DIPARTIMENTO DI SCIENZE ECONOMICHE E SOCIALI

CORSO DI LAUREA IN: DATA SCIENCE PER L'ECONOMIA E LE IMPRESE

USO DI TECNICHE DI EDUCATIONAL PROCESS MINING PER L'IDENTIFICAZIONE DI PATTERN NELLA CARRIERA DEGLI STUDENTI UNIVERSITARI.

USE OF EDUCATIONAL PROCESS MINING TECHNIQUES TO IDENTIFY PATTERNS IN THE CAREER OF UNIVERSITY STUDENTS.

Tipo tesi: Di ricerca

Relatore:
PROF./DOTT. DOMENICO POTENA

Studente:
GIANMARCO VIGANÒ

ANNO ACCADEMICO 2023/2024

SOMMARIO

ELENCO DELLE TABELLE

ELENCO DELLE FIGURE

I. INTRODUZIONE

I.1 EDUCATIONAL PROCESS MINING E LA SUA UTILITA' PER GLI STUDENTI ITALIANI

Nel campo dell'istruzione superiore in Italia, gli studenti incontrano spesso sfide che possono influenzare significativamente i loro progressi e risultati accademici. Queste sfide includono la navigazione in percorsi di studio complessi, la gestione di un supporto accademico incoerente e il superamento delle barriere che impediscono il conseguimento tempestivo della laurea. Questi problemi, incentrati sullo studente, sottolineano la necessità di soluzioni innovative che migliorino l'esperienza educativa e promuovano il successo degli studenti. Questa tesi indaga il potenziale dell'Educational Data Mining (EDM) e dell'Educational Process Mining (EPM) per affrontare direttamente queste sfide, offrendo intuizioni e strumenti per migliorare il modo in cui gli studenti interagiscono con i loro ambienti educativi.

Le università italiane, come molte altre in tutto il mondo, vedono crescere le richieste di supportare al meglio le esigenze degli studenti in un panorama educativo dinamico. Gli studenti devono spesso affrontare sfide legate alla progressione dei corsi, alla chiarezza dei requisiti accademici e alla personalizzazione delle esperienze di apprendimento. Gli alti tassi di abbandono e il prolungamento dei tempi di laurea evidenziano la necessità di pratiche educative più efficaci e adatte alle esigenze dei singoli studenti.

All'interno del più ampio ambito dell'Educational Data Mining (EDM), che applica tecniche di data mining per analizzare i contesti educativi, si trova un potente sottoinsieme chiamato Educational Process Mining (EPM). Sebbene l'EDM abbia un valore inestimabile per identificare modelli e prevedere tendenze attraverso tecniche statistiche e di apprendimento automatico, generalmente trascura i dati sequenziali e procedurali che possono rivelare le dinamiche delle interazioni educative.

L'EPM estende le capacità del data mining tradizionale concentrandosi specificamente sui processi didattici. Analizza il flusso degli eventi e la loro tempistica all'interno dei dati educativi, ricostruendo i percorsi che studenti e docenti percorrono attraverso i sistemi educativi e amministrativi. Utilizzando tecniche di Process mining, questa ricerca mira a visualizzare e analizzare i flussi di lavoro coinvolti negli esami degli studenti, identificando colli di bottiglia e deviazioni dai processi educativi ottimali. Questo approccio analitico non solo evidenzia le inefficienze, ma propone anche spunti attuabili per semplificare le operazioni e migliorare il rendimento accademico.

Questa tesi utilizza quindi il Process mining per affrontare le sfide identificate nelle università italiane. Analizzando un set di dati che descrive in dettaglio gli esami degli studenti, coprendo date degli esami, punteggi e variabili aggiuntive, raccolti grazie ai sistemi informatici che ormai caratterizzano qualsiasi Università, come ad esempio Esse3 Web, con questa ricerca si applicheranno diversi strumenti come ad esempio Panda e pm4py, che sono rispettivamente librerie di python per manipolazione dei dati e per il Process mining per costruire un modello completo del processo di esame. Si prevede che le conoscenze acquisite da questo studio forniranno strategie in grado

di migliorare l'accuratezza e l'efficienza dei processi accademici e, in definitiva, migliorare l'esperienza educativa per gli studenti.

Questo studio però non cerca solo di migliorare i processi accademici, ma anche di responsabilizzare gli studenti fornendo loro approfondimenti e strumenti che possano aiutarli a percorrere i loro percorsi accademici in modo più efficace. Rendendo questi processi più trasparenti e ottimizzati, gli studenti sono in una posizione migliore per avere successo, sapendo cosa aspettarsi e come gestire al meglio i propri progressi educativi.

I.2 IL CASO STUDIO

Le tecniche precedentemente introdotte sono state poi implementate in un caso reale di studio, che si basa sugli studenti del corso di Ingegneria Informatica e dell'Automazione presso l'Università Politecnica delle Marche nell'arco di più di 10 anni accademici (dall'anno accademico 2010/2011 all'anno accademico 2023/2024).

I dati utilizzati sono stati innanzitutto estratti tramite la piattaforma digitale Esse3, utilizzata da moltissime università italiane per immagazzinare e gestire dati di diversa tipologia relativi agli studenti come, ad esempio, l'iscrizione dello stesso, la registrazione dei corsi, degli esami sostenuti con relativi risultati e i progressi accademici. Per questo motivo Esse3 non solo facilita le operazioni amministrative quotidiane delle università, ma accumula anche grandi quantità di dati che sono di fondamentale importanza per analizzare e migliorare i processi educativi.

Questa infrastruttura online ha quindi il ruolo di fonte primaria di dati, che forniscono un livello fondamentale per l'applicazione delle tecniche di Educational Data Mining (EDM) e più in particolare di Educational Process Mining (EPM), dato

che possiamo visualizzare in maniera veloce ed efficace il percorso svolto da ogni studente.

Nel nostro caso studio quindi, basandoci sui dati estratti eseguiamo un'analisi che si articola su tre diverse categorie che dividono gli studenti in esame:

- *Early*: composta dagli studenti che si sono laureati in tempo, ossia, essendo tutti studenti di una triennale, nell'arco di 3 anni e 6 mesi dall'iscrizione;
- *One Year Late*: sono gli studenti fuori corso ma solamente di un anno e che quindi si sono laureati tra i 3 anni e 6 mesi e i 4 anni e 6 mesi dalla data di iscrizione;
- *Late*: sono tutti gli altri studenti che si sono laureati dopo 4 anni e 6 mesi dal giorno in cui si sono iscritti all'università.

Basandosi su queste tre categorie di studenti si andrà quindi ad effettuare un'analisi che cerca di individuare ed evidenziare quelli che sono i pattern di processo più frequenti innanzitutto tra coloro che si sono laureati in tempo e possibilmente con i voti migliori, e poi tra coloro che invece ci hanno messo di più a laurearsi, in modo tale da capire, rispettivamente, cosa è consigliabile fare per un percorso di studi lineare ed efficiente e quali possono essere d'altra parte le problematiche più comuni riscontrate nei percorsi di chi ha avuto più difficoltà nel conseguire il titolo di studi, in un'ottica futura che può permettere di intercettare questi problemi e rendere più piacevole e lineare il percorso di studi di più studenti possibile.

I.3 PANORAMICA DEL DATASET

Il dataset utilizzato per l'analisi, come detto precedentemente, è costituito dai dati relativi agli studenti dall'anno accademico 2010/2011 al 2023/2024 e che hanno conseguito il titolo[1] presso la triennale in Ingegneria Informatica e dell'Automazione all'Università Politecnica delle Marche.

Per costruire il dataset completo sono stati utilizzati tre diversi file .csv che si dividevano i dati in questo modo:

- Nel primo file si trovano i dati relativi solamente agli esami aventi esito positivo e perciò con conseguente votazione;
- Nel secondo file sono invece presenti i vari stati e quindi step che hanno portato al conseguimento dello specifico esame;
- Il terzo file è invece costituito da tutte le caratteristiche relative all'anagrafica dello studente.

L'analisi è stata effettuata parallelamente sulla base di due diversi dataset: df_esami che ha al suo interno il percorso degli studenti basato solamente sugli esami passati; quindi, è in pratica lo stesso contenuto del primo file e un altro dataset denominato df_merged che invece deriva da un left join effettuato tra tutti e tre i file, nel quale si trovano quindi tutte le informazioni sia riguardanti ogni studente che il percorso completo (con tanto di stati relativi agli esami) degli stessi.

Entrambi sono composti da 785 studenti, di cui ognuno è distinto da un identificatore univoco che garantisce l'anonimato e altre informazioni relative all'esame che è stato dato, il rispettivo voto con cui è stato passato e professore che tiene il corso e la data

[1] Vengono presi in considerazione solamente coloro che hanno già conseguito il titolo in modo tale da avere una visione più chiara e completa per quanto riguarda il percorso svolto.

in cui l'esame è stato superato. La differenza principale che contraddistingue il dataset "merged" è data dal fatto che quest'ultimo è costituito da tutte le variabili precedentemente elencate più altre informazioni che ci evidenziano quello che è stato il percorso che ha portato al superamento di un determinato esame da parte dello studente tramite una nuova colonna, chiamata "STA_REG_DES", che definisce lo stato dell'esame (prenotato, chiuso, verbalizzato, caricato) .A questa informazione ne è poi collegata un'ulteriore, che può riscontarsi o nella colonna "GIUD_ESA_DES" o in "TIPO_NOVERB_DES", rispettivamente se il valore di "STA_REG_DES" è verbalizzato o chiuso. Difatti nel primo caso si troverà, sotto la variabile "GIUD_ESA_DES", insufficiente se l'esito è stato verbalizzato ma lo studente non ha passato l'esame, quindi l'esame è stato sostenuto e consegnato dallo studente; mentre nel secondo caso, per la colonna "TIPO_NOVERB_DES", si può trovare o "Assente" (lo studente si è prenotato all'appello, ma non si è presentato e quindi non ha sostenuto l'esame), oppure "Respinto/Ritirato" (lo studente si è presentato all'esame, ma conseguentemente non ha consegnato la prova d'esame, per sua volontà o per volontà del professore che lo ha respinto).

Inoltre, sono presenti, rispetto al dataset "esami", anche tutte le informazioni relative all'anagrafica degli studenti, come ad esempio il sesso, l'istituto superiore in cui si è maturato (esplicitato tramite codice) e la votazione con cui ha conseguito la maturità.

