_9150.xml
-rw-r--r-- 1 nds nds  49298 ott  8 22:01 game_rank_10003_id_284751.xml
-rw-r--r-- 1 nds nds  18811 ott  8 22:01 game_rank_10004_id_1551.xml
-rw-r--r-- 1 nds nds  14489 ott  8 22:01 game_rank_10005_id_291066.xml
-rw-r--r-- 1 nds nds  16786 ott  8 22:01 game_rank_10006_id_248072.xml
-rw-r--r-- 1 nds nds  17146 ott  8 22:01 game_rank_10007_id_176678.xml
-rw-r--r-- 1 nds nds  25584 ott  8 22:01 game_rank_10008_id_7739.xml
```

- *name*, il nome del boardgame
- *description*, la descrizione del gioco, senza entrare nel dettaglio. Solitamente offre una panoramica generica del funzionamento del gioco, l'obiettivo e l'ambientazione
- *minplayer*, il numero minimo di giocatori necessari per giocare
- *maxplayer*, il numero massimo di giocatori per il gioco
- *boardgamecategory*, alcuni elementi che ricorrono spesso all'interno del gioco, vengono caratterizzati con delle parole chiave, che rappresentano la categoria di appartenenza del gioco

Figura 4.8: Algoritmo per la creazione dei dati su file CSV



- *boardgamemechanic*, le meccaniche che fanno da sfondo al gioco. Necessarie per il funzionamento del boardgame stesso, definiscono cosa bisogna fare per poter giocare
- ...

Sono stati riportati alcuni campi, esportate nel CSV, ma che non sono stati poi utilizzati ai fini della text analytics e machine learning. Si può finalmente mostrare il dataset come ottimo punto di partenza nell'immagine 4.9. Il dataset presenta 50k righe. Si conclude così la fase preparatoria per i dati. Con questi si può già cominciare con della text analytics e machine learning. Si è scelto però di seguire l'ordine degli eventi. Infatti dopo questa fase, prima dello studio vero e proprio, si è continuato andando a capire e implementare una procedura per scaricare i manuali.

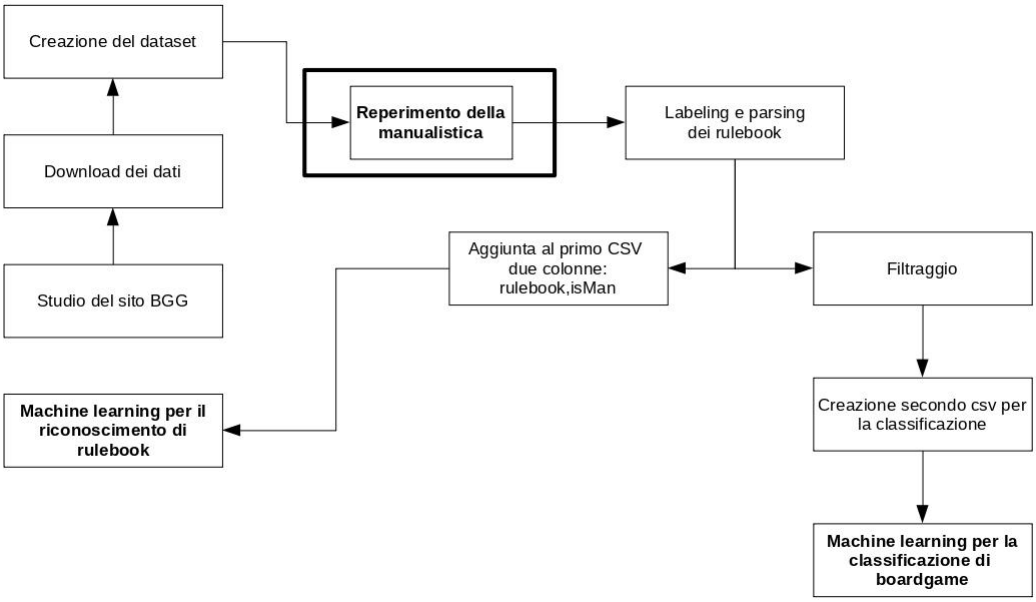
	A	B	C	D	E	F	G	H	I
1	type	id	name	description	yearpublished	minplayers	maxplayers	playingtime	minplaytime
2	boardgame	235254	Finn Billiards	Finn Billiards plays somewhat like Mölkky on a wooden surface.The playing area is a large	2017	2	16	15	15
3	boardgame	158891	Krynea Battlefield	Krynea Battlefield is a card game of strategy and tactics set in Krynea, a world of fantasy a	2014	1	2	60	60
4	boardgame	8818	Gopher It!	This children's card game is about gophers collecting their winter food supply.Each supply c	1999	2	4	10	10
5	boardgame	268276	Star Trek Chrono-Trek	In Star Trek Chrono-Trek, a time-travel game similar to Chrononauts but set in the Star Trek	2019	2	6	45	15
6	boardgame	16271	No Trumpets No Drums: The Vietnam War 1965-1975	No Trumpets, No Drums is a game of the war in Vietnam. Two players or teams representin	1982	2	2	240	240
7	boardgame	292971	Ettin	From the mists of war, a pair of great nations rise together like a two-headed giant. They ma	2020	2	8	60	0
8	boardgame	42881	Trenches of Valor	Where did the great commanders of World War 2 get their start? Fighting in the trenches of	2009	2	2	25	25
9	boardgame	5929	Clash of Empires: The Battle for France 1914	This game, first published in The Wargamer Magazine (#58), is a simulation of the beginning	1986	2	3	120	120
10	boardgame	251685	Tinian: The Forgotten Battle	Most people have never heard of the Battle of Tinian. Overshadowed by the Battle of Saipa	2019	1	2	900	120
11	boardgame	187488	Dark Mages	In a distant land covered in lush forests and bordered by hills and mountains, magicians bat	2016	2	8	60	20
12	boardgame	354	Stick 'Em	In the trick-taking card game Stick 'Em, first released as Sticheln, players seek to gather po	1993	3	8	60	30
13	boardgame	335	Breakthru	Part of the 3M Bookshelf Series.Breakthru is a modern member of the Hnefatafl family of ga	1965	2	2	15	15
14	boardgame	16030	Volley & Bayonet	[From the back of the Manual]Take control of the great battles of the 18th and 19th centuri	1994	2	4	240	240
15	boardgame	252579	Gorbachev: The Fall of Communism	In Gorbachev: The Fall of Communism you are thrust into the Soviet leadership, trying to pu	2018	1	1	60	30
16	boardgame	9169	Platsch!	The hippos want to go swimming... but they need life preservers! Players compete to get rid	2002	2	4	10	10
17	boardgame	9170	Aladdins Erbe	The players have to possess as many valuable treasures from Aladdins heritage as possibl	1987	2	4	90	90
18	boardgame	9171	Jades	A Yahtzee®-like game in which players take turns rolling (7) 12-sided dice to score various	1998	1	8	90	90
19	boardgame	9172	Die Naschbären	From the box:From the tree cabin the smart raccoons see exactly where the delicious tidbit	2002	2	6	10	10
20	boardgame	9175	The 24 Card Game	This is similar to 24 Game but public domain and using a standard deck of playing cards. Th	0	2	4	20	20
21	boardgame	9176	King's Men	A gameboard is divided into 105 squares, the 49 center squares printed in grey. These are c	1937	2	2	20	20
22	boardgame	9178	Clean Water: The Water Pollution Game	From the rulebook: "In the game, each player assumes the role of a water pollution control c	1970	2	4	30	30
23	boardgame	9180	Bug Strife	Bug Strife is a tabletop game that reminds me of many real time strategy games for the PC	1999	2	6	30	30
24	boardgame	9182	Friedensreiter	Westfaelischer Friede in Germany. The war is over and there are a many messages to trans	1999	3	6	120	120
25	boardgame	9183	Gembel	Gembel is played on a 4×4 board. There are 16 playing pieces, some of which are small, an	1999	2	2	30	30
26	boardgame	135649	Tales & Games: The Three Little Pigs	As a little pig, your dearest wish is to build a strong and beautiful house in which you can	2013	2	5	20	20
27	boardgame	9184	Spawn PowerCardz	From the Rulebook: Spawn PowerCardz is a collectible super powered combat card game b	1995	2	2	30	30
28	boardgame	9185	Input	You have to build numbers out of cards. On each card there is pattern you have to build. Yo	1998	2	3	20	20
29	boardgame	9186	Monday Morning Manager	Monday Morning Manager is a fairly detailed baseball simulation. Each player designs their	1980	2	2	30	30
30	boardgame	27525	Get In That Tub	The game involves running a device down a plastic strip. That device would then speak phr	1969	2	4	45	45
31	boardgame	27526	New Frontier Game	Game from 1962 that is similar to Monopoly and that is critical of the Kennedy Administr	1962	2	4	60	60
32	boardgame	27527	A Visit To Mother Goose	Listed on the box as a game for little people. This game featured very unique player pieces.	1943	2	4	0	0
33	boardgame	27528	Petticoat Junction	This game comes from a television show of the same name which showed in the early 1960	1963	2	4	45	45
34	boardgame	27529	Sub Attack	Object of Sub Attack is to either sink enemy ships or get them safely to port depending on	1965	2	2	60	60
35	boardgame	27530	The Undersea World Game	Game for pre-school children which requires kids to spin a spinner and then match the spin	1971	2	4	30	30
36	boardgame	27531	I Dood It	Game based on Red Skelton's radio personalities. The game plays similar to Blitz.	1947	2	4	45	45
37	boardgameexpansion	27533	Morbus Terribilis	Expansion kit of 55 new cards for "Obscura Tempora". Some are of the same kind previous	0	2	6	60	60
38	boardgame	27534	Stage Fright	Try to guess the four or five highlighted words or phrases by acting out a charades-style p	0	0	0	0	0
39	boardgame	27535	Quiz Show	A collection of three outrageous activities designed to test your retro pop-culture knowledge	0	0	0	0	0

Figura 4.9: CSV prodotto

Capitolo 5

Estensione dataset: reperimento manuali

Figura 5.1: Fase di reperimento della manualistica



Il capitolo 4 ha reso noto come siano stati possibili i download dei file XML di ogni boardgame. In particolar modo è stata fatta presente la struttura del sito BGG in ogni sua parte. Successivamente, una volta ottenuti i file XML, sono state estratte le informazioni dai vari tag di interesse, quali il nome del boardgame, la descrizione del gioco, le categorie e meccaniche di appartenenza, durata di una partita, numero min/max di giocatori, ... Fatto ciò è stato possibile creare un csv con tutte le feature precedentemente citate. Si è ottenuto così un dataset da 50k righe da poter

utilizzare immediatamente per fare machine learning. Si è scelto prima però di illustrare come sia stato possibile scaricare i manuali, e mantenere la fase di text analytics e classificazione con machine learning per ultima, in modo da discutere alla fine dei risultati ottenuti.

In questa prima parte capitolo, come mostra l'immagine 5.1, il focus è sui manuali. Viene spiegato come sono stati ricavati, le strategie che hanno portato ad una soluzione al momento efficace, e che potesse funzionare senza l'intervento dell'uomo. In conclusione, si illustra come sia stato possibile estrapolare delle informazioni dai file così ottenuti, che possono essere pdf/doc/docx.

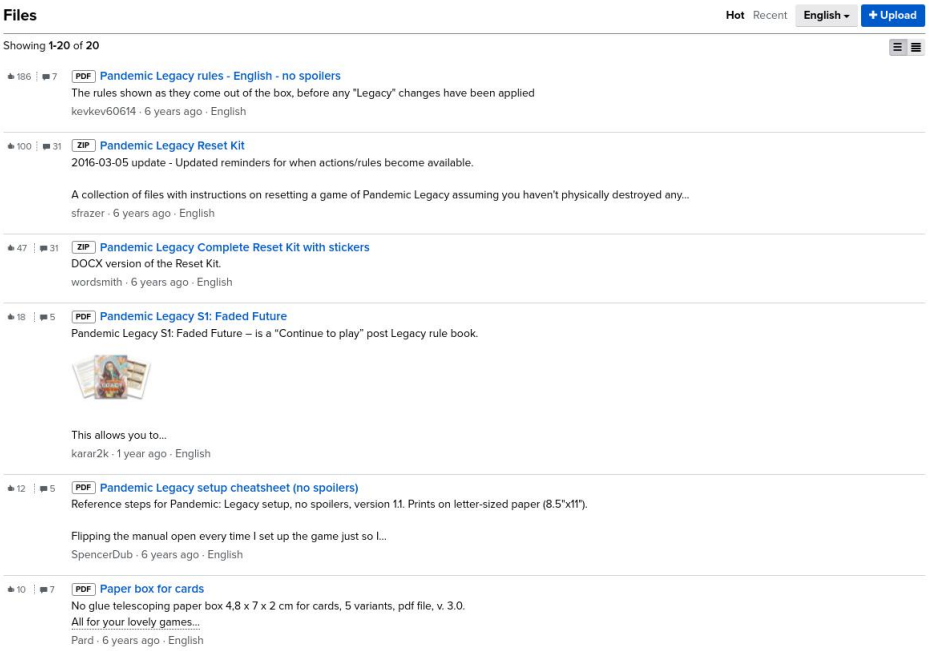
5.1 Problematiche legate alla manualistica

Il download dei manuali per ogni gioco è stato un problema che si è rivelato essere estremamente ostico. Infatti si sono riportate le seguenti osservazioni fatte, prima ancora di partire per risolverlo:

- dove reperire la manualistica
- ogni boardgame possiede o meno il rulebook
- quale è, e se esiste, lo standard che BGG segue per la manualistica
- quanto è affidabile il manuale che si trova, se si trova
- quali sono i rischi in cui si può incorrere scaricando materiale da BGG

Partendo per gradi, il reperimento della manualistica è stato reso possibile grazie a BGG. Il sito infatti possiede, all'interno di ogni gioco presente nel ranking, una scheda denominata *Files*. All'interno di quest'ultima è possibile vedere un elenco di file caricati, di varie dimensioni e formati, come riporta l'immagine 5.2. Si evince come prima cosa la necessità obbligatoria di essere registrati al sito per caricare file inerenti ai boardgame. Continuando a sviscerare il problema, ci si accorge come molti giochi possiedono una moltitudine di file caricati nella rispettiva pagina. Quantità che si assottiglia più il boardgame è in basso rispetto al ranking globale di BGG. Un gioco da tavolo che occupa le prime posizioni, presenta feedback positivi, ottime recensioni, molte persone che lo hanno comprato, e di conseguenza altrettanti utenti che magari hanno caricato il manuale. L'esempio in figura 5.2 mostra come tale gioco, secondo nel ranking, possieda molti file caricati da molti utenti. Questo perché il boardgame è famoso e possiede quantità elevate di feedback positivi. Questo è l'unico standard che possiede BGG per l'upload dei file. Chiunque interno alla

Figura 5.2: File caricati dalla community di BGG per il gioco Pandemic Legacy



community ne può caricare, e i gestori del sito non si assumono alcuna responsabilità circa il materiale caricato, poiché a priori all'interno di esso potrebbe esserci qualcosa di indesiderato. Ecco perché questo rappresenta un rischio il scaricare del materiale dal sito. L'affidabilità del manuale trova voce nel fatto che, sempre nell'immagine 5.2, siano presenti i *thumbs up*, i feedback positivi (molto elevati). Più sono alti, più è probabile che sia effettivamente un manuale a tutti gli effetti. Si prenda per esempio sempre l'immagine 5.2 il primo file abbiamo 186 thumbs up.

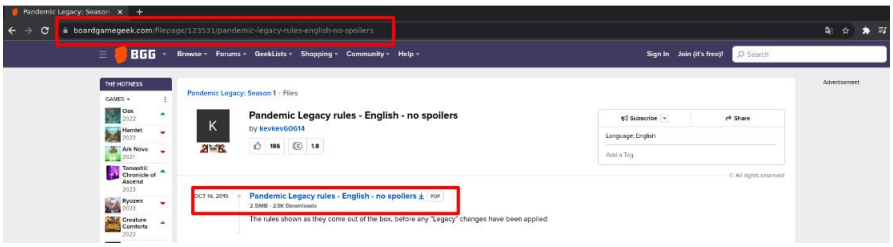
Alla luce di tutte queste osservazioni, si è giunti a conclusione che un ottimo punto da cui partire è quello di scaricare, per tutti i boardgame che lo possiedono, il primo file in inglese con il maggior numero di thumbs up. Per fare ciò è necessario un ulteriore passo poiché anch'esso risulta non banale.

5.2 Preparazione per il download dei manuali

Avendo compreso ora le problematiche legate ai manuali, si riportano i primi passi seguiti, volti ad avere più rulebook possibili. Si riporta l'immagine 5.3. Essa mostra evidenziata due link fondamentali:

- il primo link in alto nell'immagine, è ottenuto grazie alla scheda Files, cliccando sul primo file inglese con thumbs up più elevati di tutti gli altri file, interno

Figura 5.3: Primo file per numero di thumbs up per Pandemic Legacy



alla scheda Files. Viene chiamato per evitare fraintendimenti, *link del primo file*

- il secondo in basso è il link che punta al file da scaricare, questo è il cuore del problema. Viene chiamato, per evitare fraintendimenti *link alla risorsa.*

Andando più nel dettaglio, si ricorda nuovamente che quest'ultimo link alla risorsa non è visibile. Per poterlo vedere bisogna aver effettuato il login. Facendo una prova, una volta effettuato, si viene indirizzati sui AWS (Amazon Web Service, piattaforma cui BGG si appoggia), da cui è possibile scaricare il file.

Il primo step per il download dei manuali è stato quello di ricavare il *link del primo file* per ogni gioco. Si va a contattare il sito BGG per richiedere tale link, per ogni pagina di ogni boardgame su BGG. Successivamente si va a salvare su file, in aggiunta al nome che si darà di default al manuale scaricato.

Si riporta l'algoritmo in figura 5.4. Anch'esso funziona per intervalli. Una volta scelto, dopo che si contatta il sito BGG, si crea un txt vuoto. Successivamente per ogni pagina, e per ogni gioco al suo interno, sempre tramite API si ricava per ognuno di essi il link del primo file, e si genera un nome predefinito che si darà al documento che si scaricherà nelle fasi successive. Tale nome sarà *rb_id_xxxxx.ext* dove vi sarà sostituito l'ID del gioco. Infine si aggiungono queste due info al txt creato inizialmente. Si continua così per ogni gioco dentro ogni pagina dell'intervallo fino all'esaurimento. Avere un elenco di tutti i link al primo file è fondamentale per fare in modo che Firefox, quando sarà guidato da Selenium, abbia già i link alle pagine che dovrà visitare. Così facendo il browser, dopo aver effettuato ovviamente il login, andrà a scorrere tutti i link del primo file di ogni gioco, otterrà il link alla risorsa, che è il manuale, e poi lo scaricherà.