Questi dati verranno quindi utilizzati congiuntamente permettendo un'analisi accurata riguardante i percorsi di ogni studente per entrambi i dataset, con una concentrazione maggiore su quello con anche le informazioni riguardanti gli stati degli esami.

I.4 STRUTTURA DELLA TESI

La tesi si articolerà poi nei seguenti capitoli.

Si parte dal capitolo 1 che introduce i concetti teorici che si troveranno nel proseguo della tesi, i quali permettono di avere visuale teorica che permetta di comprendere al meglio l'analisi, quali Process Mining, con relative tecniche e algoritmi utilizzati per estrarre i modelli di processo, come inductive, heuristic e fuzzy miner, e Reti di Petri per la relativa visualizzazione, oltre ovviamente al concetto di grafi diretti e sottografi. Verranno infine spiegate le tecniche di clusterizzazione utilizzate nella parte finale della tesi oltre che gli strumenti che sono stati fondamentali per il progetto.

Si prosegue con il capitolo 2 che delinea la metodologia alla base dei vari step del processo, con particolare attenzione al pre-processing dei dati.

Il capitolo 3, attraverso le statistiche relative ai vari studenti, sia in generale che poi approfondendo il primo anno, ossia l'anno cruciale per la maggior parte degli studenti, ci dà di base la spiegazione per cui tutte le successive tecniche sono state scelte ed implementate. Verranno illustrate tabelle e grafici generati con Power BI.

Passando al capitolo 4 ci si addentra nel cuore dell'analisi, descrivendo i modelli di processo, come sono stati costruiti attraverso i due diversi algoritmi utilizzati (inductive e heuristic) e con conseguente confronto finale sui risultati derivanti dall'applicazione di entrambi evidenziando possibili somiglianze e differenze.

Il capitolo 5 offre una visione dettagliata di come è stato implementato il conformance checking per le varie categorie di studenti, confrontando anche i risultati in termini di fitness.

Proseguendo col capitolo 6 ci si concentrerà sull'analisi dei grafi riferiti ai percorsi di ogni singolo studente, tramite in un primo momento la costruzione dei grafi diretti e poi, tramite lo strumento Subdue, dei sottografi, che verranno utilizzati nell'ultima fase dell'analisi.

Difatti il settimo e ultimo capitolo si basa proprio sui grafi precedentemente costruiti, andando ad effettuare una clusterizzazione tra i percorsi e ne verranno spiegati sia le logiche alla base che i risultati.

L'obiettivo è dunque quello di evidenziare, attraverso l'applicazione del K-Means, i pattern più frequenti tra i vari cluster, oltre ad evidenziare quelli che sono "tipici" di un particolare cluster e quindi presenti solamente tra gli studenti che lo costituiscono, in modo tale da poter acquisire informazioni utili sia per l'Università, che per gli studenti futuri.

Queste informazioni potrebbero consistere sia nel mettere in evidenza i pattern più frequenti tra gli studenti più performanti, in modo tale da definire un percorso ottimale, sia nel sottolineare le "lacune" più significative tra quelli meno performanti, per poter poi nel futuro riuscire ad individuare studenti che seguono gli stessi pattern e intercettarli il prima possibile offrendo loro aiuto e spunti per correggere il loro percorso accademico.

A terminare la tesi si troveranno le conclusioni dove verranno discussi i risultati dell'analisi e i limiti degli approcci utilizzati definendo anche spunti per eventuali progetti e sviluppi futuri.

Capitolo 1: CENNI TEORICI

In questo capitolo si andrà a porre la base delle conoscenze teoriche che sono necessarie per comprendere al meglio le analisi che sono state effettuate e i risultati che ne derivano.

Verranno quindi descritti in linea teorica gli argomenti riguardanti il Process Mining, le Reti di Petri, gli algoritmi implementati quali Inductive, Heuristic e Fuzzy miner, per poi introdurre il conformance checking, i concetti relativi ai grafi e le tecniche di machine learning utilizzate per la clusterizzazione, concludendo con un accenno anche agli strumenti che sono stati utilizzati durante il progetto.

1.1 PROCESS MINING

Il Process Mining è un campo di ricerca che sta cominciando ad assumere molta importanza soprattutto negli ultimi anni e che si concentra sull'analizzare i dati che vengono lasciati da eventi che vengono eseguiti in diversi ambiti applicativi.

L'attenzione posta su questa disciplina deriva principalmente dal fatto che, a differenza delle tecniche più classiche del data mining, si concentra sui processi end-to-end e in particolare sulle tracce che li definiscono.

Il Process Mining è volto quindi a scoprire, tenere sotto controllo e di conseguenza migliorare i processi tramite la conoscenza che possiamo definire dagli event log che al giorno d'oggi sono facilmente disponibili tramite i sistemi informativi.

Il punto di partenza su cui si basano tutte le tecniche di Process mining è l'event log, registri elettronici all'interno dei quali ogni evento si riferisce ad una attività che si riferisce a sua volta a un particolare caso più diverse informazioni legate a quell'attività o quel caso, come ad esempio il voto che ha preso lo studente che ha svolto l'attività.

Ma gli elementi fondamentali che devono essere presenti per avere un event log definito adeguatamente sono: "Case ID", "Event ID", "Activity" e Timestamp".

Il "Case ID" è la chiave identificatrice dei casi, ossia dei processi, in particolare quindi tutte le righe dell'event log che presentano la stessa chiave fanno riferimento allo stesso processo, in modo tale da poter ricostruire il processo per intero in maniera semplice e immediata.

L'" Event ID" è invece l'identificatore dello specifico evento all'interno di un processo mentre "Activity" e "Timestamp" sono rispettivamente gli attributi dell'event log che si riferiscono all'attività svolta (esame nel nostro caso) e data in cui l'esame è stato sostenuto".

Il Process Mining si articola in tre tecniche principali che si basano su appunto questi event log e che sono: Discovery, Conformance Checking ed Enhancement. [2] [3]

[2] Wil van der Aalst – Process Mining Manifesto (2012)
[3] Wil van der Aalst – Process Mining Manifesto (2012)

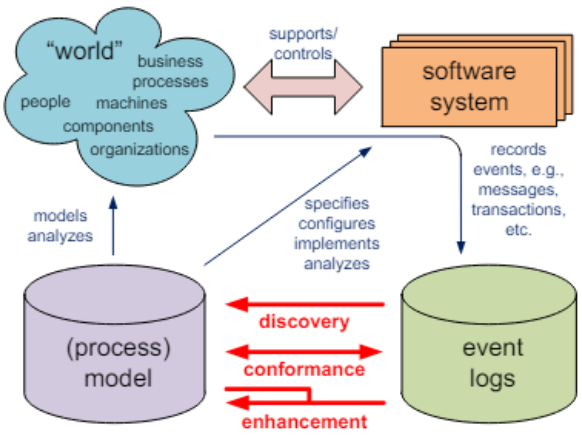

Figura 1.I: Posizionamento delle 3 principali tecniche di Process mining.

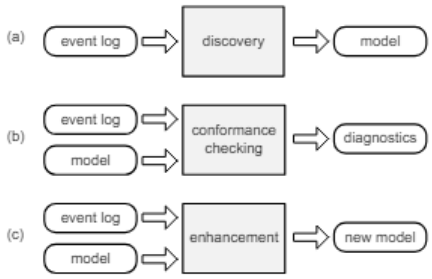

Figura 1.II: Le tre principali tecniche di Process mining spiegate in termini di input e output: (a) Discovery, (b) Conformance checking, (c) Enhancement.

Partendo dalla tipologia di tecniche, ossia il Process Discovery, parliamo di delle tecniche che hanno l'obiettivo di estrarre il processo dei dati, e quindi produrre un modello per spiegare il comportamento partendo dall'event log che abbiamo a disposizione, senza informazioni a priori disponibili (cioà oltre a quelle fornite dall'event log).

Si passa poi al Conformance Checking, che utilizza un modello di processo standard che viene precedentemente definito, che coincide con quello che dovrebbe essere il modello di processo ideale e con il quale vengono poi confrontati i modelli di

processo reali, ossia quelli che si sono effettivamente realizzati nella realtà. Questo perché l'obiettivo della conformance checking è quello di misurare quanto i processi reali sono conformi con quello ideale. Come possiamo notare dalla figura 1, rispetto al Process Discovery che ha una direzione unilaterale dall'event log al modello di processo, nel conformance checkin la direzione è bilaterale perché possiamo sia come appena detto confrontare i processi effettivamente realizzati con il processo ideale, ma anche viceversa. Infine una volta evidenziate quelle che sono le deviazioni tra processo reale e ideale tramite conformance checking, queste si possono analizzare e spiegare.

L'ultima delle 3 tecniche è infine l'Enhancement, il cui obiettivo è migliorare e ottimizzare il modello di processo a disposizione andando o a "riparare" quest'ultimo, attuando modifiche che permettono di riflettere meglio la realtà, oppure estendendolo, ossia aggiungendo dati che fanno riferimento alle performance o informazioni sulle risorse. Conformance checking ed enhancement possono inoltre essere utilizzate congiuntamente, quando ad esempio troviamo, tramite la prima, una deviazione che apporta risultati positivi al modello.

Concludendo, si può dire che il punto cruciale del Process mining è stabilire una connessione tra il modello di processo e la "realtà" rappresentata dai dati.

Perciò il termine "Play-in" si riferisce al processo di estrazione del modello a partire dall'event log. Si inizia con un log che descrive la sequenza degli eventi, da cui viene costruito uno schema che rappresenta tutte le sequenze possibili.

Il "Play-out", invece, riguarda l'uso tradizionale dei modelli di processo: si parte dal modello (che può essere ad esempio, una rete di Petri) e si genera un "comportamento". Questo approccio è spesso impiegato per la simulazione,

permettendo di simulare diverse possibili esecuzioni del processo ed estrarre statistiche e intervalli di confidenza.

Il termine "Replay" si riferisce all'identificazione delle discrepanze tra la realtà e il modello estratto. Il Replay utilizza l'event log e il modello di processo come input e verifica per ogni elemento se c'è corrispondenza tra il processo e i dati.

1.2 RETI DI PETRI

Le reti di Petri sono uno strumento di modellazione matematica teorizzato da Carl Adam Petri all'inizio degli anni 60 e che viene utilizzato per descrivere e analizzare il flusso di informazioni principalmente nei sistemi concorrenti e distribuiti.

Le componenti principali delle reti di Petri sono i *places*, le transizioni e i *token*, dove i primi rappresentano gli stati del sistema, le seconde gli eventi che possono cambiare lo stato all'interno del processo e gli ultimi non fanno parte della vera e propria struttura che definisce la rete di Petri, ma vengono utilizzati per contrassegnare i *places* indicando lo stato corrente del sistema.

Come abbiamo detto questi ultimi non fanno parte della struttura vera e propria della Petri net perché questa è graficamente un grafo bipartito dove abbiamo due tipi di nodi: i *places* che vengono rappresentati solitamente tramite cerchi e le transizioni che vengono invece rappresentate comunemente come rettangoli. Questi nodi sono collegati tramite archi diretti direzionati.

All'interno dei *places* possiamo trovare dei *token* che possono rappresentare ad esempio risorse o dati e quando una transizione viene eseguita, uno o più di questi

token possono essere consumati nei *place* d'ingresso della transizione e generati poi nei *places* di uscita.

Quando nella rete di Petri un'attività è collegata a due rami in uscita, significa che le due attività a cui è collegata devono essere eseguite simultaneamente, mentre se invece da un *place* si diramano due attività, si effettua una scelta esclusiva, ossia o si sceglie un'attività o l'altra.

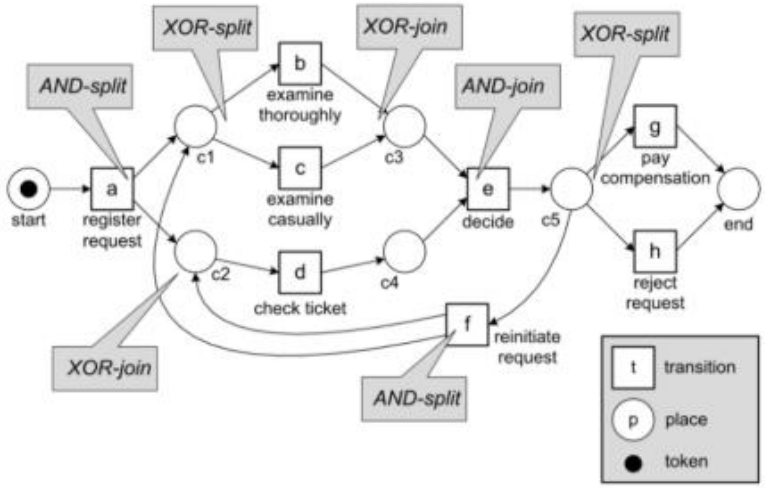

Figura 1.III: Struttura della Petri net.

Lo stato in cui una Petri net si trova in un determinato momento è dato dalla posizione dei *token* nei vari *places*, ossia dal marking, che fondamentalmente è un insieme di *place* con *token*, che ci mostra appunto quali luoghi contengono questi ultimi in uno specifico momento. Il marking iniziale ci definisce quindi lo stato della Petri net all'avvio del processo, specificando quindi quali *place* hanno i *token* al momento dell'avvio del sistema.

Il comportamento dinamico di una rete di Petri con marking è governato dalla cosiddetta "firing rule", anche chiamata regola di attivazione. Una transizione può essere eseguita se ciascuno dei suoi place di input contiene almeno un token. Quando

la transizione viene eseguita, consuma un token da ogni place di input e genera un token per ogni place di output.[4]

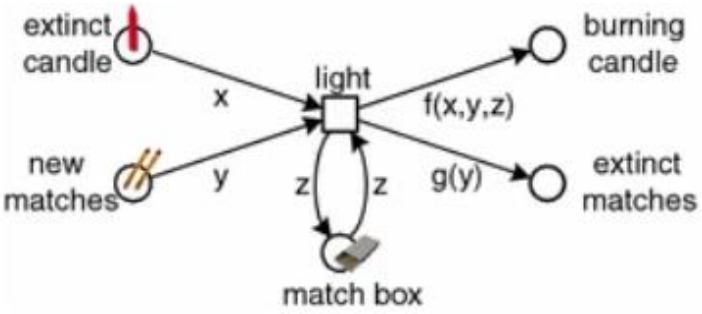

Figura 1.IV: Prima dell'attivazione.

[5]

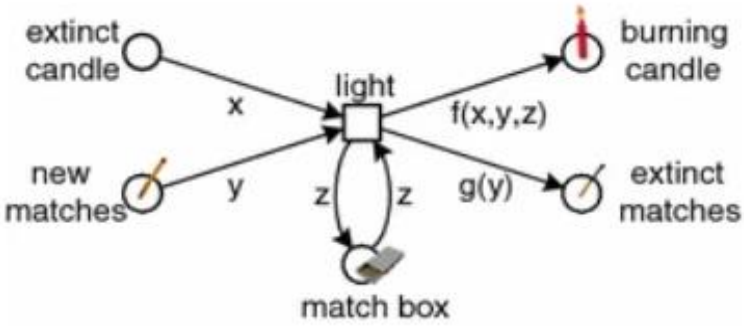

Figura 1.V: Dopo l'attivazione.

Esisono poi diverse tipologie di reti di Petri che si basano su alcune proprietà di quest'utlime. Ad esempio una Petri net può essere *safe* se, in ogni marking raggiungibile, ogni *place* contiene al massimo un *token*, queste reti sono particolarmente utili per la modellazione di sistemi in cui è necessario evitare conflitti di risorse, garantendo quindi che nessun *place* possa avere più token contemporaneamente, il che impedisce un uso eccessivo delle risorse.

[4] Desel, J., & Reisig, W. (2015). The concepts of Petri nets. Software and Systems Modeling
[5] Desel, J., & Reisig, W. (2015). The concepts of Petri nets. *Software and Systems Modeling*

Esisono poi le reti limitate, o anche dette *bounded*, che sono una generalizzazione delle precedenti, e nello specifico una rete è k-limitata se ogni *place* contiene al massimo k *tokens* per ogni possibile marking raggiungibile.
Una rete di Petri è detta viva invece se da ogni marking raggiungibile è possibile attivare una qualsiasi transizione e quindi assicura che il sistema non si blocchi e che tutte le parti del processo possano essere eseguite nelle giuste condizioni.
Infine una Petri net è detta *deadlock-free* se non è possibile raggiungere un marking da cui non sono abilitate transizioni. Questa tipologia di reti di Petri è molto utile per garantire un funzionamento continuo, dato che impedisce al sistema di raggiungere uno stato in cui non possono avvenire ulteriori azioni.

1.3 ALGORITMI

Verranno ora introdotti concettualmente gli algoritmi che andremo ad utilizzare, e in particolare algoritmi di Discovery quali l'inductive miner, l'heuristic miner e il fuzzy miner, che hanno quindi l'obiettivo di costruire dei modelli di processo accurati partendo dalle informazioni che si possono ritrovare negli event logs che si hanno a disposizione. Questi modelli saranno poi estremamente utili per capire come i processi vengono effettivamente realizzati, identificare deviazioni e scoprire delle opportunità per migliorarli. Ciascun algoritmo che verrà introdotto e spiegato avrà ovviamente i suoi punti di forza e debolezza che portano alla scelta di uno o di un altro in base alla situazione in cui si deve implementare, definita dalle caratteristiche dell'event log e l'obiettivo dell'analisi.

1.3.1 Inductive miner

Il primo che verrà presentato ed utilizzato è quindi l'inductive miner, un algoritmo di discovery noto per la sua capacità di produrre dei modelli che sono validi, precisi e generali allo stesso tempo. Funziona dividendo l'event log, in modo ricorsivo, in sottoinsiemi più piccoli, di cui ognuno dei quali rappresenta un comportamento più semplice, i quali vengono combinati dall'algoritmo in modo tale da produrre un modello finale che rifletta accuratamente il processo complessivo.

La particolarità dell'Inductive Miner è che produce una visualizzazione ad albero, la quale permette una rappresentazione gerarchica di un modello di processo.

Per costruire questo albero l'algoritmo parte innanzitutto dal dividere l'event log a disposizione in piccoli sottoinsiemi, come detto precedentemente. Questo *splitting* consiste nell'identificare i punti in cui vi sono delle particolari relazioni tra gli eventi osservati, come ad esempio un parallelismo, una scelta, una sequenzialità diretta o dei *loops*. Questo viene fatto fino a che ogni sottoinsieme non rappresenta una singola attività o una struttura semplice.

Usando poi questi costrutti, viene definito il *process tree*, il quale è costruito partendo dalle strutture più semplici che vengono conseguentemente combinati per crearne di più complesse.

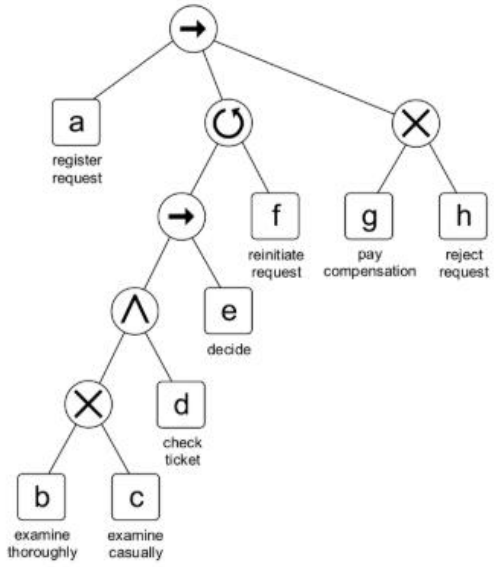

Figura 1.VI: Process Tree.

[6]

Nella figura precedentemente si possono identificare le relazioni tra gli eventi riconoscibili tramite simboli differenti. Nello specifico, partendo dall'attività "a" che è l'attività di start si trova l'operatore sequenziale, definito da una freccia che indica che, in questo caso, l'attività "a" deve essere eseguita sempre prima, perché si trova a sinistra dell'operatore, e successivamente le attività collegate al secondo operatore che si incontra, ossia il *loop*, che rappresenta una ripetizione delle attività che sottostanno ad esso. Dopodiché si possono incontrare anche operatori relativi alla scelta (operatore "or"), che dà la possibilità di scegliere tra una delle attività figlie. Infine, l'operatore parallelo permette determina che le attività figlie devono essere eseguite parallelamente.