In figura 5.5 si riporta il file prodotto alla fine dell'algoritmo. A coppie di righe, rappresentano il nome predefinito da attribuire ai file scaricati, e i rispettivi link del primo file, dove andare a scaricare il manuale vero e proprio. Si può notare come ci siano alcune righe che possiedono, al posto del nome, la stringa *no rulebook avaiable*

Figura 5.4: Algoritmo per la creazione del txt con i nomi predefiniti e link del primo file per ogni boardgame

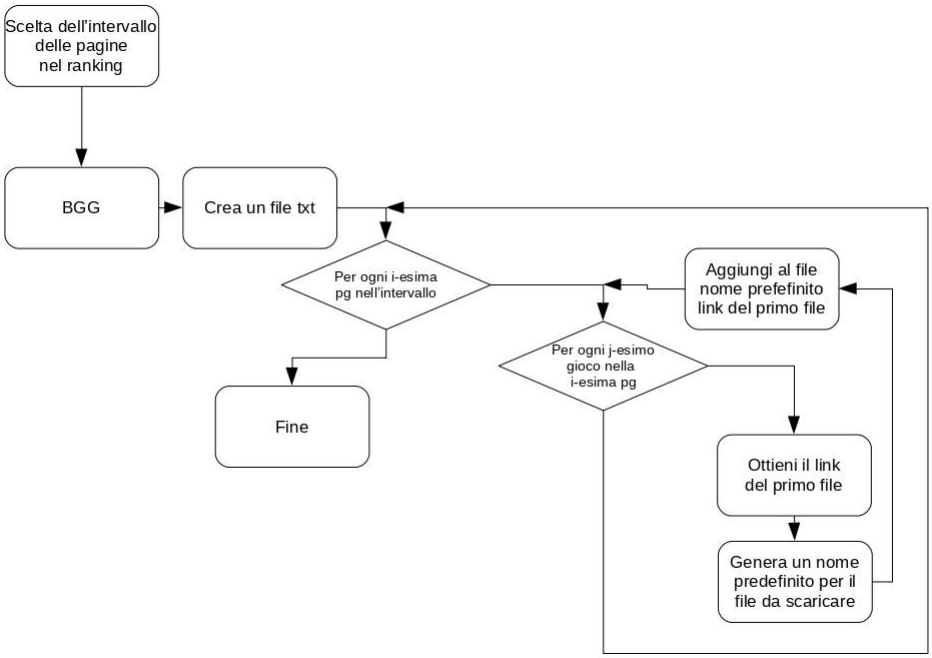


Figura 5.5: Esempio file txt prodotto dopo l'algoritmo.

```
rb_id_26528
No rulebooks avaiable for: 26528
rb_id_38149
No rulebooks avaiable for: 38149
rb_id_118705
No rulebooks avaiable for: 118705
rb_id_29069
No rulebooks avaiable for: 29069
rb_id_11642
No rulebooks avaiable for: 11642
rb_id_313723
https://boardgamegeek.com/filepage/211025/samoa-english-rulebook
rb_id_26159
No rulebooks avaiable for: 26159
rb_id_10472
https://boardgamegeek.com/filepage/111734/east-red-rules-summary
rb_id_28052
No rulebooks avaiable for: 28052
rb_id_284987
No rulebooks avaiable for: 284987
```

for: xxxxxx al posto di un link del file. Questo banalmente perché non è detto che ci siano dei file nella scheda Files dei boardgame.

Si riportano alcune osservazioni: si è preferito utilizzare un approccio verticale piuttosto che orizzontale. Questo perché è risultato più conveniente ricavare prima tutti i link del primo file e poi scaricare tutti i manuali. Non è risultato possibile invece ricavare anche tutti i link alla risorsa per ogni boardgame, poiché bisogna essere loggati. Anche in questo caso il problema più grande è che i link sono dinamici e momentanei, dopo breve tempo, scadendo, non sono più utilizzabili. Ecco perché si è creato tale file txt. Perché così facendo inizialmente si effettua il login. Poi successivamente si può navigare grazie a Selenium, all'interno del sito BGG, alla

pagina del primo file di ogni boardgame interno al txt. Poi si procede con il download, e il link alla risorsa sarà visibile, perché loggati. Finalmente così facendo si avrà a disposizione il rulebook.

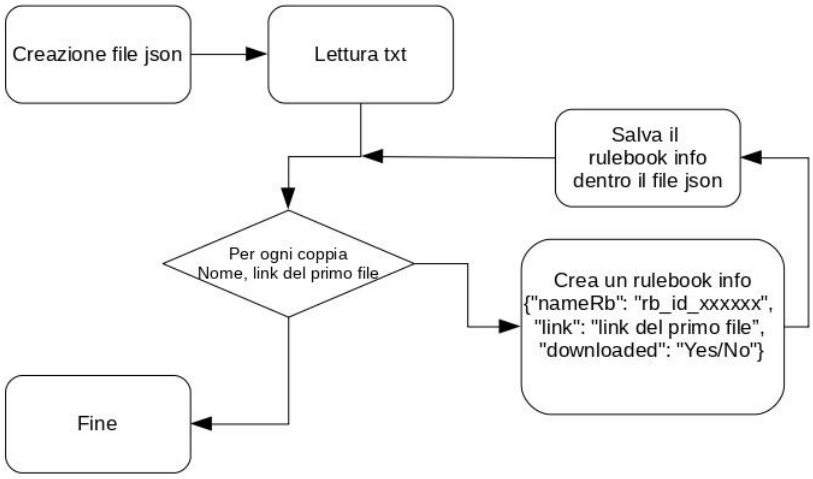
5.3 Creazione dell'entità Rulebook Info

Una volta creato il file txt, e popolato con le informazioni citate in 5.2, si prosegue il lavoro. Si è deciso, per questioni pratiche, di costruire una entità che potesse rappresentare un manuale. Questa entità, a livello di codice, non è altro che un oggetto che permette di astrarre l'entità manuale. Ogni rulebook infatti viene definito, in base a questa astrazione, da tre parametri, che sono:

- nome utilizzato per il file rulebook
- link del primo file
- status download. Ovvero se il manuale è stato scaricato o no

Questa astrazione non è altro che il contenuto del file txt citato nella sezione 5.2 utilizzabile in una maniera più comoda a livello di codice. Il materiale interno al file di testo viene elaborato, e salvato dentro un file json. Questa operazione risulta molto comoda poiché andando a lavorare con Pandas, esistono metodi molto efficaci e comodi per andare a leggere all'interno di un file json e convertire il tutto in un dataframe.

Figura 5.6: Creazione del file json



L'immagine in figura 5.6 mostra l'algoritmo. Il file txt viene letto, per ogni riga e la sua successiva, viene creata l'entità rulebook info, che non è altro (a livello

python) che una rappresentazione diversa delle informazioni interne al file txt, con l'aggiunta appunto di un booleano che indica se il manuale è già stato scaricato oppure no (valore di base inizialmente). Questo procedimento, comunque molto rapido, è stato reso necessario in quanto ritenuto più utile convertire il contenuto del file txt in un file json, al cui interno sono presenti dei dizionari, che a livello python sono facilmente interpretabili e leggibili.

Figura 5.7: Esempio di un file json contenente dei rulebook info


```
{"nameRb": "rb_id_26528", "link": "No rulebooks avaiable for: 26528", "downloaded": "No"}
{"nameRb": "rb_id_38149", "link": "No rulebooks avaiable for: 38149", "downloaded": "No"}
{"nameRb": "rb_id_118705", "link": "No rulebooks avaiable for: 118705", "downloaded": "No"}
{"nameRb": "rb_id_29069", "link": "No rulebooks avaiable for: 29069", "downloaded": "No"}
{"nameRb": "rb_id_11642", "link": "No rulebooks avaiable for: 11642", "downloaded": "No"}
{"nameRb": "rb_id_313723", "link": "https://boardgamegeek.com/filepage/211025/samoa-english-rulebook",
 "downloaded": "No"}
{"nameRb": "rb_id_26159", "link": "No rulebooks avaiable for: 26159", "downloaded": "No"}
{"nameRb": "rb_id_10472", "link": "https://boardgamegeek.com/filepage/111734/east-red-rules-summary",
"downloaded": "No"}
{"nameRb": "rb_id_28052", "link": "No rulebooks avaiable for: 28052", "downloaded": "No"}
{"nameRb": "rb_id_284987", "link": "No rulebooks avaiable for: 284987", "downloaded": "No"}
{"nameRb": "rb_id_88881", "link": "No rulebooks avaiable for: 88881", "downloaded": "No"}
{"nameRb": "rb_id_194301", "link": "https://boardgamegeek.com/filepage/143486/riftwalker-official-rule
book", "downloaded": "No"}
{"nameRb": "rb_id_1574", "link": "https://boardgamegeek.com/filepage/178713/tactics-ii-1958-rules", "d
ownloaded": "No"}
{"nameRb": "rb_id_37872", "link": "No rulebooks avaiable for: 37872", "downloaded": "No"}
{"nameRb": "rb_id_6200", "link": "No rulebooks avaiable for: 6200", "downloaded": "No"}
{"nameRb": "rb_id_20181", "link": "No rulebooks avaiable for: 20181", "downloaded": "No"}
{"nameRb": "rb_id_38806", "link": "No rulebooks avaiable for: 38806", "downloaded": "No"}
{"nameRb": "rb_id_33463", "link": "No rulebooks avaiable for: 33463", "downloaded": "No"}
{"nameRb": "rb_id_14607", "link": "No rulebooks avaiable for: 14607", "downloaded": "No"}
{"nameRb": "rb_id_5662", "link": "No rulebooks avaiable for: 5662", "downloaded": "No"}
```


L'immagine riportata 5.7 mostra come effettivamente sia più pulita l'informazione. Ogni riga rappresenta una astrazione di un manuale di un boardgame, fornito di tre parametri che sono il nome del file, il link del primo file del boardgame dentro la pagina Files, e lo status ovvero se il file è già stato scaricato o no.

5.4 Algoritmo per il download dei rulebook

Ottenuto il file json in figura 5.7, ora si può procedere con il download dei manuali vero e proprio. Selenium, come già citato nel capitolo 2, è un'interfaccia di controllo, che permette di essere utilizzata senza l'intervento manuale di un operatore. Selenium si appoggia a un generico User Agent, UA. In questo progetto di tesi UA è stato ricoperto dal browser Mozilla. Grazie a delle impostazioni predefinite, Selenium in combinazione con Mozilla, sono stati in grado di scaricare una mole di manuali molto elevata.

In figura 5.8 si mostra l'algoritmo per il download della manualistica. Una volta letto il file json con all'interno i dizionari rappresentanti ognuno di essi un Rulebook Info, vengono settate le impostazioni per selenium legate al browser Firefox. Queste impostazioni permettono di tarare alcuni parametri essenziali, e se ne riportano

alcuni: la possibilità di usare Firefox senza l'interfaccia grafica ma in background. Così facendo il sito, in caso di eventuali controlli circa sulla responsiveness non riesce a capire il comportamento del browser. Un'altro parametro importante è la navigazione in incognito.

Una volta partito il browser che si dirige verso la pagina di login, se dovessero esserci problemi di un qualsiasi tipo, l'algoritmo termina. Invece in caso dovesse andare tutto per il verso giusto, vengono chieste all'utente username e password per accedere al proprio profilo su BGG. Ovviamente è necessario in precedenza aver registrato un account sul sito per potervi accedere. Se il login per qualche motivo non dovesse andare a buon fine, l'algoritmo termina.

Una volta effettuato il login, il browser prosegue in maniera automatica con gli step successivi. Per ogni rulebook info, viene fatto il check se il link non è disponibile. In tal caso si va al rulebook info successivo. Altrimenti viene fatto il check successivo per capire se è già stato scaricato. Nel caso lo fosse, si procede al rulebook info successivo. Altrimenti finalmente si può scaricare il manuale. Viene impostato un delay casuale. Questo ritardo è fondamentale per avere un po' di tempo tra una richiesta e l'altra. In parole povere si tenta di emulare il comportamento umano, che cambia da una pagina all'altra. Si è sperimentato che in assenza di questo delay il sito blocca la procedura, resistuendo ad alto livello il codice 400 HTTP, con la conseguenza del blocco temporaneo dell'account.

Dopo il delay, il browser procede alla pagina puntata dal link del primo file. Se questo link non dovesse esserci, si prosegue con il rulebook info successivo, segnalando l'assenza del link. In caso dovesse esserci, il browser finalmente si trova in una posizione in cui risulta essere nella pagina al cui interno è presente il link alla risorsa, ovvero il manuale. E questo link, avendo una sessione di login attiva, è visibile. Finalmente non resta che scaricare il file, pdf/doc che sia.

La procedura continua ad andare avanti in maniera iterativa, fino a che non viene esaurito il file contenente i rulebook info.

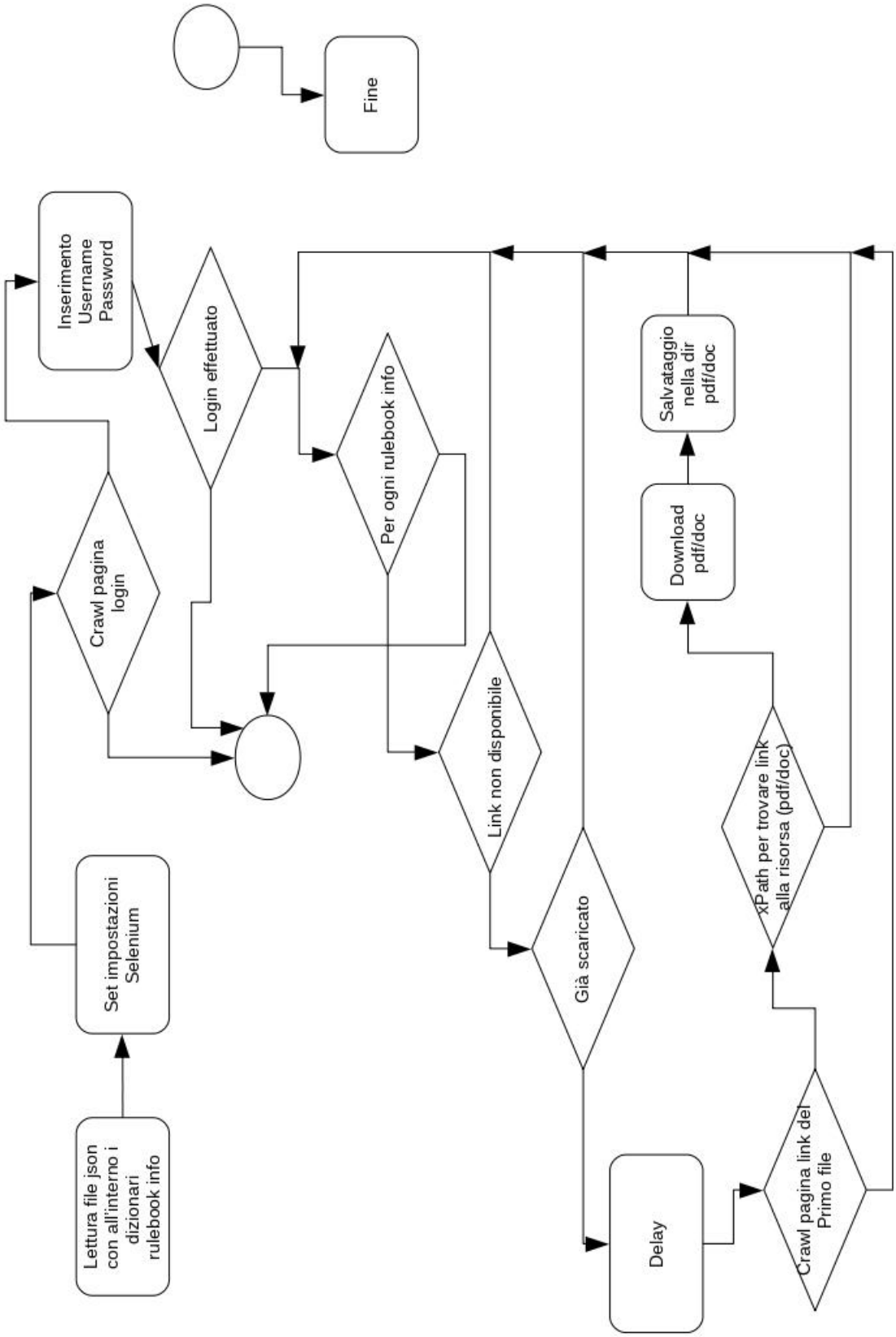


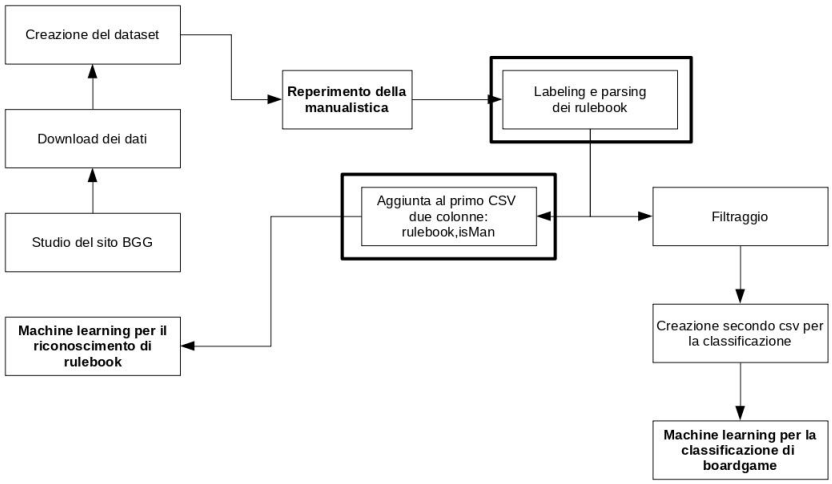
Figura 5.8: Algoritmo per il download dei dati

Capitolo 6

ML per il riconoscimento di rulebook

Nel capitolo 5 si sono mostrate le potenzialità di Selenium, strumento che permette la gestione automatizzata di uno User Agent, un browser in questo caso. Usando Firefox infatti, è stato possibile scaricare i manuali per ogni gioco. Nonostante però il ranking sia composto da circa 50k giochi, non si sono reperiti purtroppo altrettanti manuali. Il motivo è da ricercare nella popolarità dei boardgame, che diminuisce man mano che il gioco da tavolo scende di ranking. In aggiunta a quest'ultimo, proprio perché poco popolare, il boardgame magari non viene nemmeno comprato. Di conseguenza nessuno carica il manuale, che di fatto non si trova nella pagina del gioco da tavolo.