[6] Slides: Wil van der Aalst (www.vdaalst.com)

L’albero di processo viene poi trasformato in una rete di Petri in modo da avere un quadro più flessibile e facilmente visualizzabile per la successiva interpretazione dei risultati derivanti dall’implementazione dell’algoritmo. Nella trasformazione in rete di Petri ogni nodo dell’albero corrisponderà a un *place* o transizione della rete, conservando comunque la struttura gerarchica dell’albero. Per quanto riguarda come specificatamente vengono convertire le strutture dell’albero, sono visivamente spiegate nella figura sottostante: [7]

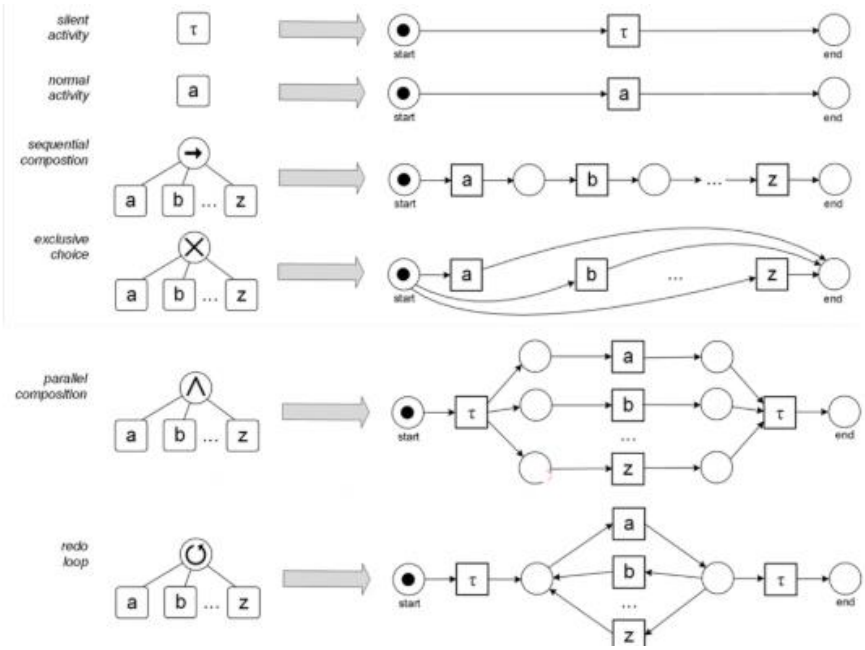

Figura 1.VII: Dal Process Tree alla Petri net.

Quindi, le *silent activities*, che nel process tree servono a organizzare la struttura dell’albero, ma senza corrispondere a delle reali attività, appaiono come transizioni senza label nella Petri net: non lasciano perciò traccia nell’event log, a differenza delle attività normali che corrispondono ad una semplice transizione.

[7] Slides: Wil van der Aalst (www.vdaalst.com)

La composizione sequenziale corrisponde ad una serie di attività appunto sequenziali nella rete di Petri, che vengono quindi eseguite una dopo l'altra.
Per quanto riguarda la scelta esclusiva, anche nella Petri net corrisponde a scegliere di eseguire una delle attività figlie, così come per l'operatore parallelo.
Il loop, che nel *process tree* è rappresentato da un nodo con una freccia in loop a cui sono collegate diverse attività, nella rete di Petri si trasforma in una struttura che permette al token di tornare indietro in un *place* precedente, consentendo un'esecuzione ripetuta di certe attività.
Parlando delle proprietà, e in particolare i punti di forza, dell'inductive miner, quelle più importanti sono sicuramente la semplicità e la generalità dei modelli di processo generati dall'algoritmo, la *soundess* degli stessi modelli (che può riprodurre l'intero event log) e la forte scalabilità dell'algoritmo, ossia la capacità di gestire anche una grande quantità di dati senza perdere di efficacia. La proprietà più distintiva è però quella definita dalla capacità di gestire i processi infrequenti, andando così a ridurre il rumore.
Per quanto riguarda gli aspetti negativi, l'inductive miner, a differenza di altri algoritmi, non esclude dal modello le attività con una frequenza bassa: questo potrebbe portare a dei modelli meno interpretabili e utili per i nostri obiettivi.

1.3.2 Heuristic miner

Il secondo algoritmo implementato nel progetto sarà l'Heuristic miner che, come per l'inductive miner è un algoritmo di Discovery che permette di definire un modello di processo partendo dall'event log a disposizione. È particolarmente utilizzato in contesti in cui bisogna gestire rumore e casi infrequenti, dato che, a differenza di altri

algoritmi, l'Heuristic si concentra sull'identificazione e rappresentazione dei pattern più frequenti e significativi.

Anche l'Heuristic miner parte quindi dall'event log come input e da qui costruisce dei dependency graphs per specificare le relazioni tra le attività nel log, in cui ogni nodo rappresenta un'attività e gli archi tra questi la probabilità che un'attività sia successiva ad un'altra.

L'algoritmo calcola poi due misure principali, quali la frequenza delle attività e la dipendenza tra le stesse e successivamente sarà l'utente a poter scegliere quanto filtrare il modello che risulterà tramite il threshold che verrà scelto.

Una delle proprietà più importanti dell'Heuristic miner è data dalla capacità di quest'ultimo di gestire il rumore tramite il filtraggio che non considera i casi meno frequenti basato sul threshold, portando ad avere modelli più puliti e accurati e di conseguenza più semplici da interpretare. Tutto ciò rende quindi questo algoritmo molto efficiente da implementare in casistiche reali, dove rumore e casi infrequenti sono molto comuni.

Parlando invece degli aspetti negativi dell'algoritmo, uno di questi è relativo al fatto che i risultati sono fortemente influenzati dal parametro del threshold che l'utente setta per il filtraggio, e che è una scelta non banale dato che potrebbe portare a dei risultati o troppo semplificati non considerando casistiche che potrebbero essere molto utili per l'analisi, o troppo complessi, con la forte presenza di rumore nei modelli di processo generati.

Inoltre, a differenza dell'inductive miner, con l'Heuristic quando i dataset cominciano ad essere di grandi dimensioni si potrebbe incorrere in problemi di scalabilità, perdendo quindi di accuratezza ed efficacia.

1.3.3 Fuzzy miner

I processi nella vita reale sono relativi però a modelli meno strutturati di quanto si crede, e gli algoritmi tradizionali hanno sfortunatamente problemi nel gestire queste tipologie di modelli, che risultano essere per la maggior parte delle volte dei modelli cosiddetti "spaghetti-like". Il Fuzzy miner parte, comunque, dall'event log a disposizione, ma può essere impiegato proprio in questi casi data la possibilità di essere configurabile e consentire di ottenere diverse versioni semplificate dei processi, applicando diverse tecniche, come ad esempio rimuovere archi non importanti, il raggruppamento di nodi altamente correlati in un unico nodo e la rimozione di cluster di nodi isolati. Per fare ciò, una volta acquisito come input l'event log, calcola le metriche basate ad esempio sulla frequenza, per poi non tenere in considerazione le casistiche che hanno dei valori più bassi rispetto al threshold definito dall'utente. Viene poi costruito un grafo iniziale dove i nodi sono le attività/eventi e gli archi le correlazioni tra gli eventi, che verrà poi semplificato andando ad esempio a fondere più nodi che hanno un'alta correlazione in un singolo nodo, come detto precedentemente, per poi raggruppare gli eventi e le loro relazioni in cosiddetti fuzzy clusters.[8] Il modello di processo che poi otterremo sarà mostrato in modo tale da evidenziare gli elementi più importanti per garantire una più immediata e semplice interpretazione.

[8] Il Fuzzy clustering è una forma di clustering dove ogni data point può appartenere a più di un cluster.

Perciò il punto di forza principale del Fuzzy è proprio relativo al fatto che può facilmente gestire anche queste situazioni in cui la variabilità è molto alta, che risultano quindi più problematiche con gli altri algoritmi.

Inoltre, il Fuzzy miner permette di regolare dinamicamente il livello di dettaglio del modello di processo che otterremo dalla sua implementazione in base all'importanza degli eventi, che viene calcolata sulla base di alcune metriche come può essere ad esempio la frequenza di questi. Inoltre, è un algoritmo flessibile, che quindi può essere personalizzabile tramite la modifica dei vari parametri utilizzabili, permettendo all'utente di renderlo il più coerente possibile con gli obiettivi dell'analisi.

Anche in questo caso, però, vi può essere un problema relativo alla scalabilità dell'algoritmo quando trattiamo dataset di grandi dimensioni, oltre al fatto che questa capacità dell'algoritmo di semplificare i modelli di processo potrebbe facilmente sfociare in una over-semplificazione degli stessi.

1.4 CONFORMANCE CHECKING

Le tecniche di conformance checking costituiscono la seconda tipologia di tecniche di Process mining enunciate nei paragrafi precedenti, dopo le tecniche e gli algoritmi di Process discovery. In particolare, se nelle ultime si andava a definire il modello di processo partendo dagli event log, con il conformance checking si esegue uno step successivo, ossia si confronta il modello che abbiamo a disposizione con un modello standard o ideale definito precedentemente, anche detto modello di riferimento e che,

come si vedrà nel nostro caso, sarà il percorso predefinito dal calendario didattico riguardo gli esami del primo anno.

Confrontando questi modelli quindi il conformance checking permette di individuare se un modello di processo si adatta a quello ideale e nel caso in cui non sia così, permette di identificare dove sono le deviazioni in modo tale da poterle analizzare e capire conseguentemente se sono delle deviazioni che apportano risultati negativi ai risultati finali oppure se migliorano il modello di processo su cui è stato basato il confronto.

Per confrontare quindi questi due modelli vengono utilizzate diverse misure quali *fitness*, *precision*, *generalization*, *simplicity*. La *fitness* misura quanto il log che abbiamo si adatta al modello di riferimento, mentre la *precision* va a penalizzare quei modelli troppo generici e che quindi simulano un comportamento improbabile dato l'event log che abbiamo.

Proseguendo con la *generalization*, questa metrica ribilancia la precedente e si concentra sull'overfitting dei modelli: infatti un modello deve essere comunque capace di generalizzare e riprodurre il possibile comportamento futuro, invece di catturare semplicemente ogni traccia dell'event log.

Infine, la *simplicity* premia i modelli che riescono a spiegare i comportamenti presenti nel log senza essere troppo complessi.

1.5 DIRECTED GRAPHS

I Directed graphs sono una componente fondamentale nel process mining, dato che sono un altro modo per definire i processi che sono stati eseguiti e che verranno

utilizzati nel progetto sia per dei task di predizione che per task di clusterizzazione, come si vedrà nello specifico in questa tesi.

Questi grafi sono composti da nodi e grafi, in cui i nodi rappresentano le attività o gli eventi per ogni caso mentre gli archi sono delle frecce direzionate che collegano i nodi, indicando la sequenzialità nello specifico processo. Questa proprietà tipica degli archi è fondamentale per i grafi perché garantisce la direzionalità del processo, ossia se abbiamo un arco che va da A a B e uno da B a C vuol dire che nel processo è stata eseguita prima l'attività A, poi la B e di conseguenza la C.[9]

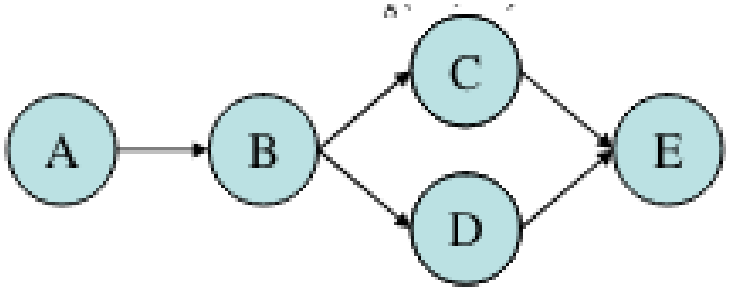

Figura 1.VIII: Esempio di un semplice Directed Graph.

1.6 TECNICHE DI MACHINE LEARNING UTILIZZATE

1.6.1 K-Means

Il K-Means è una tecnica di quantizzazione (o discretizzazione) vettoriale che ha dunque l'obiettivo di suddividere un set di n osservazioni che si hanno a disposizione in un numero finito k di cluster, definito a priori.

Ogni cluster è rappresentato dal rispettivo centroide, che consiste nella media dei valori assunti dalle *features* per ogni osservazione all'interno del cluster e ciascuna di

[9] Bae, J., Caverlee, J., & Zhang, Z. (2006). *Process Mining, Discovery, and Integration using Distance Measures*. Georgia Institute of Technology Repository.

queste osservazioni viene assegnata al cluster associato al centroide che si trova più vicino a questa osservazione.

Per questi motivi la tecnica di clusterizzazione del k-means è detta partitiva, perché segmenta le osservazioni in clustering, e *prototype-based*, perché i cluster sono rappresentati da un "prototipo", che per i dati continui è solitamente il centroide.

Dato che per avere un'ottima clusterizzazione le osservazioni all'interno dei vari cluster devono essere il più simili tra loro, matematicamente l'obiettivo è quello di minimizzare le distanze intra-clustering, ossia all'interno del singolo cluster, e massimizzare la distanza tra i punti appartenenti a cluster diversi.

Dunque, dato che il K-Means è basato sul prototipo centroide, l'obiettivo è minimizzare la somma delle distanze che vi sono tra quest'ultimo e tutti i punti appartenenti al medesimo cluster. Per fare ciò il K-Means segue diversi step relativamente semplici, ma prima di tutto bisogna definire il numero di K su cui si deve basare l'algoritmo. Il metodo solitamente più utilizzato, è il cosiddetto "*Elbow method*", ossia "metodo del gomito", utilizzato per identificare il momento in cui l'aggiunta di un ulteriore cluster non porta ad una significativa riduzione della varianza all'interno del cluster. Questa informazione è indicata dalla somma della distanza al quadrato tra ciascun membro del cluster e il rispettivo centroide, che viene quindi calcolata per ogni K all'interno di un range definito in precedenza che può andare da 1 a 10, 15, 20, etc…

Il risultato che si avrà in seguito all'applicazione di questo metodo sarà quindi un grafico simile al seguente:

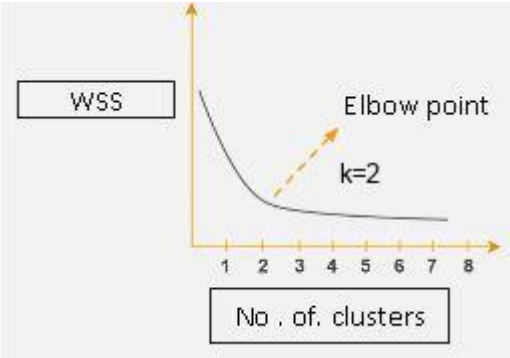

Figura 1.IX: Elbow method.

[10]

Vedendo la figura si può anche intuire perché si chiama “metodo del gomito”, ossia perché il numero ottimale per K viene scelto proprio come quello dove si crea una sorta di gomito nel grafico, ossia il punto dove l’aggiunta di nuovi cluster non comporta una significa riduzione della somma delle distanze al quadrato tra ciascuna osservazione del cluster e il rispettivo centroide.

Dopo aver scelto il numero di cluster considerato ottimale viene quindi implementato l’algoritmo del K-Means:

1) Inizializzazione: Vengono inizialmente selezionati casualmente K (definito in precedenza) osservazioni del dataset considerate come centroidi iniziali della clusterizzazione;
2) Assegnazione: Per ogni punto del dataset, viene calcolata la distanza che intercorre tra quest’ultimo e ciascuno dei K centroidi(solitamente sulla base della distanza Euclidea[11]), per poi assegnare il punto al cluster rappresentato dal centroide che presenta la distanza minima;
3) Aggiornamento dei centroidi: Una volta che sono stati assegnati tutti i dati ai vari cluster, i centroidi vengono ricalcolati come media di tutte le osservazioni assegnate agli specifici cluster;
4) Ripetere passaggi 2) e 3) fino al momento in cui o si raggiunge un determinato numero di iterazioni definite come massime, oppure la situazione rimane invariata tra due iterazioni consecutive;

[10] https://www.simplilearn.com/tutorials/machine-learning-tutorial/k-means-clustering-algorithm#:~:text=K%2DMeans%20clustering%20is%20an,objects%20belonging%20to%20another%20cluster.

[11] Esistono anche altre distanze come ad esempio la *Manhattan distance, Cosine distance, etc.*.

5) Risultato finale: Una volta che questa convergenza è stata raggiunta, l'algoritmo fornisce i centroidi finali dei cluster ed ogni osservazione è assegnata ad uno specifico cluster.

Una problematica relativa al K-Means è relativa al fatto che la scelta del numero di centroidi incide in modo particolarmente significativo sul risultato finale della clusterizzazione e, nonostante l'elbow method, non abbiamo la certezza che quella sia la soluzione finale ottimale per la segmentazione dei nostri dati a disposizione. Una possibile soluzione potrebbe essere effettuare diverse inizializzazioni e per ognuno di queste si sceglie quella che risulta essere ottimale in termini di somma delle distanze al quadrato, per poi continuare sulla base di quella inizializzazione. Questa potrebbe essere una soluzione efficiente, ma comunque non si avrà la certezza che sia quella ottimale.
Inoltre, il K-Means presenta problemi relativi alla dimensionalità, densità e forma non sferica dei cluster che deve trattare. Per questo si è andati ad implementare, per un'analisi più completa, una seconda tecnica di clusterizzazione, il DBSCAN.

1.6.2 Valutazione del clustering

Dopo aver definito le tecniche di clusterizzazione che verranno utilizzate, bisogna anche specificare il come queste tecniche sono state valutate nell'analisi, in modo tale da poter definire se la clusterizzazione è stata efficiente o meno.
Innanzitutto, esistono metriche di tipo supervisionato, quando vengono utilizzate anche informazioni esterne quali ad esempio le classi delle osservazioni, e metriche di tipo non supervisionato, ossia che non utilizzano informazioni esterne.
Partendo da quest'ultima categoria, si dividono in misure di coesione, che valutano la somiglianza tra osservazioni dello stesso cluster, e misure di separazione, che

valutano quando un cluster sia distinto o separato dagli altri cluster. In base a queste due misure la situazione migliore è quella in cui si ha un valore di coesione basso e un valore di separazione alto.

La più importante tra queste metriche *unsupervised* è la *silhouette*, che si basa su altre due misure che sono chiamate a(x) e b(x):

$$a(x) = \frac{1}{|C_i| - 1} \sum_{y \in C_i, y \neq x} proximity(x, y)$$

$$b(x) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{y \in C_k} proximity(x, y)$$

Figura 1.X: a(x) e b(x) per calcolare la silhouette.

Dove b(x) è detto cluster "vicino", dato che è quello che minimizza la misura di prossimità per tutti quei cluster a parte quello in cui il punto è effettivamente parte. Quindi la silhouette s(x) viene calcolata per ogni osservazione e in particolare:

- = 1- a(x)/b(x), se a(x) < b(x) ;
- = 0, se a(x) = b(x) ;
- = b(x)/a(x) – 1, se a(x) > b(x).

Perciò se s(x) ha un valore simile vicino a 1 vuol dire che il punto è clusterizzato bene, se è vicina a -1 ad essere il migliore per l'osservazione è il cluster cosiddetto "vicino", mentre se è intorno allo zero, l'osservazione è posizionata al bordo del cluster in cui è inizialmente inserita e il cluster più vicino. Infine, per calcolare quindi la bontà del modello di clustering, viene calcolata la media dei valori assunti dalla silhouette per tutte le osservazioni analizzate.

1.7 STRUMENTI

Ora verranno introdotti gli strumenti utilizzati durante l'analisi, partendo da Disco e finendo introducendo Subdue, passando per la libreria di python pm4py.

1.7.1 Disco

Disco è uno strumento ideato da Fluxicon, che grazie alla sua rivoluzionaria tecnologia di Process mining permette di creare rapidamente delle mappe visive dai dati che si hanno a disposizione.