Figura 6.1: Fase di labeling dei rulebook



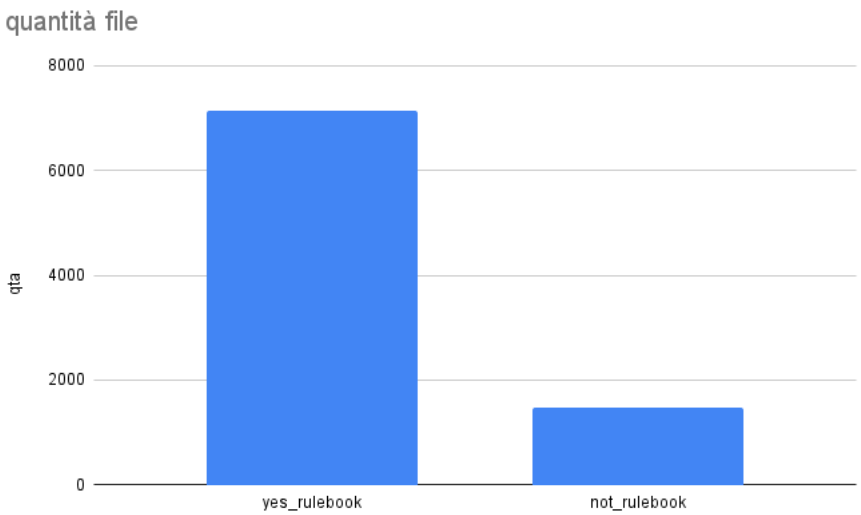
Nell'immagine 6.1 si evidenziano le fasi che vengono trattate in un primo momento.

Il riconoscimento, sempre in tale immagine, non è evidenziato poiché sarà trattato solo dopo aver illustrato i 3 step precedenti necessari per comprendere alcune dinamiche.

6.1 Labeling dei rulebook

Scaricata la manualistica il lavoro deve continuare. L'approccio circa la scelta dei manuali per indice di gradimento elevato sicuramente è una ottima strategia, ma deve essere verificata. Sono stati scaricati *circa 8.8k manuali complessivamente*. Di questi banalmente non è detto che tutti lo siano. La fase di labeling consta *nell'aprire ogni file scaricato e verificare che effettivamente sia un manuale o no*. La label che identifica se un manuale è considerato tale oppure no, è quella identificata dal nome *isMan*, un booleano presente anche nei csv che verranno, che assume valore True o False a seconda che il documento sia o no un rulebook.

Figura 6.2: Risultati labeling dei manuali scaricati

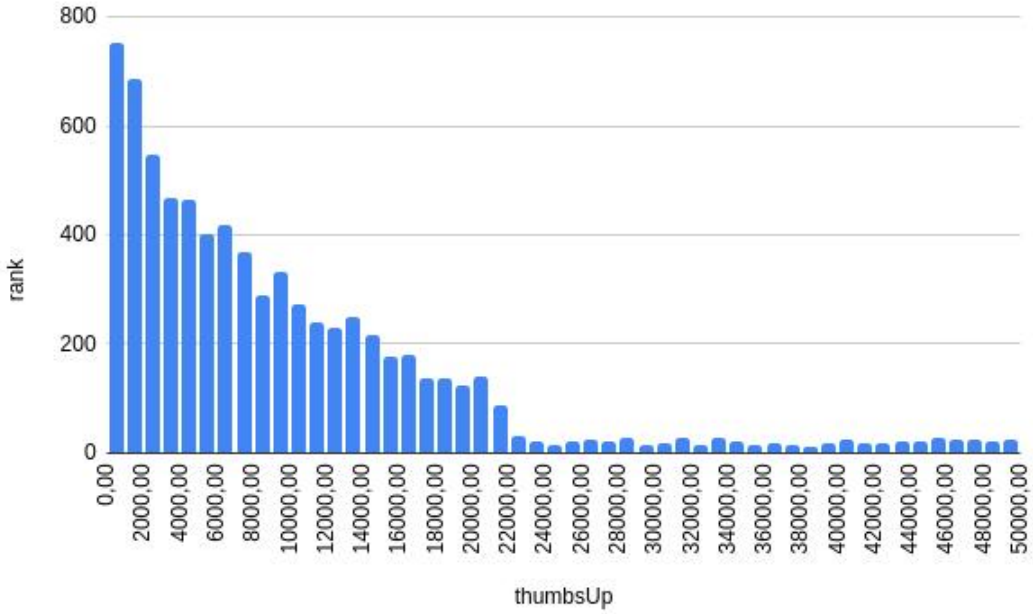


Si evince dall'immagine 6.2 come dei 8.8k manuali, poco più di 7.3k siano stati etichettati come manuali, il rimanente 1.5k no. In proporzione sono degli ottimi risultati, più dell'80% sono manuali effettivi. I rimanenti non lo sono per i motivi di seguito esposti:

- al posto di manuali ci sono documenti con all'interno le FAQ inerenti al gioco
- la plancia da gioco, una screen o una foto dei componenti del gioco invece che il manuale

- in alcuni boardgame vengono caricati invece che il manuale, ad esempio, le schede di gioco di alcuni personaggi del boardgame, non da considerarsi come manuali veri e propri

Figura 6.3: Istogramma thumbs up



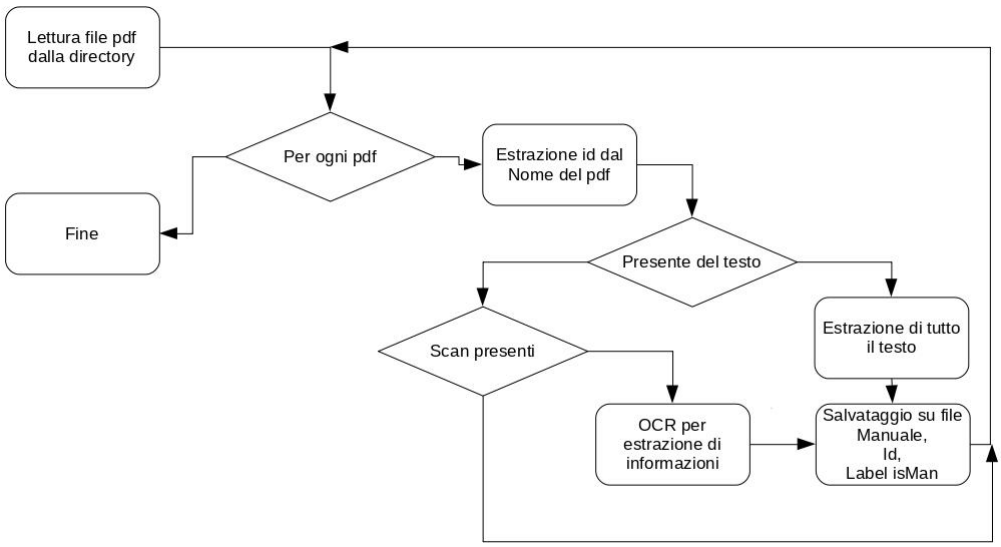
Nell'immagine 6.3 si riporta l'istogramma di frequenza dei thumbs up. Si può notare come gli intervalli il cui ranking arriva fino al 20k, i giochi abbiano dei thumbs up maggiori di 150. Ciò è un buon sintomo circa il fatto che ci sia un minimo di affidabilità nei manuali caricati dagli utenti.

6.2 Rulebook parsing e aggiunta di rulebook,isMan

Nella sezione 6.1 si sono riportate alcune osservazioni circa il labeling manuale fatto dall'uomo inerente ai rulebook. Si procede ora con il parsing dei manuali, iniziando dai pdf.

Algoritmo iterativo, come illustra l'immagine 6.4. Per ogni pdf all'interno di una cartella predefinita si procede estraendo l'id interno al nome del file. Questo id è una sequenza numerica che identifica in maniera univoca il boardgame sia come file, che come gioco all'interno del csv la cui creazione è stata illustrata nella sezione 4.3. Così facendo, una volta ricavata l'informazione dal file, sarà possibile molto comodamente andarla ad inserire nel csv senza problemi.

Figura 6.4: Algoritmo lettura informazioni dai pdf



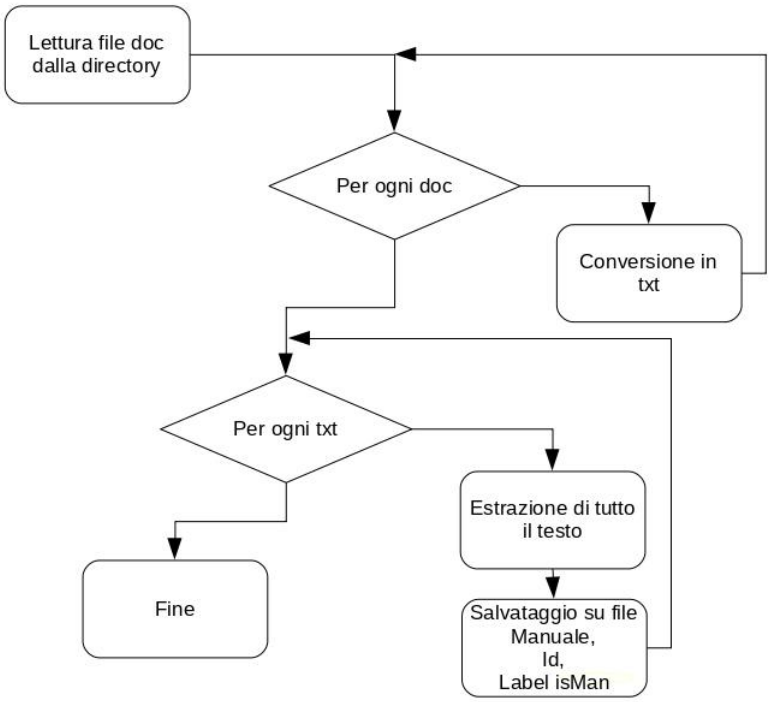
Se nel file è presente del testo allora viene estratto tutto, dopodiché vengono salvate le tre informazioni necessarie: *contenuto del manuale, id, isMan (label indicante se il documento è un manuale o no)*. Se non è presente del testo allora significa che sono presenti delle scansioni. Ciò rende necessario l'uso di OCR. Come prima, viene estratto il contenuto del pdf e poi vengono salvate le tre informazioni appena citate. Se non sono presenti, allora il file può avere qualche errore di codifica o errore generico, non viene estrapolato nulla e si passa semplicemente al file pdf successivo.

Si continua ad estrarre informazioni dai manuali, questa volta dai doc. Anche l'algoritmo in figura 6.5 è iterativo. Per ogni file doc interno alla cartella predefinita, viene aperto e convertito in txt. Questo step risulta più conveniente in quanto leggere informazioni da un file di testo è più semplice rispetto alla lettura da un doc.

Finita la conversione, per ogni txt viene estratta tutta l'informazione possibile, per poi salvare le tre infromazioni citate in precedenza: contenuto del manuale, id e label isMan.

Con le informazioni a disposizione in questo momento, è possibile estrarre dal dataset da 50k tutti i campioni che dispongono solamente del manuale. Così facendo si ottiene un dataset da 8.8k righe ideale per effettuare text analytics e quindi successivamente machine learning per riconoscere ciò che è un manuale da ciò che non lo è.

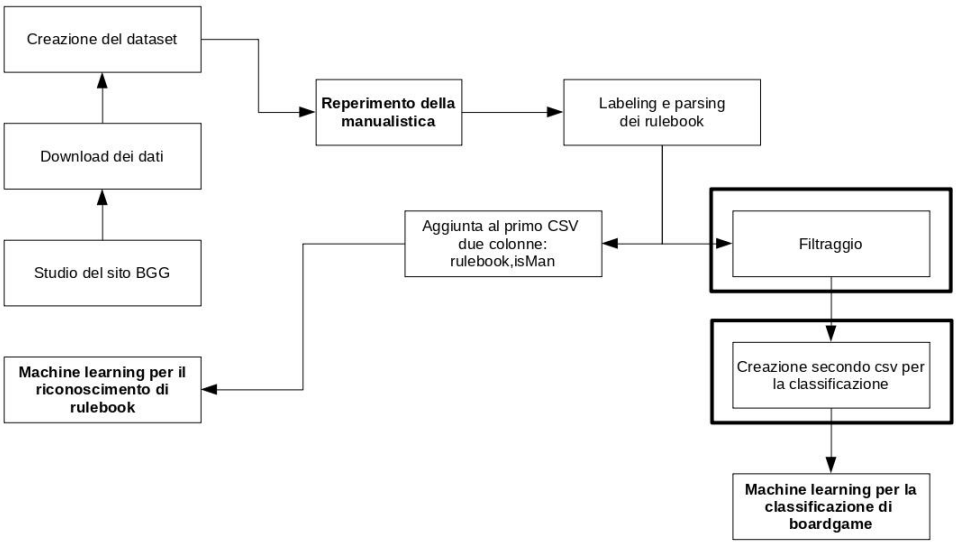
Figura 6.5: Algoritmo lettura informazioni dai doc



6.3 Filtraggio

Una volta parsati ed estrapolata l'informazione dai rulebook, è stata necessaria una fase di filtraggio, mostrata nel workflow in figura 6.6. Pur avendo etichettato True/False manualmente ogni rulebook scaricato, è necessaria questa fase di filtraggio dei manuali veri e propri, la cui feature isMan è settata a True. Viene proposta in due modi: *manuale,automatica.* La prima consiste nell'apertura del csv, controllare il campo dove è presente il testo del rulebook e fare manualmente un check con lo status, ovvero se è un manuale oppure no. Pur sembrando una ripetizione della sezione 6.1, in realtà questa fase è necessaria poiché dopo il parsing dei documenti, purtroppo non c'è alcuna garanzia a sostegno del fatto che il contenuto sia leggibile. Infatti, andando a studiare meglio il contenuto parsato di tutti i manuali che effettivamente lo sono, si è scoperto come purtroppo alcuni di essi, dopo la fase di parsing, siano diventati illeggibili. La causa di ciò è ignota, e può dipendere da molteplici fattori: chi ha convertito il file, chi lo ha creato, inserimento di caratteri che sono stati mal decodificati in fase di parsing, software con cui si è creato il pdf/doc ...

Figura 6.6: Fase filtraggio e successiva aggiunta di informazioni al precedente csv



6.3.1 Filtraggio manuale dei rulebook

Il filtraggio manuale consta semplicemente nell'andare ad aprire il csv e rimuovere manualmente tutte le righe che pur avendo un manuale vero e proprio, è oggetto di problemi appena citati. Fatto ciò, si evince come dalla quantità di 7.3k dopo la fase di labeling, a quella di filtraggio manuale, siano rimasti utilizzabili poco meno di 6k rulebook effettivi dei rispettivi giochi.

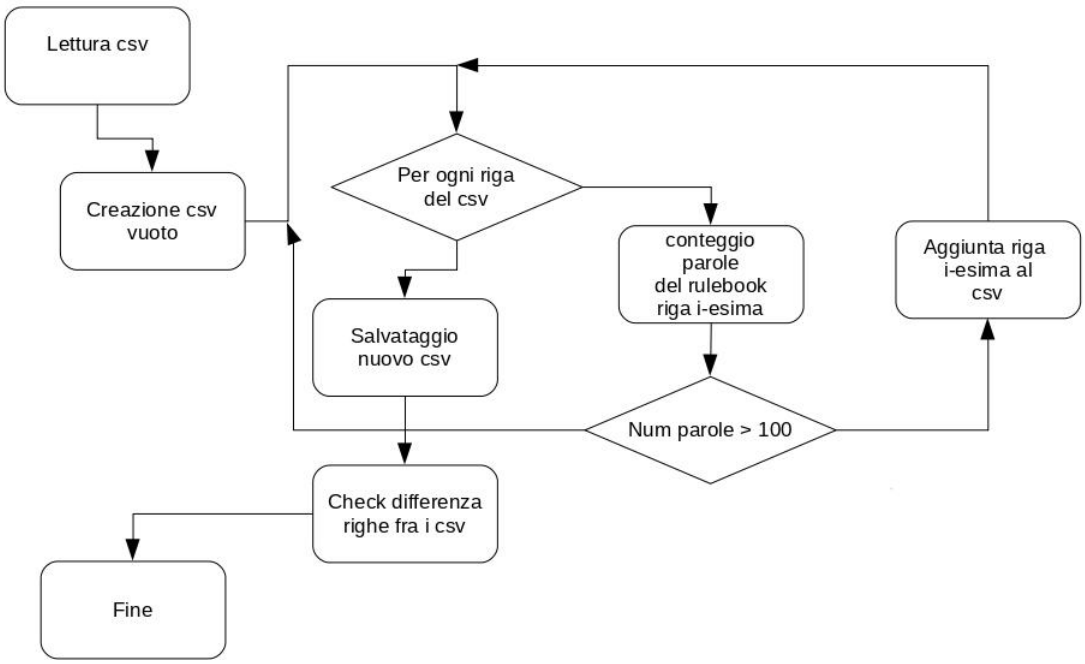
Figura 6.7: Fase filtraggio manuale dei rulebook

La figura 6.7 vuole riportare i vari numeri dei manuali dopo le varie fasi. Inizialmente dopo la fase di labeling se ne avevano 7.3k. Dopo la rimozione manuale, che ha portato a togliere 1.4k rulebook, se ne sono ottenuti 5.9k. Concretamente si sono rimosse appunto 1.4k righe dal dataset, poiché non essendo il manuale utilizzabile, allora non ha più utilità mantenere tutta l'informazione legata al gioco il cui manuale non è utilizzabile. Tutti questi rulebook, pur etichettati come tali, risultavano inutili per le motivazioni già espresse nel capitolo attuale, di conseguenza sono stati rimossi.

6.3.2 Filtraggio automatico dei rulebook

Questa sezione riporta un algoritmo naive che permette il filtraggio automatico. Lo scopo è quello di filtrare automaticamente dal file csv i boardgame con il rulebook inutilizzabile andando a contare semplicemente il numero di parole interne a ogni rulebook e vedere se il conteggio supera una certa soglia. Essa è stata fissata a 100 parole e motivata successivamente.

Figura 6.8: Algoritmo filtraggio

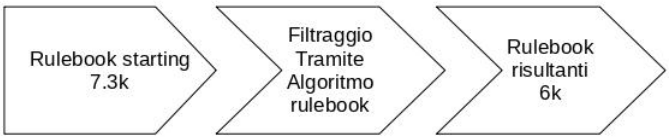


Nell'immagine 6.8 si riporta la procedura usata. Una volta letto il csv con anche le due colonne *rulebook, isMan* aggiunge nella sezione 6.2, viene creato un dataset vuoto. Esso verrà popolato man mano solo con le righe del dataset iniziale che l'algoritmo ritiene opportune in quanto possiedono un rulebook utilizzabile ai fini del machine learning.

L'algoritmo scorre il dataset su ogni riga. Effettua il conteggio delle parole che possiedono solo caratteri dell'alfabeto classico all'interno del campo rulebook. Se il numero di parole risulta più grande di 100, allora viene aggiunto al dataset vuoto. Altrimenti si prosegue con la riga successiva.