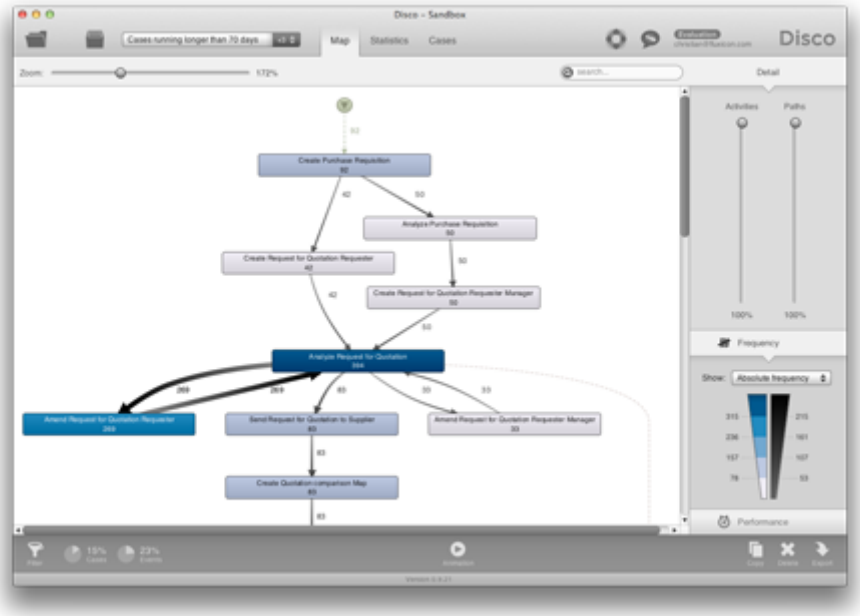

Figura 1.XI: Esempio di mappa visiva creata tramite Disco.[12]

Disco è uno strumento realizzato da esperti accademici con molti anni di esperienza nell'ambito del Process mining basandosi sull'idea di uno strumento efficace ed efficiente che si adatta alle esigenze di lavoro dei professionisti, oltre che contenere gli algoritmi di estrazione dei processi più veloci.

[12] https://fluxicon.com/disco/

La rivoluzionaria tecnologia di Process mining di Disco è in grado di creare automaticamente mappe di processo direttamente dai dati grezzi, tramite la scelta da parte dell'utente di diversi parametri quali ad esempio il livello di astrazione desiderato. Le mappe create con disco possono essere anche dinamiche, ossia si può visualizzare il processo così come viene svolto, in modo tale da poter individuare più velocemente i colli di bottiglia presenti nel nostro processo.

Disco permette poi di filtrare le visualizzazioni sulla base di un drill-down per performance dei casi, tempi, variazioni, attributi, relazioni tra eventi o endpoint, permettendo di ripulire i dati di processo e garantire un focus dell'analisi su modelli filtrati.

Lo strumento permette anche di visualizzare statistiche dettagliate riguardo le attività e i valori degli attributi relativi agli eventi, oltre che vedere tutti i casi presenti nell'event log con i rispettivi processi e le varianti, che consistono in tutte le possibili varianti del processo che si individuano all'interno dell'event log, con tanto di informazione relativa al numero di casi che sottostanno a quella variante.[13]

[13] Screenshot effettuato da me tramite utilizzo di Disco

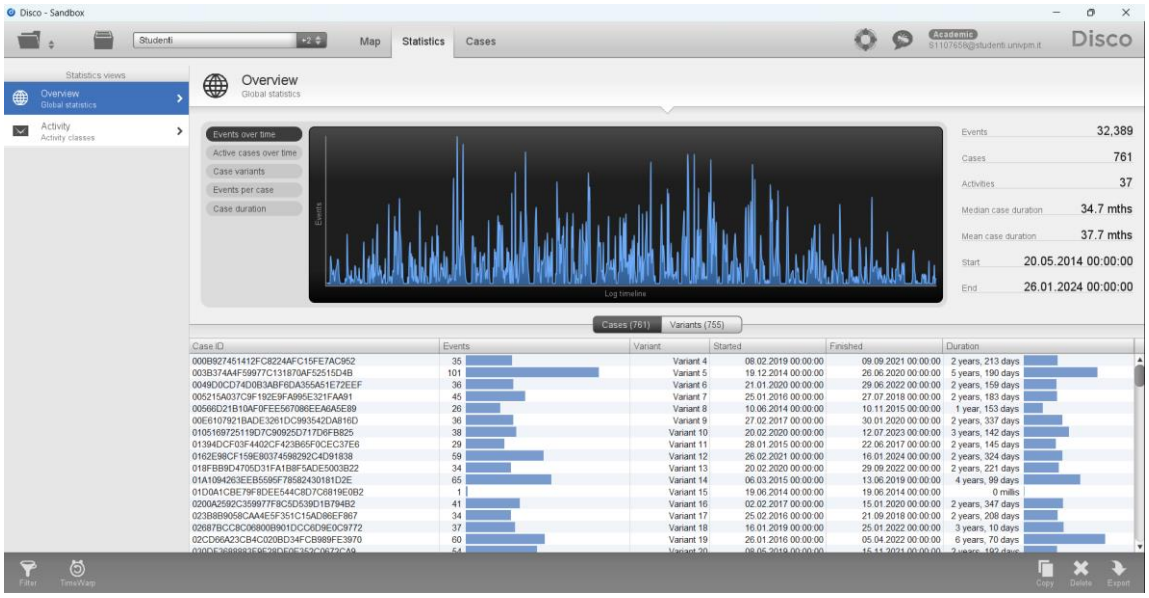

Figura 1.XII: Visualizzazione statistiche globali in Disco.

1.7.2 PM4PY

Pm4py è una libreria open-source di Python che fornisce una serie completa di strumenti per il Process mining, che consentono di prendere decisioni basate sui dati ottimizzando quindi i processi, ad esempio, delle aziende che lo utilizzano.

Infatti, per il Process discovery, include l'alpha miner, l'inductive miner, l'heuristics miner, che abbiamo già visto precedentemente oltre che l'ILP miner, o il correlation miner.

Per quanto riguarda invece il conformance checking, pm4py utilizza ad esempio la tecnica token-based replay, o gli alignments per comparare i modelli osservati con i modelli di processo ideali predefiniti.

Un'ulteriore funzionalità di pm4py che lo rende uno strumento fondamentale quando si tratta di Process mining è la sua capacità di funzionare con una serie di formati standard di event log e modelli di processo, che lo rendono adattabile a diversi

sistemi, difatti supporta sia il tradizionale formato per gli event log, ossia il formato XES (eXtensible Event Stream), che, per i registri di eventi incentrati sugli oggetti, il formato OCEL (Object-Centric Event Log). Per quanto riguarda invece i modelli di processo, supporta il formato delle Petri nets PNML (Petri Net Markup Language) e il formato dei process trees PTML (Process Tree Markup Language).

1.7.3 Subdue

Subdue è invece uno strumento che utilizzeremo nella parte finale dell'analisi e che permette di scoprire patterns e sottostrutture all'interno dei grafi che abbiamo a disposizione e che consistono nei processi dell'event log. E' infatti particolarmente utile per trovare i sottografi significativi, per poi implementare tecniche di machine learning per la predizione o clusterizzazione.

Subdue parte quindi dall'ottenere un grafo come input e da questo lo strumento individua *subgraphs* che potrebbero potenzialmente compressare il grafo originario basandosi sulla capacità di questi di ridurre la *description lenght* del grafo, ossia la quantità di informazioni necessaria per codificarlo. Quindi più un sottografo comprime il grafo originario più questo sarà considerato significativo: quindi questa valutazione si basa sul principio della *Minimum Description Lenght* (MDL), che mira a trovare la spiegazione più semplice (sottografo) che meglio descrive i dati.

Dopodiché Subdue perfeziona iterativamente i *subgraphs* candidati, cercandone sempre di migliori per migliorare la compressione fino a che non ne vengono più trovati altri di significativi e l'output finale sarà perciò un insieme di sottografi che risulteranno essere i migliori e che potranno poi essere analizzati e utilizzati in

tecniche di machine learning per ottenere informazioni ancora più significative dai dati che abbiamo a disposizione.

Capitolo 2: METODOLOGIE

In questo capitolo si effettuerà un focus su quelle che sono le metodologie che abbiamo utilizzato durante la nostra analisi, basandoci sui dati a disposizione oltre che sui concetti teorici precedentemente descritti, con una visione verso gli obiettivi finali dell'analisi.

Partiremo con il pre-processing dei dati, fondamentale per introdurre efficientemente l'analisi, per poi continuare spiegando le statistiche che abbiamo estratto per gli studenti sia in generale che con un focus sul primo anno, considerato quello più cruciale per l'efficiente andamento della carriera accademica dello studente.

Si passerà poi all'esposizione della metodologia per quanto riguarda la costruzione dei modelli di processo, come è stato applicato il conformance checking su di questi e sulla base di che modello di riferimento, per poi soffermarsi sulla costruzione dei grafi, sia i *directed* che l'estrazione dei *subgraphs*, concludendo con le tecniche di clusterizzazione implementate su quest'ultimi.

2.1 PRE-PROCESSING DEI DATI

Come detto nel capitolo introduttivo, i dati sono stati estratti dalla piattaforma online Esse3 e perciò è stato necessario eseguire il pre-processing dei dati per renderli il pulire e trasformare i dati adattandoli agli obiettivi prefissati della nostra analisi.

Questa fase introduttiva è stata svolta interamente in python e prima di tutto sono stati importati i tre file csv relativi a tre differenti tabelle: una prima tabella che riporta i dati relativi esclusivamente legati agli esami che sono stati passati con relativo voto e altre informazioni quali ad esempio il professore che ha tenuto il corso o la data di superamento dell'esame (in questa tabella per uno stesso studente si avrà quindi una sola osservazione per esame).

Nel secondo file csv, non sono presenti le informazioni specifiche dell'esame, come voto, data superamento, i crediti formativi, ma si trovano le informazioni riguardanti li percorso che è stato effettuato per arrivare a passarlo (quindi in questo caso potremmo avere più righe per lo stesso esame sostenuto da uno studente), mentre nell'ultima tabella si hanno informazioni relative all'anagrafica degli studenti.

Una volta importate tutte le tabelle necessarie sono state innanzitutto eliminate le osservazioni che presentavano duplicati in seguito all'estrazione, per poi andare a costituire il file che sarà la base della nostra analisi, effettuando la join tra tutte le tabelle, in modo tale da avere una tabella con tutte le informazioni relative a ogni studente necessarie per l'analisi.