Al termine della procedura viene effettuato il conteggio delle righe di entrambi i csv che hanno la label isMan a True, per vedere quante righe del csv iniziale col manuale sono state mantenute nel csv copia.

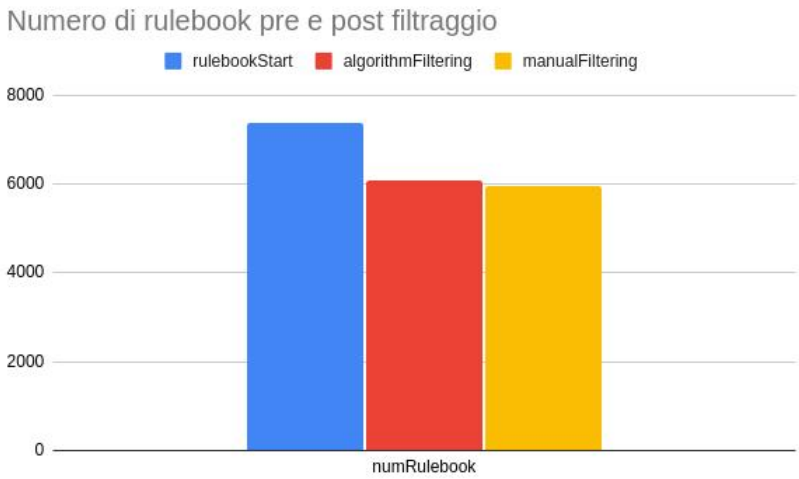
Figura 6.9: Fase filtraggio automatico, tramite algoritmo, dei rulebook



Viene fatta una osservazione circa il numero di parole presenti mediamente in un rulebook: dopo la fase di labeling, che ha portato ad aprire ogni singolo file, ci si è resi conto anche facendo un po' una stima grossolana, che un centinaio di parole come minimo fossero necessarie per un manuale. Questo perché ogni volta che si apre un manuale, bisogna effettuare una lettura rapida ma non troppo approfondita. Si sono riscontrati manuali di gioco con poche parole, ma comunque superiori al centinaio.

In figura 6.9 si nota come la procedura abbia portato a scartarne 1.3k righe dal dataset, ovvero tutti boardgame che hanno un manuale inutilizzabile. Risultanti sono le 6k righe del dataset che possiedono un rulebook effettivamente utilizzabile a tutti gli effetti.

Figura 6.10: Barplot differenza dei rulebook prima del filtraggio e dopo, sia manuale che automatico)

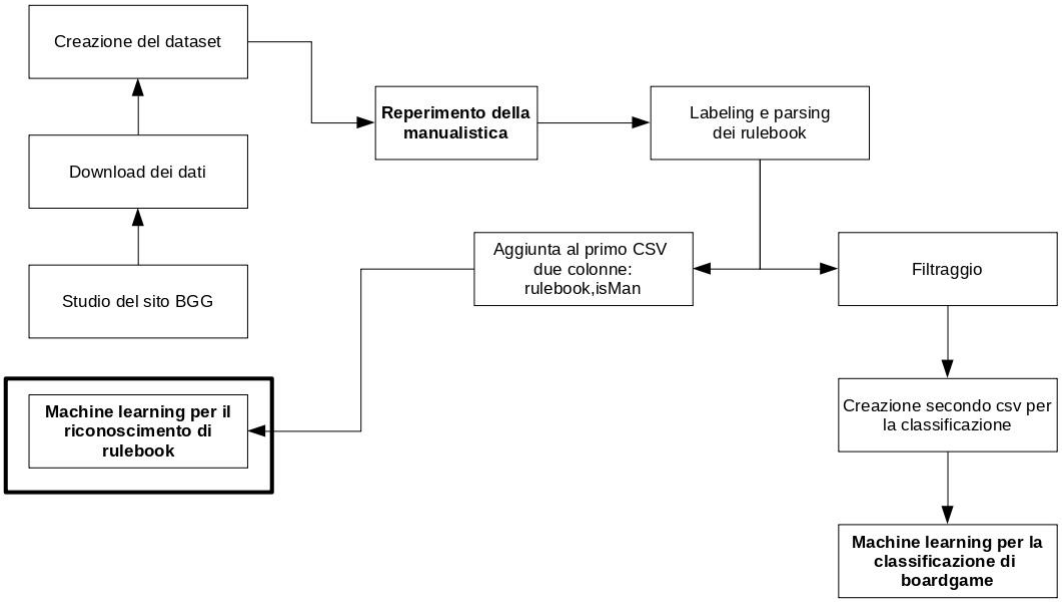


Il grafico in figura 6.10 mostra come sia il filtraggio ottenuto manualmente che quello automatico, abbiano prodotto i medesimi risultati. Il 97 % delle righe del csv filtrato a mano è presente anche nelle righe del dataset filtrato tramite algoritmo.

6.4 Machine learning classificazione dei manuali

La sezione comincia col riportare la fase a cui si è finalmente giunti, evidenziata nell'immagine 6.11.

Figura 6.11: Fase machine learning per il riconoscimento



In aggiunta viene riportato l'iter seguito che va a mostrare il machine learning usato per classificare i vari rulebook dei boardgame, la cui label per categorizzare è proprio la variabile *isMan*. Si riporta nell'immagine 6.12 l'ordine degli step per effettuare machine learning sui manuali.

La costruzione dei modelli e le predizioni, il cuore del lavoro, avviene all'interno della CV. A livello teorico è stata già ampiamente illustrata nel capitolo 3, la cross validazione è utilizzata per avere performance più realistiche evitando casi di overfitting. In figura 6.13 si riportano n split, in cui è possibile vedere come dentro ognuno di essi ci sia una parte lasciata fuori quella di train, usata per allenare il modello, quella di test, usata per predire i campioni. Banalmente i campioni di test non devono fare parte del train, poiché andrebbero a invalidare le prestazioni.

Nell'immagine 6.14 si riporta nello specifico l'interno della Cross Validazione. Python fornisce apposite librerie che in maniera efficace implementano meccanismi di split dei dati in train e test, e che possono in maniera automatica creare modelli sui dati di allenamento, per poi testarli sui dati di testing appunto. Il numero di split viene deciso a priori. Negli esperimenti cui risultati sono presenti nella sezione

Figura 6.12: Machine learning

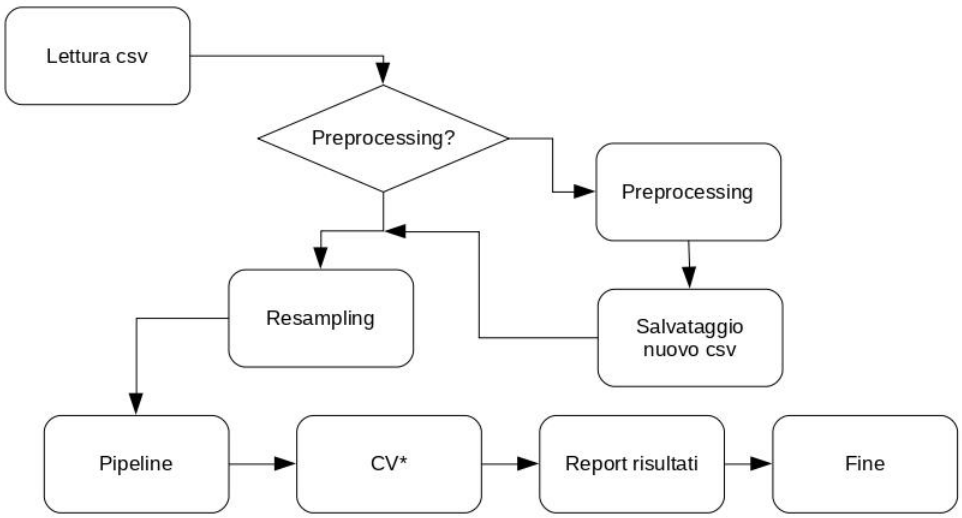


Figura 6.13: Esempio di split in train-test

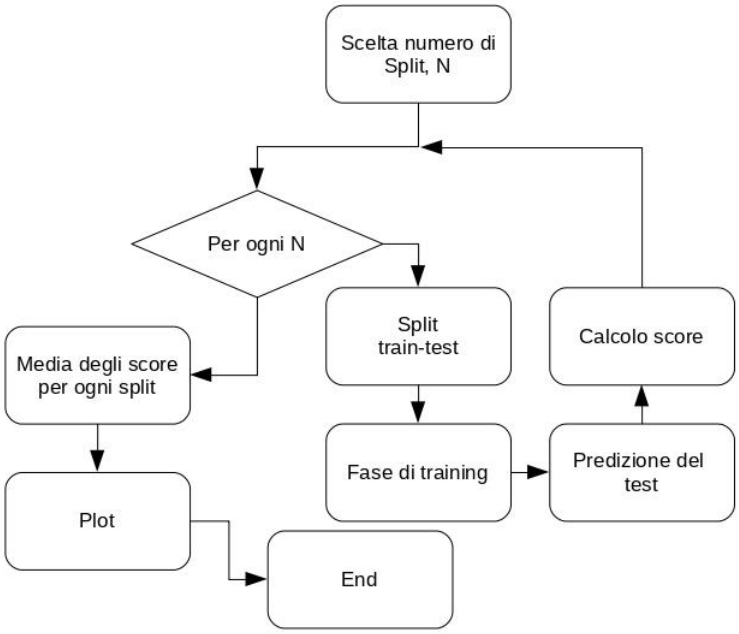
6.4.2, sono stati usati 10 split. Per ogni split è stata calcolata l'accuracy, cui poi è stata fatta la media. Ciò è stato svolto per ognuno dei 4 algoritmi riportati sempre nella rispettiva sezione.

6.4.1 Preprocessing, pipeline e resampling

La fase di preprocessing è sempre obbligata. Trattare i dati senza un minimo averli puliti prima, porta solo a pessime performance da parte del machine learning, con risultati di conseguenza non affidabili. Una volta effettuato il preprocessing, per comodità, si salva una copia del file già pretrattato. Questo perché ad una successiva esecuzione, è possibile caricare il dataset già preprocessato senza dover rifare nuovamente questa fase.

Il preprocessing del testo consiste appunto nell'avvalersi della text analytics, che viene in aiuto al machine learning. Viene così elaborata una mole enorme di testo cercando di andare ad estrarre l'informazione testuale utile. Si ricorda

Figura 6.14: Cross validazione



che la text analytics possiede una varietà di tecniche: dalla sentiment analysis, al calcolo del term frequency, al topic modeling, al ricooscimento di entità nominate. Il preprocessing effettuato ha portato a svolgere lo stemming, la tokenization, rimozione delle stopword, rimozione di caratteri speciali e degli spazi, conversione del testo in minuscolo.

La pipeline, intesa allo stesso modo nell'ambiente Unix, è stata ampiamente utilizzata in python. Sklearn offre la possibiltà di costruire delle pipeline necessarie per effettuare il machine learning in maniera comoda e semplice. L'output di uno step è l'input di quello dopo. La pipeline usata per ogni esperimento è strutturata in 3 step, che vengono spiegati: *CountVectorizer(),TfidTransformer(),algoritmo.*

- *CountVectorizer* è un metodo che permette di codificare l'informazione testuale in un formato con cui gli algoritmi di machine learning possono lavorare. Esso genera una matrice grazie a un insieme di file testuali, in questo elaborato si è usata la descrizione dei boardgame, il rulebook, entrambi. La matrice possiede tante colonne quanti sono i token complessivi presenti all'interno dell'insieme dei documenti testuali forniti (salvo un limite posto manualmente). Ogni cella di questa matrice possiede il numero di volte cui un determinato termine appare all'interno del testo associato alla riga. Questa quantità si chiama *Term Frequency*

- *TfidTrasformer* trasforma le quantità appena citate in una misura chiamata TF-IDF. Essa si ottiene moltiplicando la TF per una quantità chiamata IDF, Inverted Document Frequency:

$$IDF(t) = log\left(\frac{n}{df(t)}\right) \tag{6.1}$$

  questa misura è più alta tanto quanto t risulta raro dentro un set di documenti. Ciò implica che se un termine è raro in un set, vuol dire che aiuta a distinguerlo meglio. Documenti diversi che possiedono al loro interno un termine particolare nella collezione, è altamente probabile che condividano la stessa classe poiché i termini rari sono spesso specifici di alcune classi. TD-IDF assegna un punteggio alto a un termine che compare spesso dentro i documenti. Contemporaneamente però se tale termine è presente in tanti documenti, IDF, un numero tra 0 e 1, riduce lo score perché moltiplicato per la TF. Termine raro in un set, IDF alto.

- *algoritmo* di machine learning, citati nel capitolo 3, sono l'ultimo step della pipe, che permettono di effettuare appunto classificazioni e predizioni grazie agli input che sono arrivati a questo punto.

Il resampling, o ricampionamento, è una strategia che consente di sistemare il dataset a livello di numerosità campionaria di ogni classe. Ognuna di esse dopo il resampling avrà lo stesso numero di campioni di ogni altra classe. Questo permette di avere risultati più affidabili, perché le classi essendo bilanciate a livello di campioni, le predizioni non andranno a pendere a favore di alcuna classe. Si tende a scegliere una soglia, che comunque è il minimo rispetto alla classe che ha meno campioni delle altre. Se non si dovesse fare così, la classe che non arriva alla soglia, avrà una fase di ricampionamento casuale, dove alcuni campioni vengono duplicati fino a che non si arriva alla soglia. Questo può esser problematico poiché avere duplicati può influire negativamente sulle prestazioni. Negli esperimenti riportati, avendo due classi, la meno numerosa possedeva circa 1.4k campioni, è stato fatto un resampling su questa quantità, per entrambe le classi. In conclusione si hanno, per i risultati proposti di seguito, 2.8k campioni divisi in due classi *yes_rulebook,not_rulebook*.

6.4.2 Risultati

Sono doverose ovviamente alcune osservazioni circa i risultati ottenuti: si parte dal presupposto sì di aver fatto resampling per avere un numero di campioni bilanciato in ambedue le classi. Tuttavia per effettuare di base del machine learning, la numerosità dei campioni è di vitale importanza per avere delle prestazioni e delle performance

Figura 6.15: Media dell'accuracy per ogni algoritmo

algoritmo	Accuracy
MNB	0.80
RF	0.94
SVC linear	0.92
SGD	0.91

affidabili. I classificatori prodotto in questo esperimento sono puramente binari. Ogni classificatore ha il compito di differenziare i documenti per capire se il campo testuale fosse un rulebook oppure no.

Figura 6.16: Accuracy in fase di CV

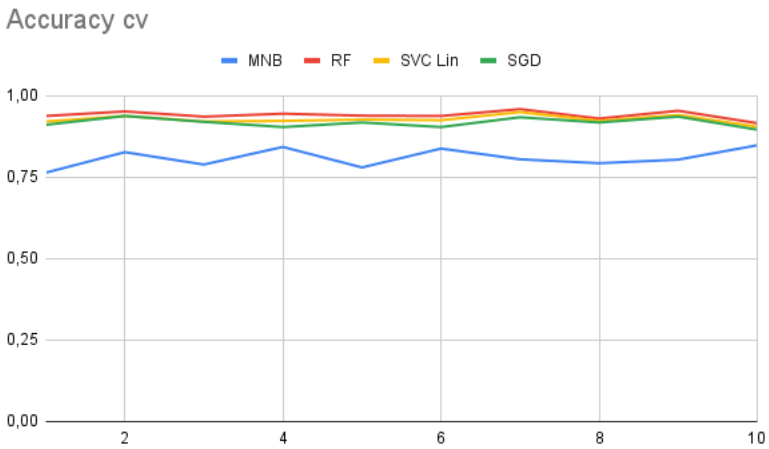
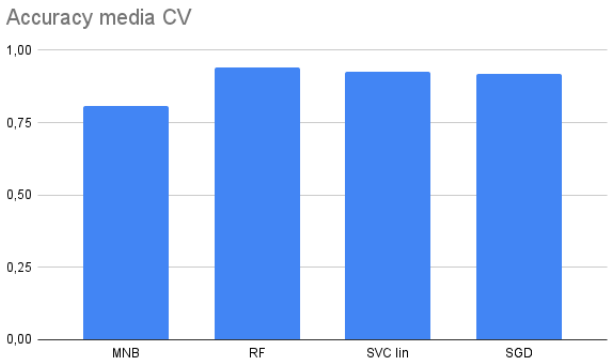


Figura 6.17: Barplot accuracy media degli algoritmi

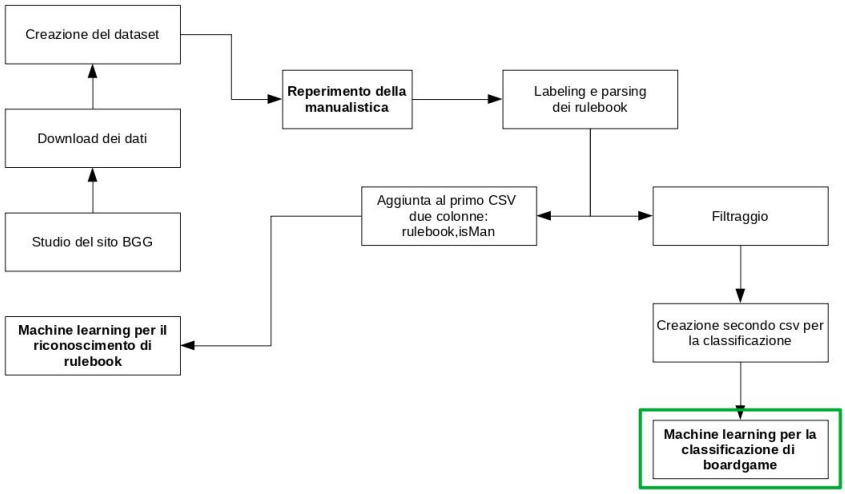


Capitolo 7

ML per la classificazione dei boardgame

Nel precedente capitolo 6 si è riportato il complesso di operazioni che hanno portato a etichettare i rulebook con successivo parsing. Dopodiché si è inserito nel dataset di studi sia la label che identifica se il file è o meno un manuale, sia il contenuto di ogni manuale per il gioco corrispondente, e si è svolto del machine learning per riconoscere ciò che è un rulebook da ciò che non lo è. Si è poi effettuato un filtraggio che ha portato a escludere manuali effettivi, ma che non si potevano utilizzare perché l'informazione una volta decodificata, è risultata illeggibile e inusabile per fare machine learning.

Figura 7.1: Accuracy in fase di CV

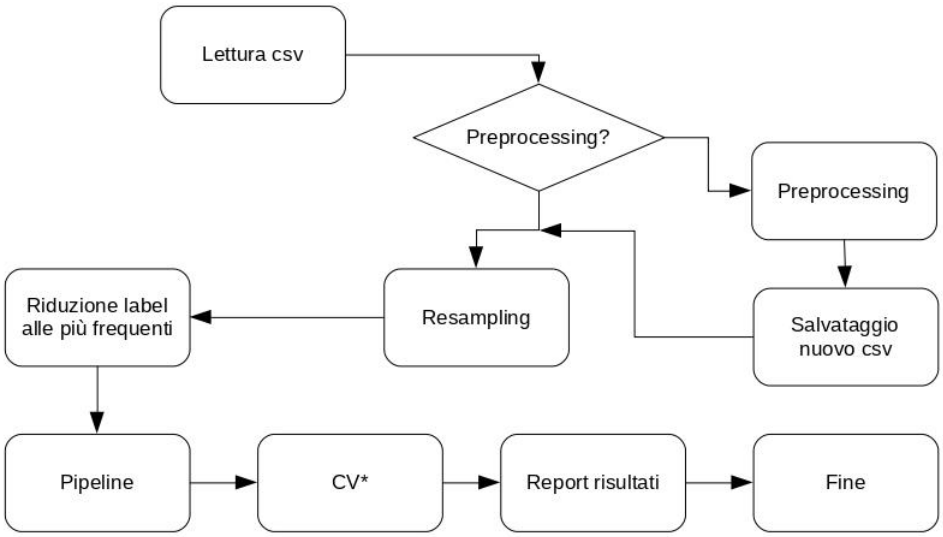


Nel capitolo attuale, come viene mostrato in figura 7.1, si propone una serie di

modelli di machine learning che vogliono classificare i boardgame per meccaniche/-categorie tramite l'uso di un campo testuale, che nel corso del capitolo potrà essere la descrizione del boardgame, il rulebook o entrambe unire come campo testuale unico. Anche qui viene effettuata la CV, presente in figura 6.14 e non riportata nuovamente in questo capitolo perché l'ordine degli step è pressoché identico. Viene invece riportata in figura 7.2 il flusso di operazioni seguito per il machine learning e classificare i giochi per meccanica/categoria.

Nonostante sia equivalente all'immagine 6.12 del precedente capitolo, è stata aggiunta una fase importante che ha portato alla semplificazione del problema, ovvero la *riduzione delle label*. Questa fase ha portato ogni boardgame a possedere una sola meccanica e una sola categoria al fine di essere un problema più trattabile, essendo comunque un progetto incentrato su un campo totalmente inesplorato. La label mantenuta per ogni boardgame, sia meccanica che categoria è quella più frequente nel dataset. Effettuare questa operazione porta a ridurre banalmente il numero di meccaniche e categorie, non eccessivamente, tuttavia alcune scompaiono perché poco frequenti, e con questa operazione svaniscono del tutto.

Figura 7.2: Machine learning



Una volta effettuato il *preprocessing*, che comporta rimozione delle stopwords, stemming e lemmatizing, viene fatta una copia del dataset con i campi testuali preprocessati, di modo da non dover ripetere lo stesso step per le volte successive e caricare così il dataset già pretrattato.

Anche qui per questi esperimenti è stato effettuato il *resampling*. La differenza col capitolo precedente è data dal fatto che qui sono stati scelti 250 campioni per

ogni meccanica (300 per le categorie), e sono state mantenute le prime 5 meccaniche (categorie) più frequenti. Una osservazione che viene immediatamente fatta è la seguente: per svolgere una analisi di questo tipo, dopo il resampling sicuramente si hanno pochi campioni. Per avere risultati più significativi è necessario avere molti più dati. Di certo è un buon punto di partenza su cui effettuare delle ricerche.

Successivamente viene scelta la *pipeline*, già riportata nella sezione 6.4.1, si esegue la CV, anch'essa riportata nella sezione 6.4.1, vengono riportati e discussi i risultati ottenuti. Gli algoritmi, così come alcune funzioni della pipeline in python, possiedono degli iperparametri. Non sono stati scelti casualmente, bensì sempre grazie al lavoro qui proposto di cui si sono seguite le orme [2]. In tale lavoro è stata utilizzata la gridsearch, un metodo fornito da sklearn e che permette di ottenere la migliore combinazione di iperparametri ideali andando a creare moltissimi modelli facendo tutte le combinazioni possibili per trovare quella migliore. Tuttavia nel lavoro appena citato non sono presenti alcuni algoritmi che invece in questo progetto di tesi sono stati utilizzati, che usano anch'essi degli iperparametri. Quindi per gli algoritmi cui è possibile, sono stati usati gli stessi parametri di questo studio, per quelli non usati, si è optato per lasciare i valori di default.

Figura 7.3: Iperparametri usati per i problemi di classificazione per meccaniche

Boardgamemechanic, CV used = ShuffleSplit, 10 fold	
Algorithm	Hyperparams (Algorithm & CountVectorize)
MNB	max_features=2500,idf=True
SVC	kernel=rbf,C=1,max_features=2500,idf=True
RF	n_est=1000,max_features=4000,idf=True
LinearSVC	kernel=linear,C=1,max_features=2500,idf=True
SGD	loss=hinge,penalty=elasticnet,max_iter=10^6

Figura 7.4: Iperparametri usati per i problemi di classificazione per categorie

Boardgamecategory, CV used = ShuffleSplit, 10 fold	
Algorithm	Hyperparams (Algorithm & CountVectorize)
MNB	max_features=30000,idf=True
SVC	kernel=rbf,C=2,max_features=30000,idf=True
RF	n_est=1000,max_features=20000,idf=False
LinearSVC	kernel=linear,C=2,max_features=30000,idf=True
SGD	loss=hinge,penalty=elasticnet,max_iter=10^6

7.1 Classificazione basata su meccaniche

I primi esperimenti proposti si basano sulla classificazione dei boardgame per meccaniche. Questa sezione riporta i risultati ottenuti in questo contesto, partendo dall'utilizzo della sola descrizione, per poi arrivare al solo uso dei rulebook, e concludere con l'utilizzo di entrambi i campi testuali. La tabella in figura 7.3 riporta gli iperparametri utilizzati per questa classificazione. I risultati qui proposti, si ricorda ulteriormente, che sono stati ottenuti dal dataset le cui label di boardgame sono state sostituite con la meccanica più frequente. Ci si ritrova in una situazione che è la seguente, in 7.5. Il dataset, molto sbilanciato, è per questo motivo che è stato oggetto di resamping sulle 5 meccaniche più significative, usando 250 campioni per ognuna di esse, come mostra il grafico in figura 7.6.

Figura 7.5: Dataset no res, col mantenimento delle meccaniche più frequenti

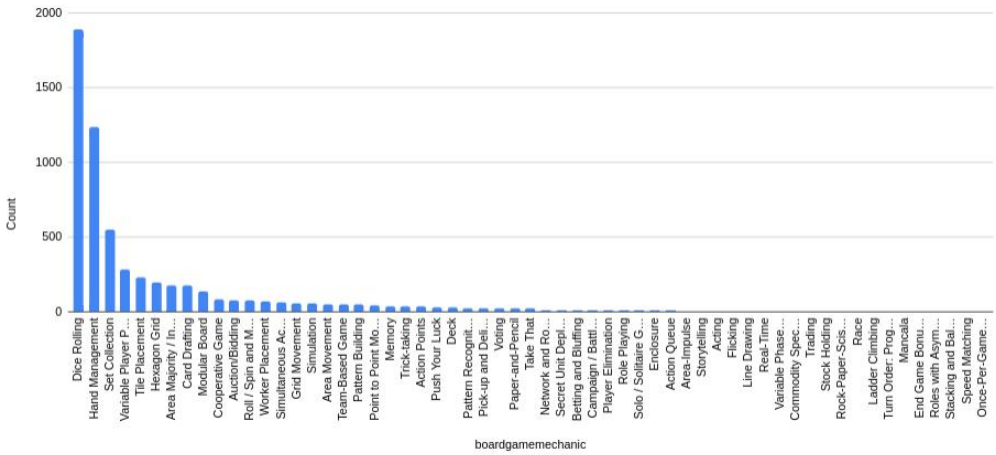
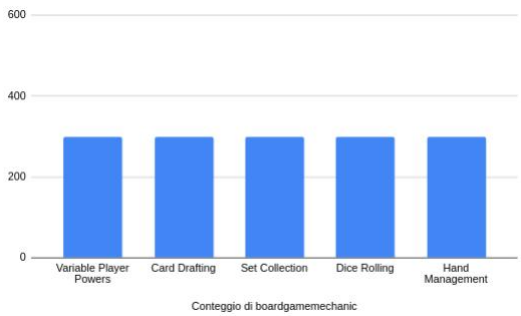


Figura 7.6: Dataset ricampionato



7.1.1 Uso del rulebook

Figura 7.7: Risultati classificazione meccaniche tramite rulebook

	accuracy	precision	recall	f1
MNB	0,57	0,58	0,57	0,57
RF	0,76	0,78	0,77	0,76
LinearSVC	0,68	0,68	0,68	0,67
LinearSVC ovo	0,68	0,69	0,69	0,68
SGD	0,67	0,66	0,66	0,66
SVC	0,68	0,69	0,68	0,68
SVC ovo	0,68	0,69	0,68	0,68

Figura 7.8: Andamento CV MNB

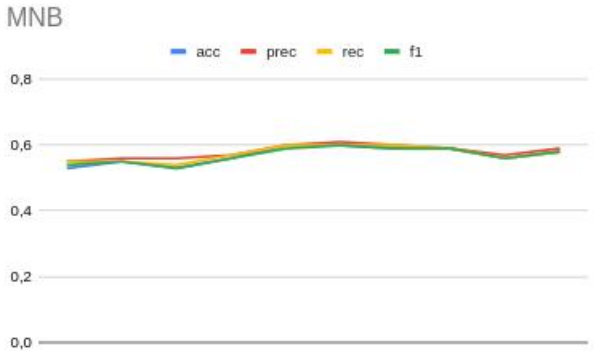


Figura 7.9: Andamento CV RF

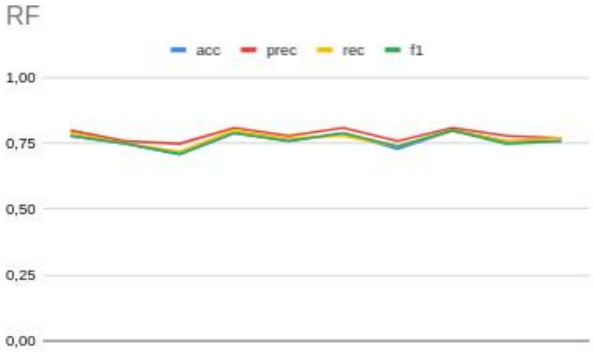


Figura 7.10: Andamento CV linear SVC

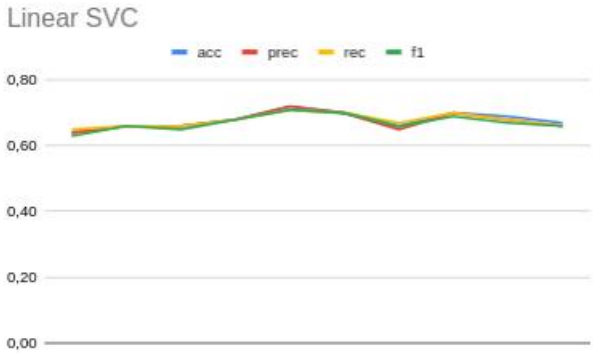


Figura 7.11: Andamento CV linear SVC ovo

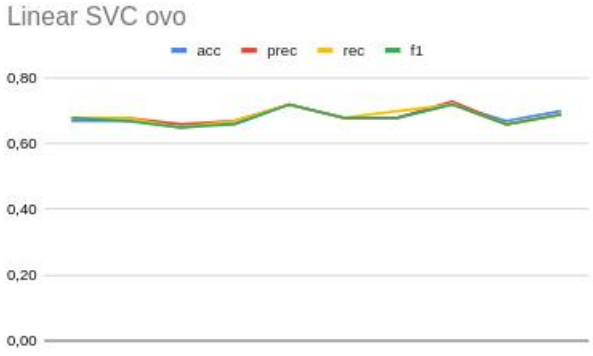


Figura 7.12: Andamento CV SGD

Figura 7.13: Andamento CV SVC

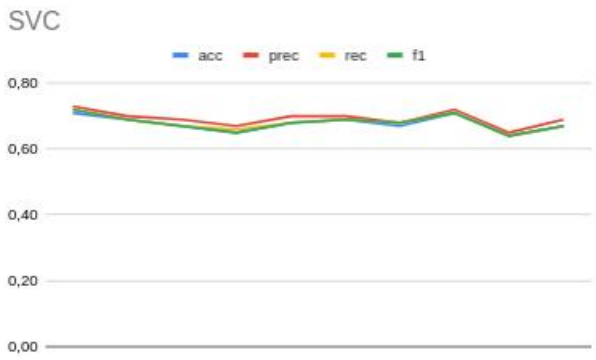
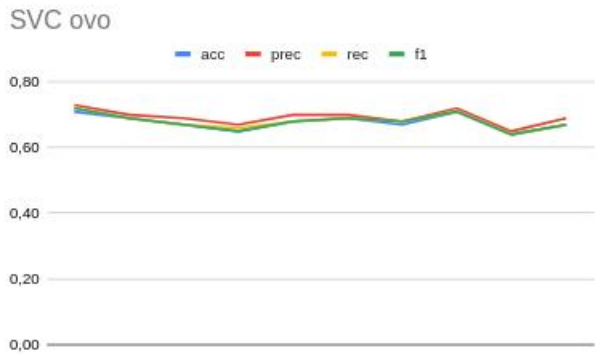


Figura 7.14: Andamento CV SVC ovo



7.1.2 Uso della description

Figura 7.15: Risultati classificazione meccaniche tramite descrizione

	accuracy	precision	recall	f1
MNB	0,58	0,60	0,58	0,58
RF	0,70	0,72	0,70	0,70
LinearSVC	0,64	0,65	0,65	0,64
LinearSVC ovo	0,66	0,67	0,67	0,66
SGD	0,64	0,65	0,65	0,64
SVC	0,68	0,69	0,68	0,68
SVC ovo	0,68	0,69	0,68	0,68

Figura 7.16: Andamento CV MNB

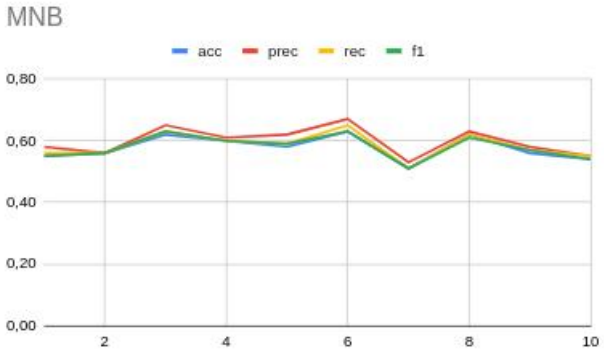


Figura 7.17: Andamento CV RF

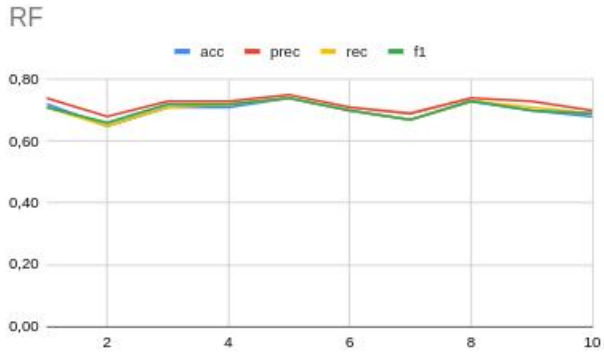


Figura 7.18: Andamento CV linear SVC

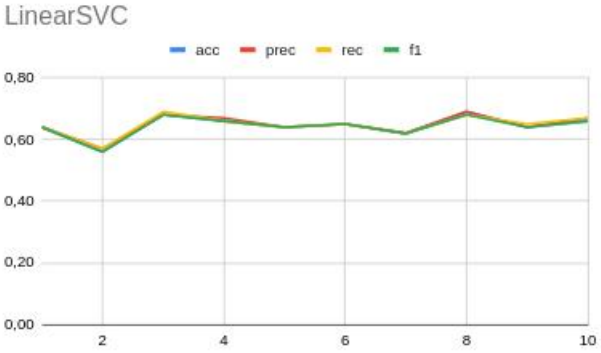


Figura 7.19: Andamento CV linear SVC ovo