Sono stati poi eliminati dal dataset coloro che presentavano un tempo di laurea sia troppo basso che troppo alto. I primi sono stati eliminati perché presentavano un tempo di laurea troppo basso (inferiore ai tre anni) e quindi impossibile: infatti tramite una ricerca nella tabella si poteva notare come avessero registrato alcuni esami anche un anno prima della sua iscrizione all'università; mentre i secondi non sono stati considerati perché non ritenuti utili all'analisi, dato che sono casi particolari in cui può esserci stata anche un'interruzione e ripresa del percorso

accademico (la selezione è stata effettuata sulla base di un threshold, 2400 giorni, ossia coloro hanno impiegato più di sei anni accademici per laurearsi).

E' importante sottolineare che la colonna utilizzata per la precedente selezione è la colonna denominata "tempo_laurea_AA", che conta i giorni che ha impiegato uno studente a laurearsi considerando l'inizio dell'anno accademico in cui si è iscritto (a prescindere quindi dall'effettiva data di iscrizione) e la data di laurea e che si distingue quindi dalla colonna chiamata "tempo_laurea", i cui valori in giorni sono la differenza tra la data effettiva di iscrizione all'università e la data di laurea.

DATA_SUP.1	DATA_ISCR	AA_ISCR_ID	tempo_laurea	tempo_laurea_AA
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0
2021-10-28 00:00:00+00:00	2018-10-01 00:00:00+00:00	2018.0	1123.0	1092.0

Figura 2.I: Screenshot del dataset che confronta le due colonne relative al tempo di laurea.

Come si può notare dalla soprastante, che raffigura l'esempio di uno studente, il nella colonna "tempo_laurea" il risultato in giorni è dato dalla differenza tra la data di laurea ("DATA_SUP.1") e la data in cui è precisamente avvenuta l'iscrizione all'università da parte dello specifico studente ("DATA_ISCR"), mentre la

successiva colonna “tempo_laurea_AA” è data dalla differenza tra la data di laurea che è la stessa e la data di inizio dell’anno accademico in cui si è iscritto lo studente, che in questo caso coincide con l’inizio di novembre, ed è per questo motivo che la il primo valore risulta maggiore del secondo, dato che lo studente si è iscritto molto presto, precisamente il primo ottobre.

Sulla base del tempo di laurea è stata definita poi una funzione che permette di dividere gli studenti in tre diverse categorie già citate nel capitolo introduttivo, ossia “*Early*”, composta dagli studenti laureatisi in tempo (entro tre anni e sei mesi), “*One Year Late*”, per coloro che si sono laureati un anno in ritardo (entro quattro anni e seie mesi) e infine “*Late*”, categoria composta da coloro che si sono laureati dopo quattro anni e sei mesi. Queste categorie verranno poi inserite all’interno di una nuova colonna chiamata “STATO_LAUREA”.

```
def determina_stato_laurea(tempo):
    if tempo < 1305:
        return 'Early'
    elif tempo >= 1305 and tempo < 1670:
        return 'One_year_late'
    else:
        return 'Late'
df_merged['STATO_LAUREA'] =
df_merged['tempo_laurea_AA'].apply(determina_stato_laurea)
df_esami['STATO_LAUREA'] =
df_esami['tempo_laurea_AA'].apply(determina_stato_laurea)
```

Per dare un'idea più precisa di quando gli esami sono stati sostenuti nel percorso dello studente, sono state poi create due nuove attività che però non sono veri e proprio esami, ma bensì le date che sono state definite come fine del primo semestre (due marzo dell'anno accademico) e fine del primo anno (due ottobre dell'anno accademico), valori che verranno quindi inseriti all'interno della colonna chiamata "DES" che fa riferimento alle attività, ossia agli esami.

Per il dataset "df_merged", ossia quello derivante dalle operazioni di join effettuate tra le varie tabelle, è stato effettuato un ulteriore step in quanto pre-processing. Difatti, dato che si ha la colonna "STA_REG_DES" che contiene cinque valori che sono "Prenotato", "Chiuso", "Verbalizzato", "Caricato" e "Annullato", si è prima di tutto analizzato a cosa corrispondessero questi valori e si è arrivati alla conclusione che lo stato è "Prenotato" significa che lo studente si è semplicemente prenotato all'esame mentre "Chiuso" si ha quando lo studente o si è ritirato/è stato respinto oppure era assente all'esame. Se lo studente si è invece presentato e successivamente consegnato l'esame si potrà avere nella colonna "STA_REG_DES" o il valore "Verbalizzato" o "Caricato", se l'esame risulterà rispettivamente insufficiente (votazione inferiore al 18) o passato. Infine, lo stato "Annullato" è ambiguo, perché un esame può essere annullato sia prima del caricamento del voto sia successivamente; perciò, questo valore corrisponde sia ad un esame passato che no, ma è comunque presente per poche osservazioni e quindi non significativamente rilevante per l'analisi.

Ciò che è stato fatto quindi per cercare di marginare questa possibile confusione tra i dati è stato compattare, all'interno di un'ulteriore colonna chiamata "DESC_FINALE", i valori degli stati, definendo quindi per ogni materia un valore

“Materia-Non Passato” se l’esame non è stato passato (sia perché assente/respinto/ritirato che perché insufficiente) e “Materia-Passato” se invece l’esame è stato passato. Per fare ciò è stata quindi innanzitutto definita la seguente funzione:

```
def determine_status(row):
    if row['DES'] in ['Fine primo semestre', 'Fine primo anno', 'Start', 'End']:
        return row['DES']
    if row['STA_REG_DES'] == 'Caricato':
        return 'Passato'
    elif row['STA_REG_DES'] == 'Chiuso' and row['TIPO_NOVERB_DES'] in ['Assente', 'Respinto/Ritirato']:
        return 'Non Passato'
    elif row['STA_REG_DES'] == 'Verbalizzato' and row['GIUD_ESA_DES'] == 'Insufficiente':
        return 'Non Passato'
    else:
        return None
```

che verrà applicata in una colonna chiamata “STATO_ESAME” all’interno della quale si avranno solamente valori “Passato” o “Non passato”, che verranno poi uniti alla materia all’interno della colonna “DES” per costituire “DESC_FINALE”.

DES	STATO_ESAME	DESC_FINALE
FISICA GENERALE I	Passato	FISICA GENERALE I-Passato
ANALISI MATEMATICA 1	Passato	ANALISI MATEMATICA 1-Passato
ALGEBRA LINEARE E GEOMETRIA	Passato	ALGEBRA LINEARE E GEOMETRIA-Passato
ECONOMIA DELL'IMPRESA	Non Passato	ECONOMIA DELL'IMPRESA-Non Passato
ECONOMIA DELL'IMPRESA	Passato	ECONOMIA DELL'IMPRESA-Passato
FISICA GENERALE II	Passato	FISICA GENERALE II-Passato
FONDAMENTI DI INFORMATICA	Non Passato	FONDAMENTI DI INFORMATICA-Non Passato
FONDAMENTI DI INFORMATICA	Passato	FONDAMENTI DI INFORMATICA-Passato
ALGEBRA LINEARE E GEOMETRIA	Passato	ALGEBRA LINEARE E GEOMETRIA-Passato
ANALISI MATEMATICA 1	Passato	ANALISI MATEMATICA 1-Passato
FISICA GENERALE I	Passato	FISICA GENERALE I-Passato
ECONOMIA DELL'IMPRESA	Passato	ECONOMIA DELL'IMPRESA-Passato
ANALISI MATEMATICA 2	Passato	ANALISI MATEMATICA 2-Passato
FONDAMENTI DI INFORMATICA	Passato	FONDAMENTI DI INFORMATICA-Passato
FISICA GENERALE II	Passato	FISICA GENERALE II-Passato
FISICA GENERALE I	Passato	FISICA GENERALE I-Passato
ANALISI MATEMATICA 1	Passato	ANALISI MATEMATICA 1-Passato
ALGEBRA LINEARE E GEOMETRIA	Non Passato	ALGEBRA LINEARE E GEOMETRIA-Non Passato
ALGEBRA LINEARE E GEOMETRIA	Non Passato	ALGEBRA LINEARE E GEOMETRIA-Non Passato
ALGEBRA LINEARE E GEOMETRIA	Passato	ALGEBRA LINEARE E GEOMETRIA-Passato
ANALISI MATEMATICA 2	Passato	ANALISI MATEMATICA 2-Passato
ALGEBRA LINEARE E GEOMETRIA	Passato	ALGEBRA LINEARE E GEOMETRIA-Passato
ANALISI MATEMATICA 1	Passato	ANALISI MATEMATICA 1-Passato
FISICA GENERALE I	Passato	FISICA GENERALE I-Passato
FONDAMENTI DI INFORMATICA	Passato	FONDAMENTI DI INFORMATICA-Passato
ALGEBRA LINEARE E GEOMETRIA	Non Passato	ALGEBRA LINEARE E GEOMETRIA-Non Passato

Figura 2.II: Screenshot di esempio costituzione "DESC_FINALE".

Infine, dato che come si è precedentemente detto, il primo anno è quello più cruciale ed è su questo che è stata basata l'analisi, si sono andati a filtrare i dataset considerando solamente gli esami dati da ogni studente nel primo anno di studi. Per fare ciò sono state prese in considerazione tre differenti date: "DATA_ISCR" che è la data in cui lo studente si è iscritto all'università, "DATA_SUP", ossia la data di superamento dell'esame e infine "DATA_FINE_PRIMO_ANNO", settata precedentemente come due ottobre dello specifico anno accademico e andando successivamente ad eseguire la differenza tra "DATA_SUP" e "DATA_ISCR" e tra "DATA_FINE_PRIMO_ANNO" e "DATA_ISCR". Le due differenze venivano poi confrontate e se la prima risultava superiore della seconda significava che l'esame era stato dato in un momento successivo al corso del primo anno dello studente e quindi non tenuto in considerazione nei nuovi dataset, che sono stati poi denominati

"df_esami_filtrato" e "df_merged_filtrato", da cui sono stati poi creati i tre subsets relativi alle etichette che definiscono le tre diverse tempistiche di laurea definite.