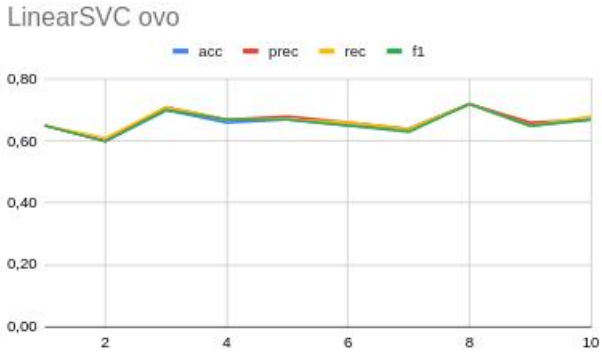


Figura 7.20: Andamento CV SGD

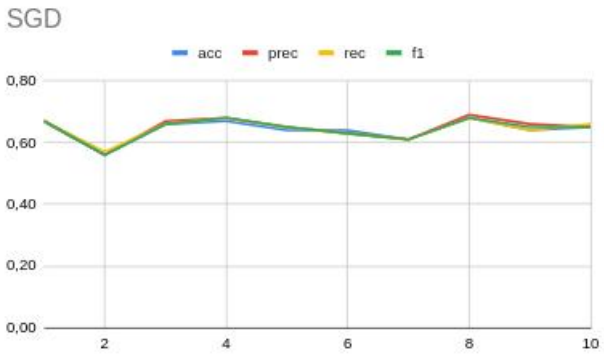


Figura 7.21: Andamento CV SVC

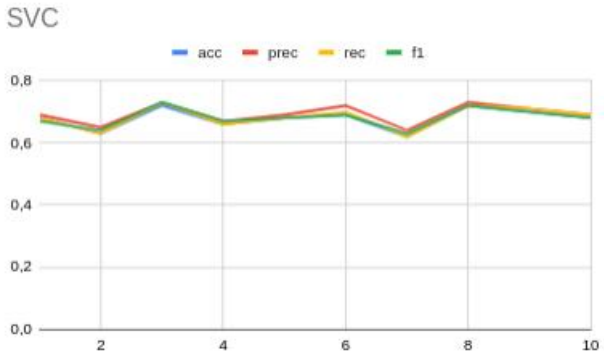
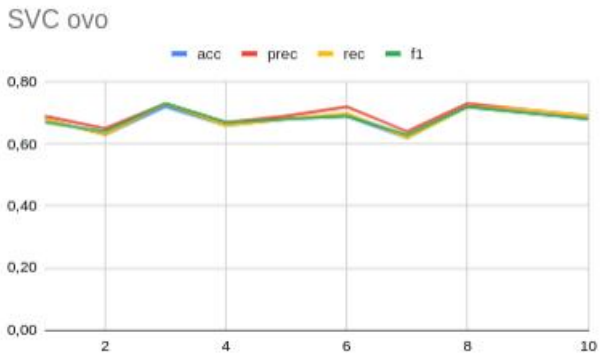


Figura 7.22: Andamento CV SVC ovo



7.1.3 Uso di entrambi

Figura 7.23: Risultati classificazione meccaniche tramite l'uso di entrambi

	accuracy	precision	recall	f1
MNB	0,55	0,57	0,55	0,55
RF	0,76	0,77	0,75	0,75
LinearSVC	0,66	0,66	0,66	0,65
LinearSVC ovo	0,67	0,68	0,68	0,67
SGD	0,65	0,66	0,65	0,65
SVC	0,68	0,70	0,69	0,69
SVC ovo	0,69	0,70	0,69	0,69

Figura 7.24: Andamento CV MNB

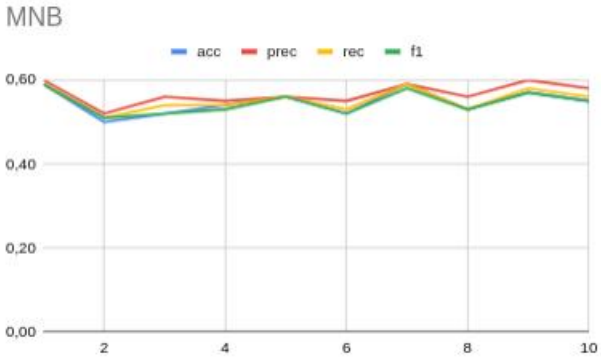


Figura 7.25: Andamento CV RF

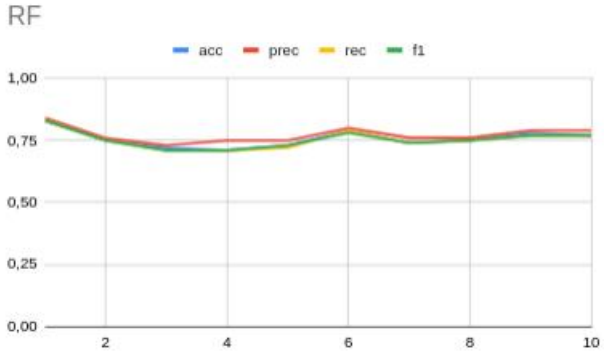


Figura 7.26: Andamento CV linear SVC

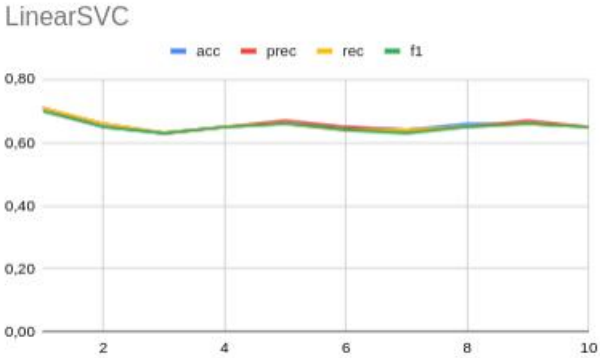


Figura 7.27: Andamento CV linear SVC ovo

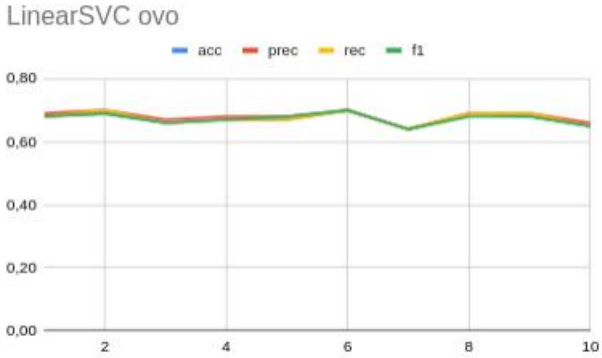


Figura 7.28: Andamento CV SGD

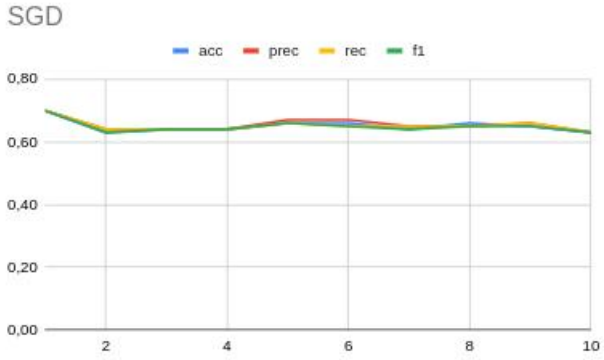


Figura 7.29: Andamento CV SVC

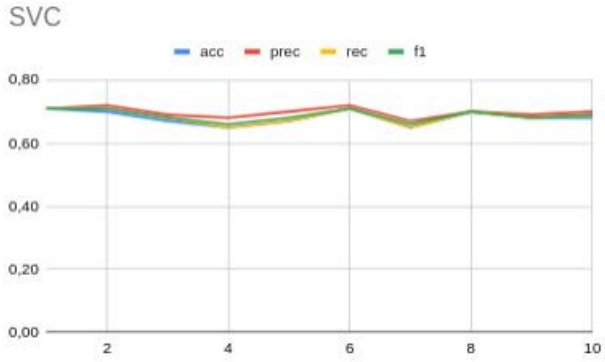
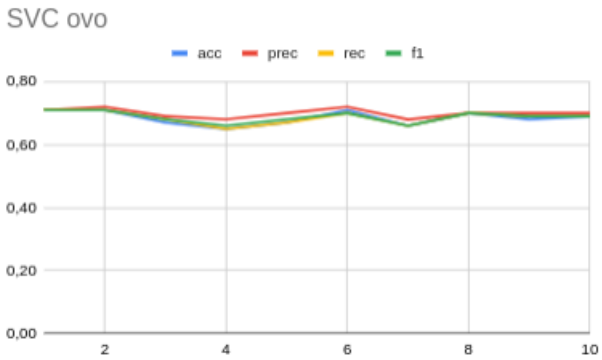
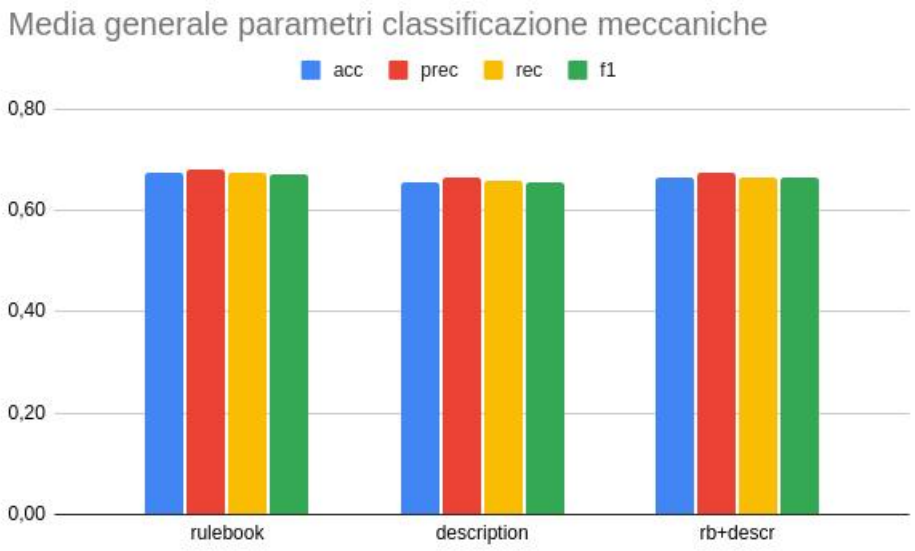


Figura 7.30: Andamento CV SVC ovo



7.1.4 Osservazioni sui risultati

Figura 7.31: Media generale per la classificazione delle meccaniche



I grafici riportati nelle sezioni precedenti vogliono illustrare i parametri, accuracy precision recall ed F_1, evolvono durante la cross validazione. Non si sono riportate tabelle che rappresentavano tali numeri, invece si è preferito per ogni algoritmo e per i vari campi testuali, riportare un plot che illustrasse tale andamento. Nelle rispettive tabelle si sono riportate le medie di ogni grandezza ottenuta in ogni split, per avere una visione decisamente più compatta della situazione. Si evince come l'algoritmo più efficiente sia il random forest. Il motivo è dato anche dalla moltitudine di alberi dentro la foresta, scorrelati fra loro, per aggiungere diversità ottenendo delle predizioni più accurate.

Si riportano due grafici riassuntivi della situazione attuale. Il primo, figura 7.31 viene riportata per ogni campo testuale usato per classificare, una *media delle medie di tutti gli algoritmi.* Nel contesto della classificazione per meccaniche *l'uso della sola descrizione rispetto al solo manuale degrada leggermente le prestazioni. L'uso di entrambi non porta alcun miglioramento significativo.*

Il secondo, In figura 7.35 si ha un plot che mostra due aspetti importanti:

- le prestazioni confrontate in termini di grandezze di machine learning fra tutti gli algoritmi utilzzati. Emerge come il Random Forest sia il più efficiente sotto tutti i punti di vista per questa classificazione

- per ogni algoritmo sono riportate le performance utilizzando il solo rulebook, la sola descrizione ed entrambi. Ogni algoritmo sull'asse delle X infatti ha il proprio nome riportato 3 volte per rispecchiare queste 3 situazioni appena ricordate. Qui si può notare come la situazione generale tenda ad essere quella che è stata appena citata. *L'uso del rulebook tende ad avere le stesse prestazioni, o leggermente superiori rispetto al solo uso della descrizione del boardgame, mentre l'uso di entrambi i campi testuali non porta miglioramenti a vista d'occhio.*

7.1.5 Focus sul Random Forest, analisi most important feature

Viene in questo paragrafo effettuato un breve focus sui risultati ottenuti all'interno dell'algoritmo Random Forest. Anche in questo caso i risultati proposti sono da intendere per il dataset ricampionato, con 250 campioni mantenendo le prime 5 meccaniche più frequenti (*Variable Player Powers, Card Drafting, Set Collection, Dice Rolling, Hand Management*). Per i metodi ensemble, quelli basati sulla costruzione di alberi come il RF, ma non solo, esistono degli indici di grandezza che aiutano a capire l'importanza di alcune feature, testuali in tal caso. Usando entrambi i campi testuali per classificare i boardgame per meccaniche si ottiene un leggero degrado delle prestazioni.

Nell'immagine 7.32 si può notare come quelle più in rilievo siano *roll (la più influente),dice,die,card,roll dice,play* .... Da queste si evince come siano dei verbi la maggioranza, volti appunto a spiegare il funzionamento del gioco, in quanto presenti nei manuali. In figura 7.33 si nota invece come le più influenti siano *dice (la più influente), cards, card, card game.* Tutti vocaboli presenti nelle descrizioni di un boardgame. L'ultima nota è data dall'immagine 7.34. Saltano all'occhio le due più pesanti, che sono rispettivamente *roll, la più influente per il solo uso del manuale, dice, analogamente ma usando solo la description*; in quest'ultimo contesto entrambe le feature vanno a bilanciarsi, ciò può essere anche dato dal fatto che usando entrambi i campi testuali, non si ottengano miglioramenti significativi ma addirittura un lieve degrado di perfomance. *Dice, dice roll* sono presenti anche nel contesto del solo uso del rulebook, ma risultano decisamente poco influenti. *Ottenere dei risultati migliori usando il rulebook, classificando per meccaniche, obiettivamente può risultare più sensato in quanto sono presenti vocaboli decisamente molto specifici che indicano i meccanismi di un determinato boardgame.*

Figura 7.32: Top 20 feature per importanza usando il manuale

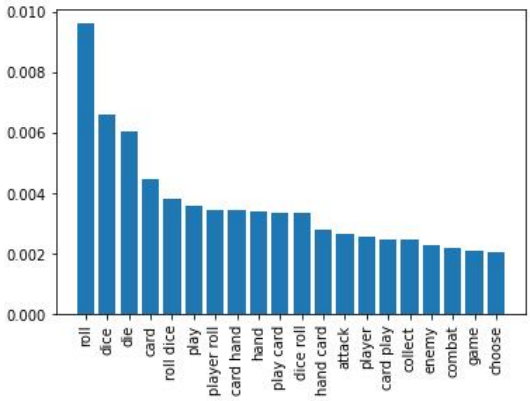


Figura 7.33: Top 20 feature per importanza usando la descrizione

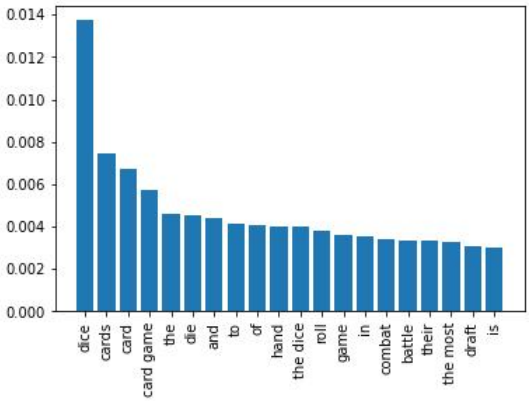
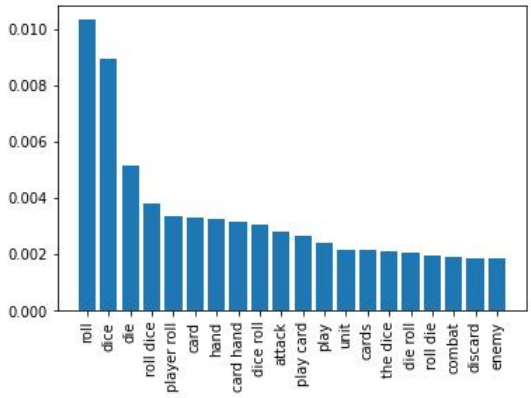


Figura 7.34: Top 20 feature per importanza usando entrambi



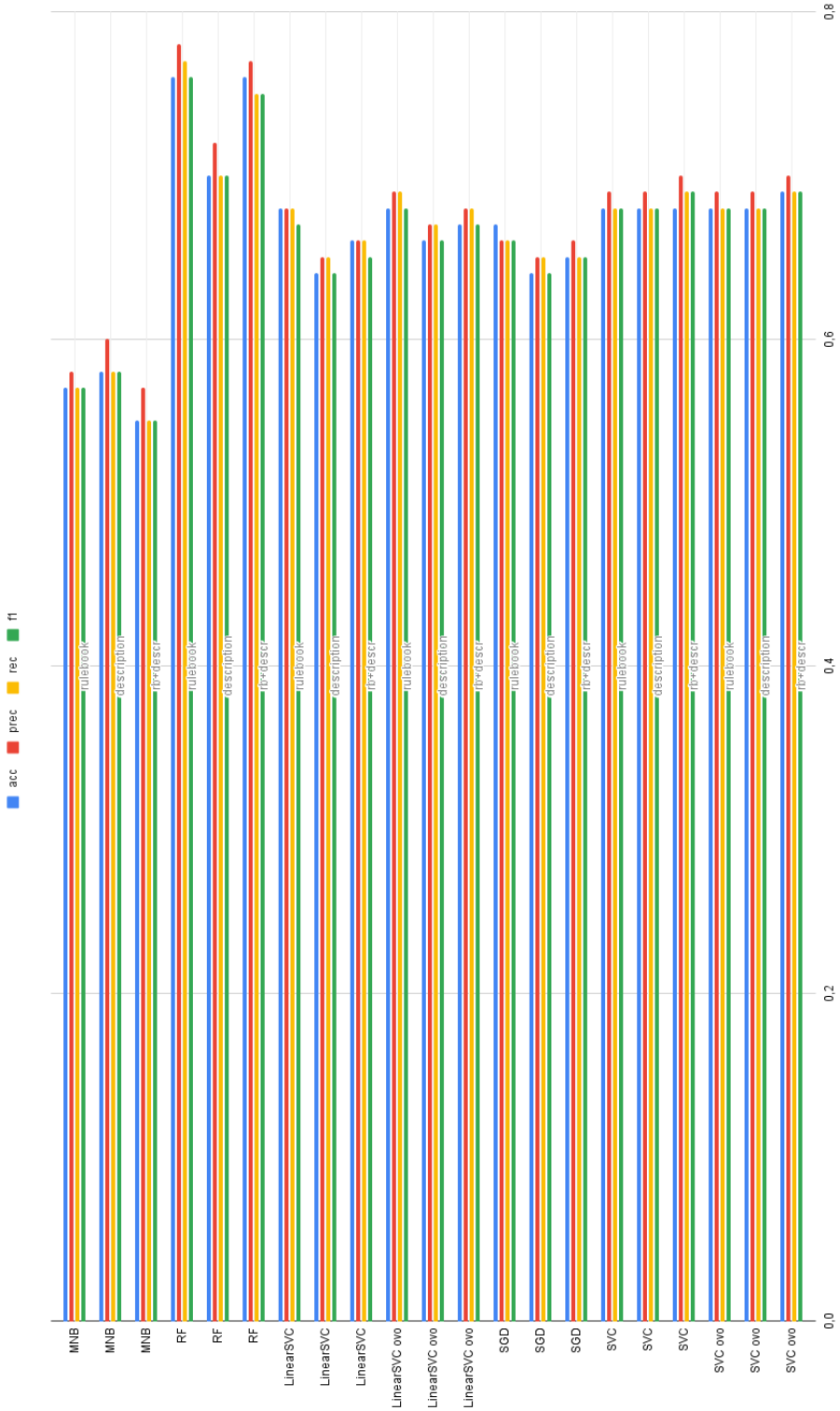


Figura 7.35: Check performance per i vari campi testuali nella classificazione per meccaniche

7.2 Classificazione basata su categorie

Gli esperimenti continuano, proponendo in questa sezione la classificazione dei boardgame per categorie. Questa sezione riporta i risultati ottenuti in questo contesto, partendo dall'utilizzo della sola descrizione, per poi arrivare al solo uso dei rulebook, e concludere con l'utilizzo di entrambi i campi testuali. La tabella in figura 7.4 riporta gli iperparametri utilizzati per questa classificazione. I risultati qui proposti, facendo un discorso analogo alle meccaniche, sono stati ottenuti dal dataset le cui label di boardgame sono state sostituite con la categoria più frequente. Ci si ritrova in una situazione che è la seguente, in 7.36. Il dataset, molto sbilanciato, è per questo motivo che è stato oggetto di resamping sulle 5 categorie più significative, usando 300 campioni per ognuna di esse, come si evince in figura 7.37.

Figura 7.36: Dataset no res, mantenimento delle categorie più frequenti

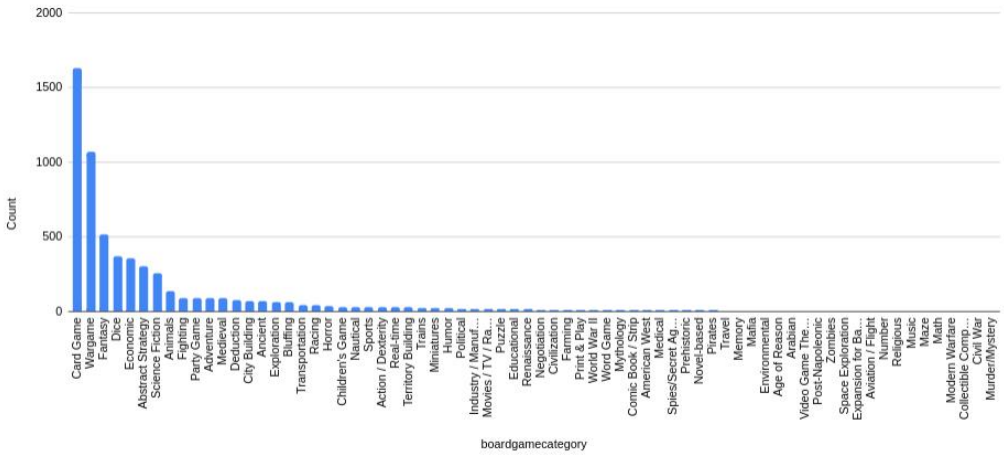
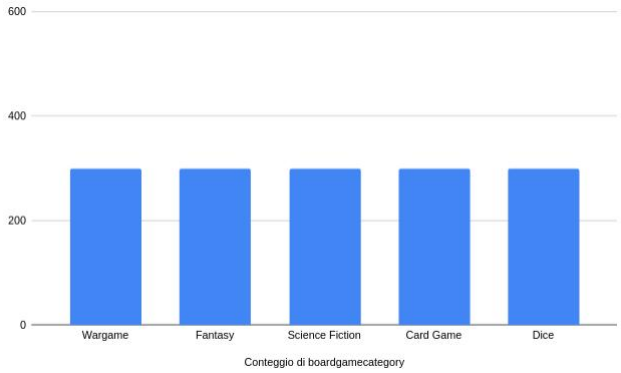


Figura 7.37: Dataset ricampionato



7.2.1 Uso del rulebook

Figura 7.38: Risultati classificazione categorie tramite rulebook

	accuracy	precision	recall	f1
MNB	0,78	0,78	0,78	0,78
RF	0,82	0,83	0,82	0,82
LinearSVC	0,81	0,81	0,81	0,81
LinearSVC ovo	0,81	0,82	0,81	0,81
SGD	0,80	0,80	0,80	0,80
SVC	0,81	0,81	0,81	0,81
SVC ovo	0,81	0,81	0,81	0,81

Figura 7.39: Andamento CV MNB

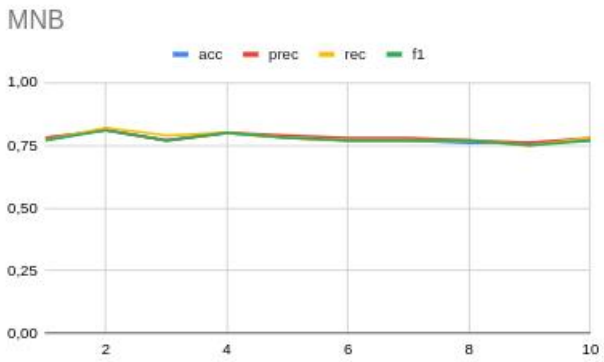


Figura 7.40: Andamento CV RF

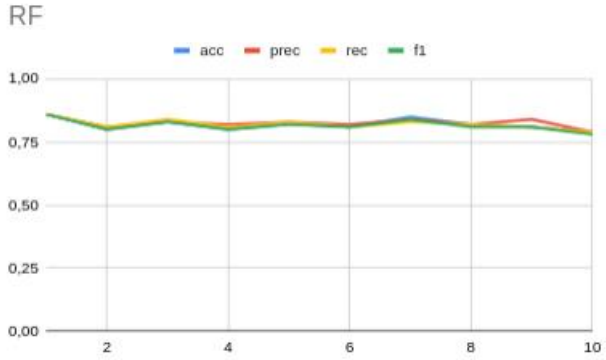


Figura 7.41: Andamento CV linear SVC

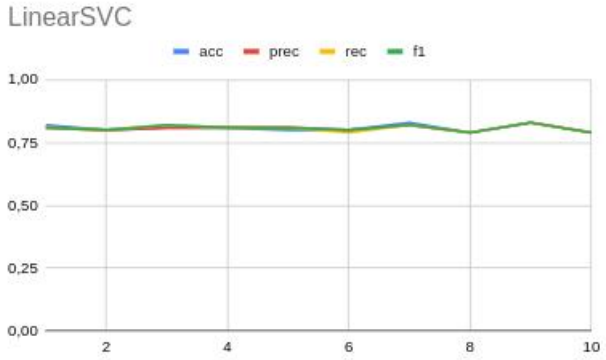


Figura 7.42: Andamento CV linear SVC ovo

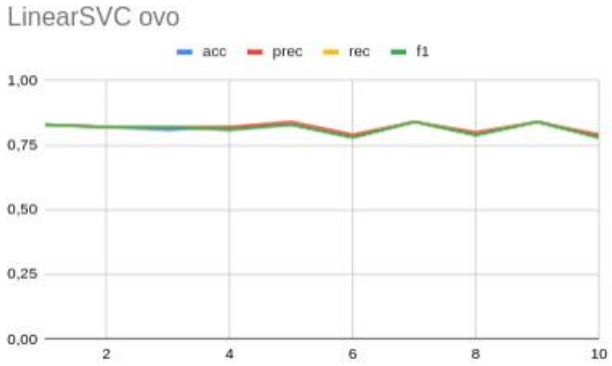


Figura 7.43: Andamento CV SGD

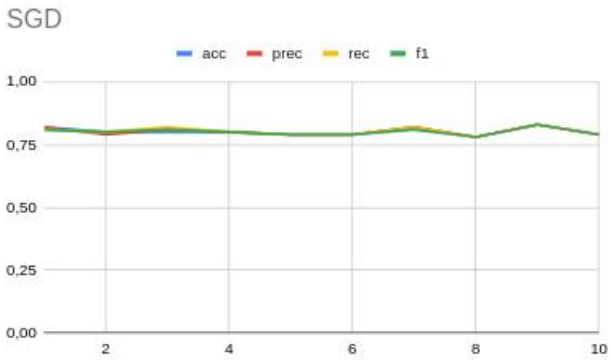


Figura 7.44: Andamento CV SVC

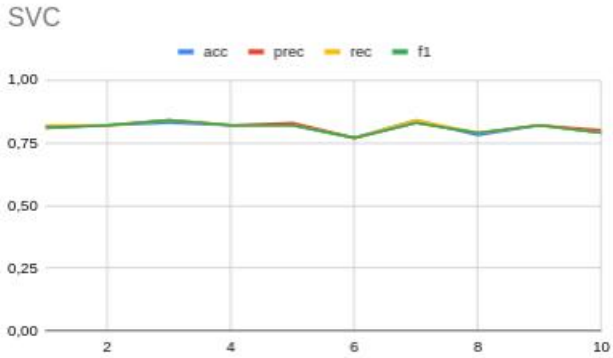
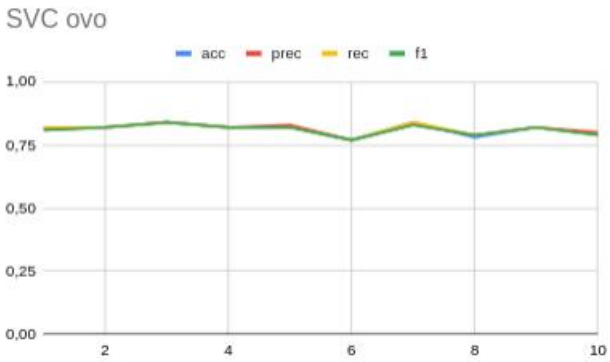


Figura 7.45: Andamento CV SVC ovo



7.2.2 Uso della description

Figura 7.46: Risultati classificazione categorie tramite descrizione

	accuracy	precision	recall	f1
MNB	0,79	0,81	0,79	0,79
RF	0,81	0,81	0,81	0,81
LinearSVC	0,82	0,83	0,83	0,82
LinearSVC ovo	0,82	0,82	0,82	0,82
SGD	0,82	0,83	0,82	0,82
SVC	0,81	0,82	0,81	0,81
SVC ovo	0,81	0,82	0,81	0,81

Figura 7.47: Andamento CV MNB

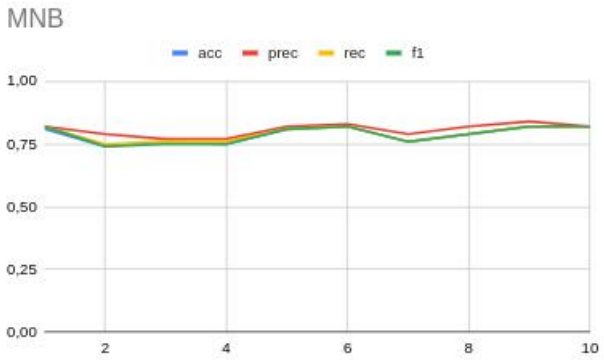


Figura 7.48: Andamento CV RF

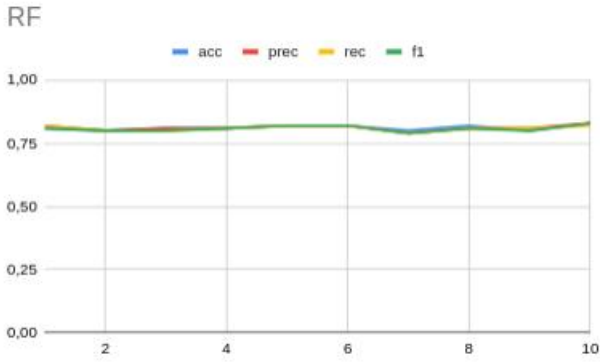


Figura 7.49: Andamento CV linear SVC

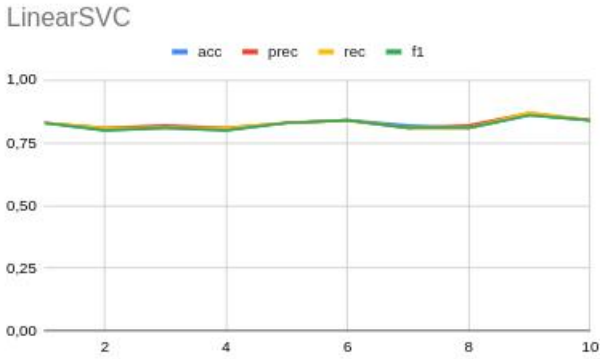


Figura 7.50: Andamento CV linear SVC ovo

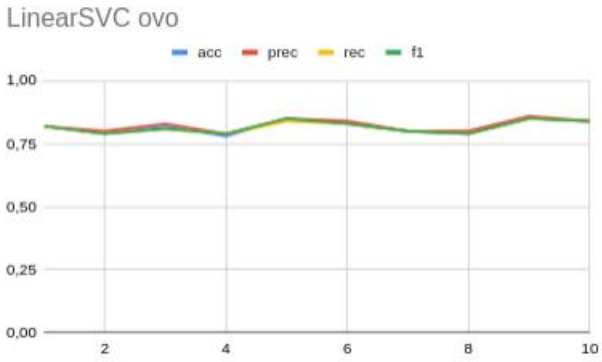


Figura 7.51: Andamento CV SGD

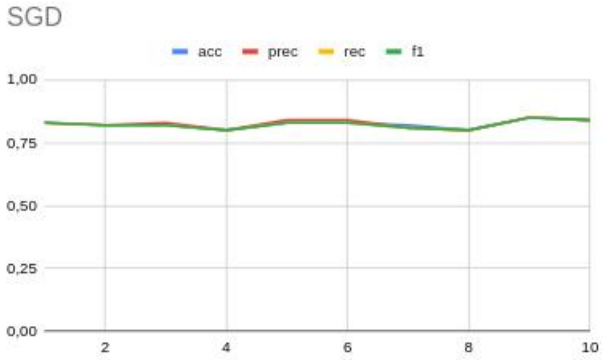


Figura 7.52: Andamento CV SVC

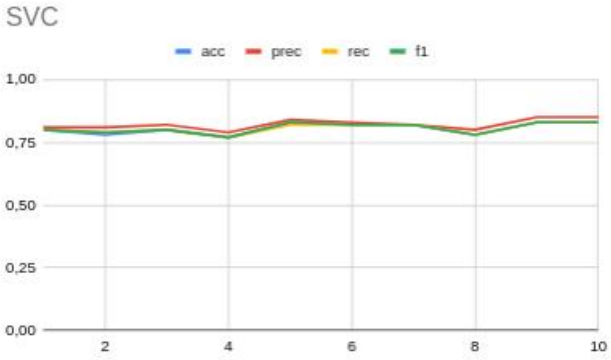
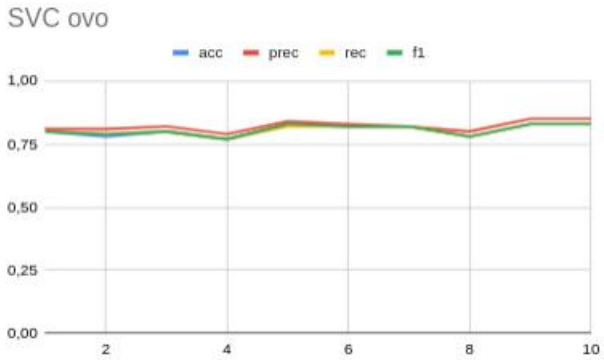


Figura 7.53: Andamento CV SVC ovo



7.2.3 Uso di entrambi

Figura 7.54: Risultati classificazione categorie tramite l'uso di entrambi

	accuracy	precision	recall	f1
MNB	0,79	0,80	0,80	0,79
RF	0,84	0,84	0,84	0,83
LinearSVC	0,81	0,81	0,82	0,81
LinearSVC ovo	0,83	0,83	0,83	0,82
SGD	0,81	0,81	0,82	0,81
SVC	0,82	0,82	0,82	0,82
SVC ovo	0,82	0,82	0,82	0,82

Figura 7.55: Andamento CV MNB

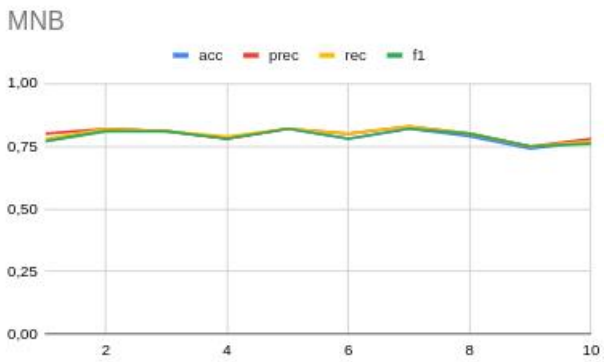


Figura 7.56: Andamento CV RF

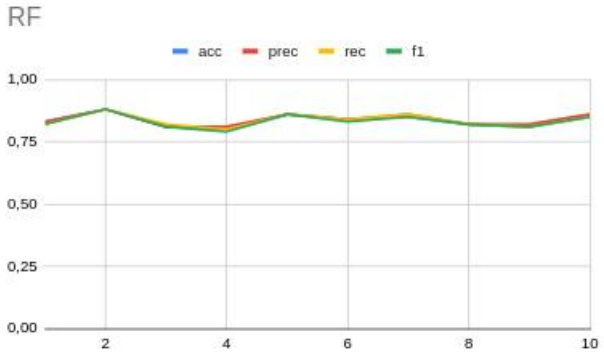


Figura 7.57: Andamento CV linear SVC

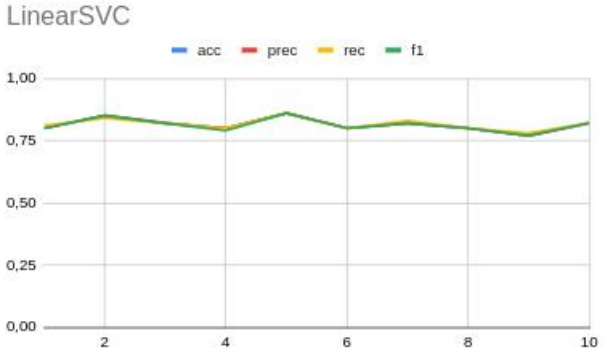


Figura 7.58: Andamento CV linear SVC ovo

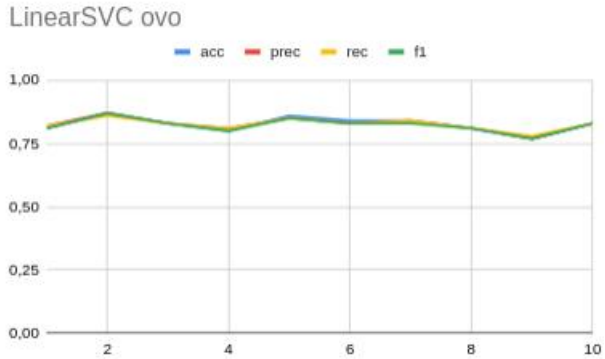


Figura 7.59: Andamento CV SGD

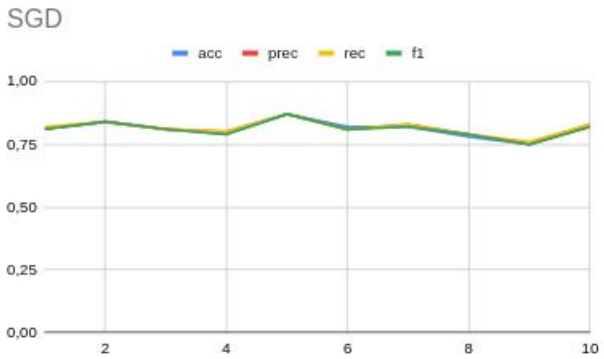


Figura 7.60: Andamento CV SVC

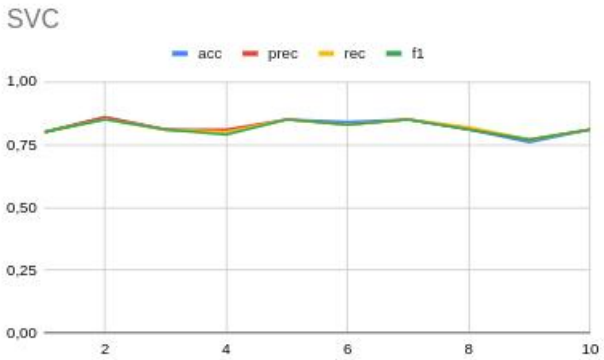
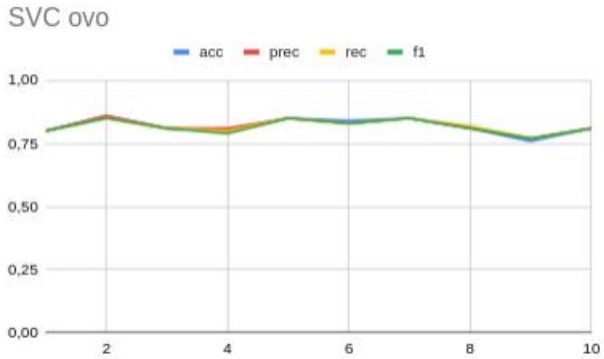
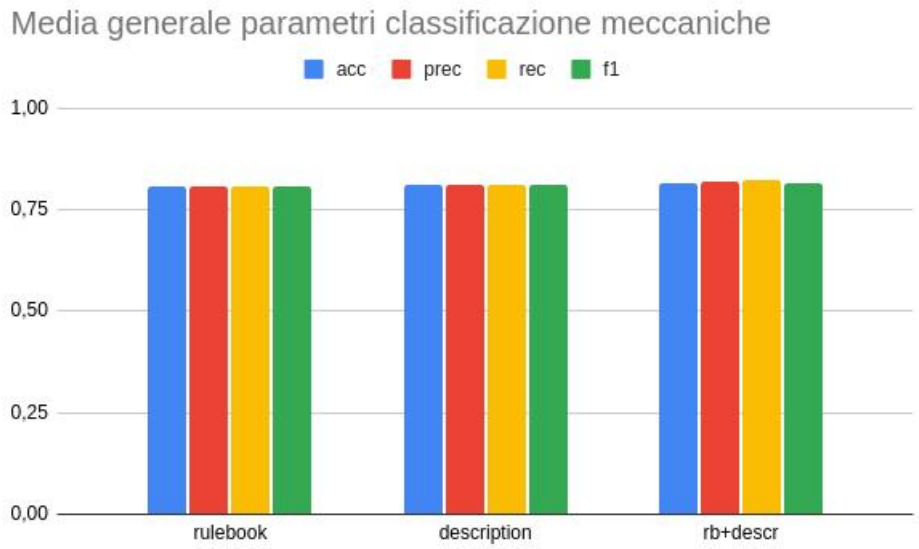


Figura 7.61: Andamento CV SVC ovo



7.2.4 Osservazioni sui risultati

Figura 7.62: Media generale per la classificazione delle categorie



I grafici riportati nelle sezioni precedenti, come per le meccaniche così anche per le categorie, vogliono illustrare i parametri, accuracy precision recall ed F_1, evolvono durante la cross validazione. Anche qui si è optato, per ogni algoritmo e per i vari campi testuali, riportare un plot che illustrasse tale andamento. Nelle rispettive tabelle si sono riportate le medie di ogni grandezza ottenuta in ogni split, per avere una visione decisamente più compatta della situazione. Ad eccezione del MNB, che comunque presenta ottime performance, bene o male tutti i rimanenti algoritmi mostrano delle performance importanti. Si riportano due grafici riassuntivi della situazione attuale. Il primo, figura 7.62 viene riportata per ogni campo testuale usato per classificare, una *media delle medie di tutti gli algoritmi.* Qui, a differenza della classificazione per meccaniche, *l'uso della sola descrizione rispetto al solo manuale fa ottenere performance quasi equivalenti. L'uso di entrambi porta ad un sensibile miglioramento.* In figura 7.35 si ha un plot che mostra due aspetti importanti:

- le prestazioni confrontate in termini di grandezze di machine learning fra tutti gli algoritmi utilzzati. Emerge come il Random Forest sia il più efficiente sotto tutti i punti di vista per questa classificazione
- per ogni algoritmo sono riportate le performance utilizzando il solo rulebook, la sola descrizione ed entrambi. La situazione è equivalente pressoché per tutti gli algoritmi: miglioramento delle prestazioni se usati in coppia.

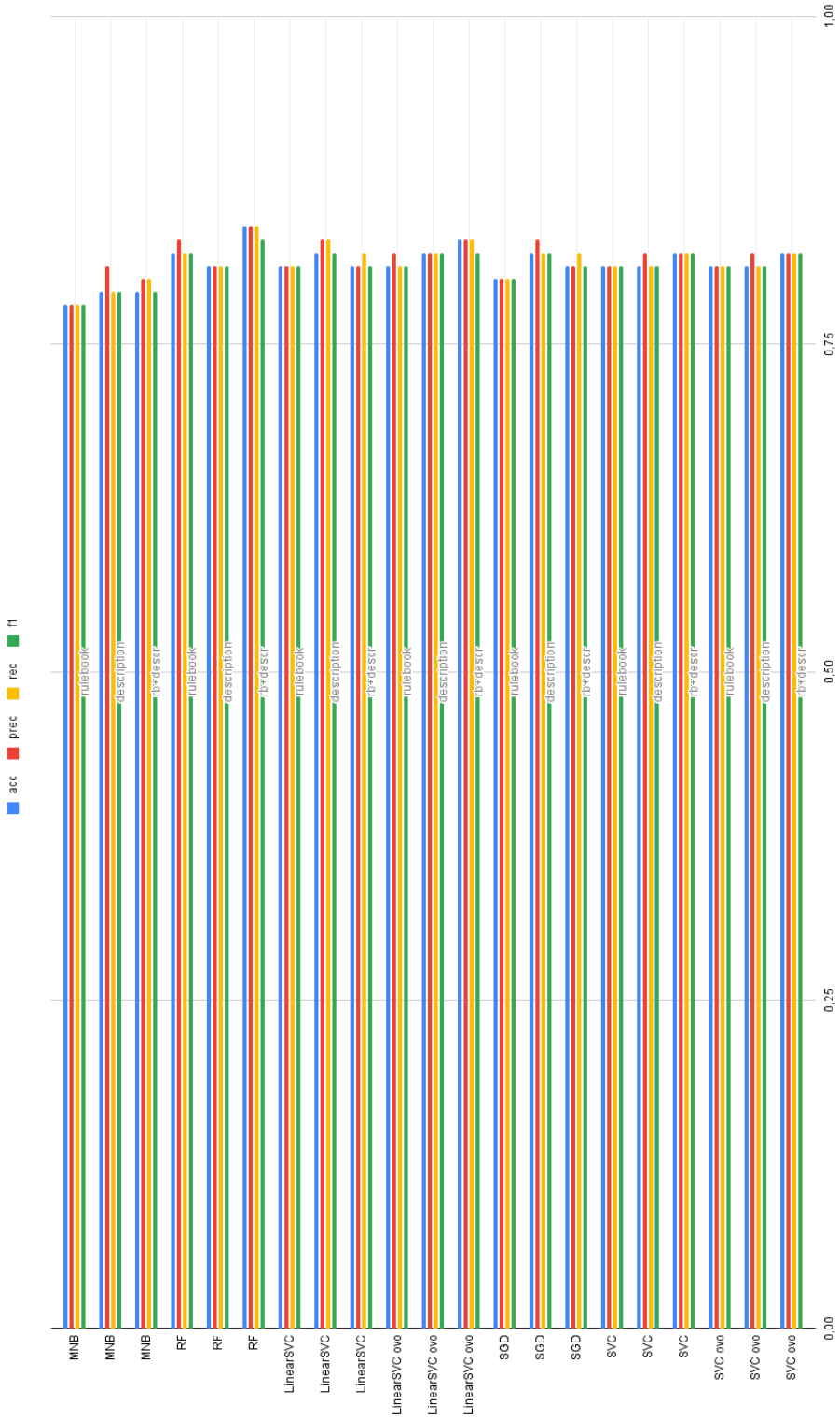


Figura 7.63: Check performance per i vari campi testuali nella classificazione per categorie

7.2.5 Focus sul Random Forest, analisi most important feature

Anche per le categorie viene fatto un focus sui risultati ottenuti all'interno dell'algoritmo Random Forest. Discorso analogo per le categorie, come illustrato per le meccaniche, tenendo sempre le prime 5 più frequenti (*Variable Player Powers, Card Drafting, Set Collection, Dice Rolling, Hand Management*). Anche in questa situazione si sono prese le 20 feature più importanti per ognuno dei 3 casi, ovvero rulebook, descrizione ed entrambi. Usando solo il manuale, le feature che pesano di più sono *dice, roll, unit, card, roll dice* .... Con la sola descrizione si hanno invece *dice, card game, cards, roll,card,*..., ciò si evince guardando rispettivamente le figure 7.64 e 7.65. Nella classificazione per categorie si ricorda come si abbiano dei miglioramenti usando entrambi i campi testuali, e ciò trova riscontro nel fatto che *dice* sia la feature più importante sia usando solo il rulebook, sia usando la sola descrizione, inevitabilmente lo è anche se si usano entrambi. *Roll* è un'altra feature pesante in entrambi i casi (ovvero solo rulebook al secondo posto, solo descrizione al quarto posto), la sua importanza aumenta anche se si usano entrambi i campi di testo (la seconda più importante).

Figura 7.64: Top 20 feature per importanza usando il manuale

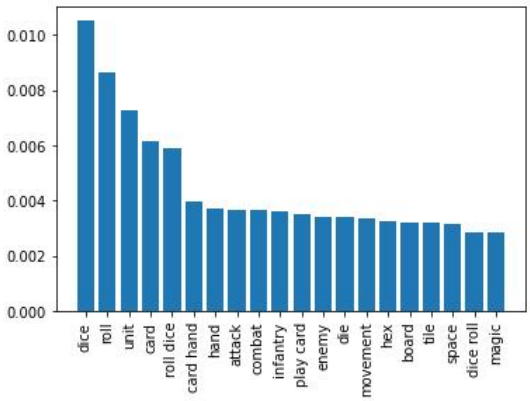


Figura 7.65: Top 20 feature per importanza usando la descrizione

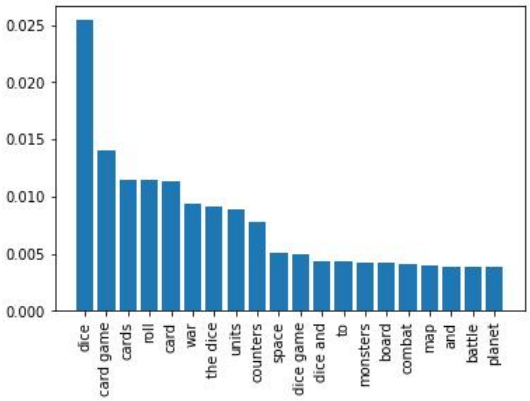
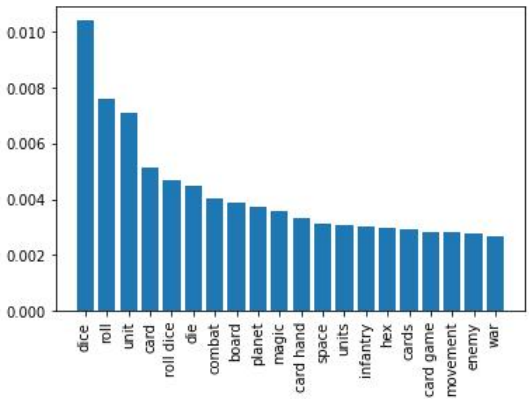


Figura 7.66: Top 20 feature per importanza usando entrambi



Conclusioni e sviluppi futuri

In questo progetto di tesi, gli ambiti che si sono toccati e le problematiche affrontate sono state molteplici, su più sfumature, inerenti al Machine learning, al reperimento di dati più o meno strutturati volti alla creazione di veri e propri dataset utilizzati per le analisi riportate lungo tutto il progetto.

In particolare per questo progetto si è scelto di approfondire una branca scientifica che è ancora ampiamente in via di esplorazione, la *Game Science Data Analytics.* L'idea è partita dal Computationa Thinking, che si è solo accennato. Il focus degli studi invece si è diretto su ciò che può influenzare il CT. Ecco perché ci si è decentrati da quest'ultimo, per concentrarsi invece su ciò che lo innesca: le meccaniche e categorie dei boardgame.

Si è partiti quindi dallo studio di meccaniche e categorie, come il *card drafting, wargame, dice rolling, dice, set collection* .... Esse sono presenti in maniera intrinseca in ogni boardgame, tale per cui è stato necessario un approfondimento circa il dove ricavare le informazioni atte a poter costruire modelli di machine learning. Si è sfruttata quindi tutta la conoscenza che il sito BGG metteva a disposizione, per scaricare più pagine di boardgame possibili da cui poter attingere ogni sorta di informazione, tra cui proprio le meccaniche/categorie di ogni gioco da tavolo. Rappresentando le meccaniche/categorie delle vere e proprie etichette che caratterizzano i boardgame, si è reso necessario estrarre direttamente dalle pagine dei boardgame stessi scaricate da BGG anche dei campi dei testuali, come la *description.* Tramite tecniche di *text analytics*, è stato possibile modellare le description di ogni boardgame, affinché potessero essere usate per creare modelli matematici atti a generare conoscenza.

Il passo successivo è risultato più complesso del precedente. Si è reso necessario approfondire la conoscenza usando un'altra tipologia di testo, differente dalla description: *la manualistica dei boardgame.* Il problema, estremamente complesso, è stato scomposto in problemi di natura più trattabile. Così è stato possibile scaricare i rulebook di più boardgame possibili, sempre da BGG, con tutte problematiche del

caso esposte nel progetto di tesi. Estrapolata tutta l'informazione da ogni rulebook per ogni boardgame, sempre tramite tecniche di *text analytics* si sono costruiti di ulteriori modelli di machine learning utilizzando come campo testuale per classificare i boardgame il contenuto dei rulebook. Si sono effettuati i confronti a livello di prestazioni fra i modelli che usavano solamente la description, o solo il rulebook, o l'unione di entrambi i campi.

Prima di questo passo importante, ci si è posti un quesito importante, ovvero il come riconoscere ciò che è un manuale da ciò che non lo è. Questo problema ha trovato la soluzione proposta nell'elaborato. Si è quindi etichettato manualmente ogni rulebook, aprendolo e verificando la sua veridicità. Così facendo si è potuto costruire un dataset, la cui natura è stata data da un controllo umano, che si presta ad essere usato per il machine learning supervisionato. Di conseguenza si è costruito un modello che possa riconoscere manuali e non. Così facendo si può eventualmente allargare la quantità di manuali da poter inserire nel complesso. In mezzo a questo problema, si è inserito anche un aspetto che ha trovato spazio, ovvero come filtrare i manuali correttamente utilizzabili e scartando i rulebook, che pur essendo tali, non si sono potuti usare, per motivi già ampiamente citati. Ciò è stato fatto vedere nell'elaborato in due maniere, manualmente o tramite algoritmo. Così facendo si è andata a proporre una infrastruttura che potesse automatizzare il filtraggio, senza dover togliere a mano i manuali non utilizzabili.

Con uno sguardo orientato sugli sviluppi che questo progetto può avere nell'immediato futuro, sarebbe decisamente interessante procedere provando a estrapolare ulteriori informazioni, e capire come e se altri campi dei boardgame possano influenzare la costruzione e la classificazione dei boardgame in determinate categorie. Più precisamente ci sono una moltitudine di campi che non sono stati utilizzati perché si è optato più per lavorare sull'informazione testuale. Il problema oggetto di studi in questo progetto di tesi, per renderlo più trattabile, è stato semplificando supponendo che ogni boardgame avesse solo una categoria/meccanica, quella più frequente interna al dataset. Sarebbe decisamente interessante focalizzarsi su altri approcci di tipo multilabeling per classificare boardgame in maniera più attenta su una, più o nessuna classe.

Non si può certo escludere un'altra branca dell'intelligenza artificiale che non è stata oggetto di studi in questo progetto, ma su cui si potrebbero avere dei riscontri importanti, a livello di risultati: *il deep learning*. Attraverso costruzioni di reti neurali, si possono creare degli ottimi classificatori basati sui neuroni che emulano il cervello umano. In questo progetto di tesi più volte si è fatto presente che la

quantità di campioni utilizzati lasciasse un poco a desiderare. Avere una numerosità a livello di campioni decisamente alta può garantire dei risultati ottimi. Quelli che si sono ottenuti in questo progetto sono comunque incoraggianti e obiettivamente migliorabili cercando di ottenere sempre di più manuali per avere classificatori sempre più precisi.

Bibliografia

Riferimenti cartacei

[1] *Liu, Yuxi. Python Machine Learning By Example. 2nd ed. Packt Publishing, 2019. Web. 28 Sept. 2021.*

[2] *R. Martoglia, M. Pontiroli (2021): Let the Games Speak by Themselves: Towards Game Features Discovery Through Data-Driven Analysis and Explainable AI. IEEE International Conference on Data, Information, Knowledge and Wisdom (DIKW), Haikou, China, 2021.* . URL: `http://www.isgroup.unimore.it/publications.html`.

[3] *Katerina Tsarava, Korbinian Moeller, Manuel Ninaus, Training Computational Thinking through board games: The case of Crabs and Turtles.* URL: `https://www.researchgate.net/publication/326002159_Training_Computational_Thinking_through_board_games_The_case_of_Crabs_Turtles`

[4] *Random Forests, Cutler, Adele and Cutler, David and Stevens, John,2011,Machine Learning - ML*

[5] *Stochastic Gradient Descent in Theory and Practice, Panos Achlioptas, Stanford.* URL: `https://ai.stanford.edu/~optas/data/stanford_qual_exams.pdf`

Siti web consultati

[6] *game analytics*. URL: https://cmps-people.ok.ubc.ca/bowenhui/game/readings/ch2-game-metrics.pdf

[7] *BoardGame Geek*. URL: https://boardgamegeek.com/.

[8] *python*. URL: https://www.python.org/.

[9] *theses-proposed*. URL: https://www.isgroup.unimo.it/theses-proposed.html.

[10] *xml-basic*. URL: https://www.geeksforgeeks.org/xml-basics/.

[11] *API-bgg*. URL: https://boardgamegeek.com/wiki/page/BGG_XML_API2.

[12] *BeautifulSoup*. URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/.

[13] *Pandas*. URL: https://pandas.pydata.org/.

[14] *Selenium*. URL: https://www.selenium.dev/.

[15] *pypdf2*. URL: https://pypi.org/project/PyPDF2/.

[16] *textract*. URL: https://textract.readthedocs.io/en/stable/.

[17] *sklearn*. URL: https://scikit-learn.org/stable/.

[18] *jupyter*. URL: https://jupyter.org/