2.2 STATISTICHE

Prima di addentrarsi in quello che è il fulcro dell'analisi, ossia la costruzione dei modelli di processo, dei grafi e dei sottografi, seguiti dal loro utilizzo per l'implementazione delle tecniche di machine learning, si andranno a calcolare e mostrare, grazie all'utilizzo di PowerBI, alcune statistiche utili sia per gli stessi studenti che per l'università con l'obiettivo di capire eventuali problematiche dei percorsi accademici che stanno alla base del motivo per cui viene effettuata questa analisi. Verranno perciò rappresentate statistiche innanzitutto generali (che quindi si basano sul percorso accademico intero), per poi concentrarsi su quello che è considerato solitamente come l'anno cruciale, ossia il primo, e per finire con alcune statistiche definite tramite l'utilizzo del software Disco.

2.2.1 Statistiche generali

Come detto, partiremo con le statistiche generali, ossia quelle che danno informazioni riguardanti l'intero percorso accademico degli studenti che hanno conseguito la laurea, in modo tale da avere una visione innanzitutto generale dell'oggetto di studio.

Prima di tutto sono stati calcolati quanti sono gli studenti appartenenti ad ogni categoria (*Early*, *One year late* e *Late*) dato che proprio su questa distinzione si baserà l'analisi: per capire quali sono i comportamenti adottati dagli studenti delle varie categorie in modo tale da capire quali siano i comportamenti "vincenti" e quelli che invece potrebbero portare ad un rallentamento nel percorso.

La seconda informazione chi si è voluto mostrare è quella relativa ai crediti formativi conseguiti in media ogni anno dalla totalità degli studenti in un grafico per poi dettagliare in tre diversi grafici riguardanti ognuna delle tre categorie. Inoltre, è stata calcolata, per ogni studente, la percentuale di esami dati nell'anno in cui dovrebbero essere effettivamente dati. Per fare ciò è stata innanzitutto creata una nuova colonna denominata "ANNO_ATTESO" che mappa ogni esame con il rispettivo anno, basandosi sugli anni definiti dall'offerta didattica presente sul sito del corso preso in considerazione. Dopodiché, sulla base dell'anno d'iscrizione e la data in cui è stato superato l'esame sono stati calcolati e inseriti anche i valori relativi all'anno in cui l'esame è stato effettivamente passato nella colonna "AA_DATO" in modo tale da poter facilmente calcolare la percentuale media per ogni anno.
Per concludere le statistiche generali verrà poi mostrata, tramite box plot, la distribuzione delle votazioni degli studenti per le diverse categorie, per cercare di capire se ad un tempo di laurea ridotto corrispondono anche votazioni più alte e se quindi vi è un'ulteriore motivazione a seguire i comportamenti degli studenti laureatisi in tempo.

2.2.2 Statistiche primo anno

Dopo aver evidenziato le statistiche più generali, ci si addentrerà nel mostrare le informazioni riguardanti gli studenti presi in analisi per quanto riguarda quello che è considerato come anno cruciale nel percorso accademico degli stessi, ossia il primo anno.
Sono stati infatti calcolati prima di tutto due statistiche che rappresentano praticamente la stessa cosa ma in maniera differente, ossia gli esami dati da ogni

studente al primo anno e i relativi crediti formativi universitari ottenuti nello stesso anno dando questi esami, aggiungendo quindi un'ulteriore informazione, che è l'importanza degli esami che sono stati dati, considerando che alcuni esami del primo anno valgono 9 cfu (come ad esempio Analisi matematica I) e sono quindi tendenzialmente più impegnativi e altri invece 6 cfu (Algebra lineare e geometria per esempio).

Gli esami sostenuti sono stati poi suddivisi sulla base del semestre per avere gli esami dati sia nel primo semestre che nel secondo, per poi metterli a rapporto con quelli che sono gli esami attesi in entrambi i semestri, che rimangono fissi per ogni studente e sono rispettivamente tre per il primo semestre e quattro per il secondo.

Sempre riguardo gli esami, sfruttando il dataset dove sono presenti anche gli stati relativi agli stessi, sono stati calcolati per ogni studente le volte che hanno sostenuto un esame venendo bocciati. Questa informazione, per ogni gruppo di studenti (*Early*, *One Year Late*, *Late*) verrà poi visualizzata graficamente mettendo in evidenza per ogni materia del primo anno la media di volte in cui è stata provata prima di essere passata, per capire quali esami vengono provati di più solitamente e quali necessitano, in media, di una maggiore preparazione.

Infine, l'ultima statistica visualizzata per il primo anno è relativa alla media dei voti ottenuta dagli studenti, informazione utile che può anche essere messa a confronto con la media voti finale per capire se un buon andamento iniziale è solitamente sintomo di buoni voti anche alla fine del percorso accademico, e quindi di una buona

base di voto[14] per la laurea. Questa statistica verrà presentata mettendo in evidenza la distribuzione delle medie tra gli studenti compreso di media, mediana, i quartili e il valore massimo e minimo della distribuzione.

2.3 MODELLI DI PROCESSO

Si vuole quindi ora partire dall'event log pre-processato per estrarre, tramite gli algoritmi di discovery già citati, i modelli di processo, prima per tutti gli studenti in generale, poi per ogni categoria di studenti nello specifico, ossia "*Early*", "*One_Year_Late*" e "*Late*" in modo tale da poter confrontare i risultati e sulla base di questi definire un confronto più dettagliato tra i vari percorsi accademici.
La base per applicare questi algoritmi ed avere come risultato un modello di processo è avere un event log in formato. *xes* dove vengano specificate il Case_ID, l'attività (Activity) e un valore in formato data che permetta di definire la sequenzialità tra gli eventi (Timestamp). Per il primo parametro per tutti i dataset il valore sarà dato dall'identificatore dello studente (STU_ID). Per quanto riguarda invece attività e timestamp, nei dataset "df_esami" saranno dati dalla colonna "DES" e "DATA_SUP", mentre nei dataset "df_merged" saranno rispettivamente "DESC_FINALE" e "DATA_APP", ossia la data dell'appello, poiché anche le date in cui l'esame non è stato superato diventano rilevanti in questo caso.
E quindi in quest'ultimo caso, che è quello su cui si basa l'analisi, avremmo un event log dove ogni studente rappresenta un caso e gli esami (ossia le attività) dello

[14] Si parla di base di voto di laurea perché per sapere la votazione finale di uno studente è necessaria anche la votazione conseguita per la tesi, ma questa è un'informazione che non abbiamo a disposizione.

studente, sia quando sono stati passati che quando non lo sono, rappresentano gli eventi con le relative date nella colonna “Timestamp”.

Sono poi stati utilizzati sia l’inductive miner che l’heuristic miner per poter estrarre i vari modelli di processo e confrontarne i risultati, entrambi grazie alla libreria per Process mining pm4py messa a disposizione da Python.

Partendo dall’inductive miner è stato definito un loop che permette di individuare il threshold k migliore sulla base del quale costruire il modello di processo andando a definire modelli di processo per un threshold che va da 0 a 1 con salti di 0.1 (quindi il primo avrà un k di 0.1, il secondo di 0.2, etc…). Più il valore del threshold sarà basso più il rumore non verrà filtrato e quindi i modelli risulteranno più complessi (“spaghetti-like”) per i modelli costruiti con un threshold più basso per poi diventare sempre più filtrati e semplici da capire e analizzare all’aumentare del k, aumentando però anche il rischio che pattern importanti vengano omessi.

Per ogni modello di processo creato vengono anche definite le metriche per avere una valutazione più accurata degli stessi e di conseguenza scegliere quale tra questi è il migliore. In seguito, il codice utilizzato per estrarre i modelli di processo tramite l’utilizzo dell’inductive miner:

```
for k in np.arange(0,1.1,0.1): # loop per scoprire alberi di processo e convertirlo in rete di Petri, valutazione di fitness, precisione, generalizzazione e semplicità
    mod = pm4py.discovery.discover_process_tree_inductive(df_merged_caricato, k)
    net, initial_marking, final_marking = pm4py.convert_to_petri_net(mod)
    fitness = pm4py.fitness_token_based_replay(df_merged_caricato, net, initial_marking, final_marking)
    if __name__ == "__main__":
```

```
        prec = pm4py.precision_token_based_replay(df_merged_caricato, net,
initial_marking, final_marking)
        gen = generalization_evaluator.apply(df_merged_caricato, net, initial_marking,
final_marking)
        simp = simplicity_evaluator.apply(net)
    row = pd.DataFrame({'Threshold':k,'Fitness':fitness['average_trace_fitness'],
                        'Precision':prec, 'Generalization':gen, 'Semplicity':simp},index=[0])
    result0 = pd.concat([result0, row])
```

Viene costruito quindi un loop sulla base di k, ossia il threshold su cui viene definito il modello di processo, che assumo valori da 0 a 1, con degli step di 0.1, per cui verranno calcolate metriche per la valutazione del modello quali *fitness*, *precision*, *generalization* e *simplicity*. Sulla base di queste metriche è stato poi scelto il modello considerato migliore e questo è stato visualizzato graficamente tramite rete di Petri, grazie al seguente codice:

```
tree = pm4py.discovery.discover_process_tree_inductive(df_merged_caricato, k)
net, initial_marking, final_marking = pm4py.convert_to_petri_net(tree)
pm4py.view_petri_net(net, initial_marking, final_marking)
```

all'interno del quale bisogna sostituire al parametro k il valore che definisce il miglior modello di processo.

Questi step sono stati seguiti per ognuna delle categorie degli studenti, sostituendo a "df_merged_caricato" prima "df_merged_caricato_early", poi "df_merged_caricato_1_year_late" ed infine "df_merged_caricato_late".

Successivamente verrà utilizzato il secondo algoritmo di process discovery, l'heuristic miner per costruire altri modelli di processo basandosi sullo stesso event

log, con la differenza che per l'heuristic miner il threshold minimo verrà settato più alto, perché a differenza dell'inductive, con un threshold più basso, il modello non potrebbe essere considerato valido. Di seguito il codice per l'implementazione dell'heuristic:

```
result = pd.DataFrame(columns=['Threshold','Fitness','Precision', 'Generalization', 'Semplicity'])
for k in np.arange(0.8,1.1,0.02):
    mod = pm4py.discovery.discover_heuristics_net(df_merged_caricato, dependency_threshold=k)
    net, initial_marking, final_marking = pm4py.convert_to_petri_net(mod)
    fitness = pm4py.fitness_token_based_replay(df_merged_caricato, net, initial_marking, final_marking)
    if __name__ == "__main__":
        prec = pm4py.precision_token_based_replay(df_merged_caricato, net, initial_marking, final_marking)
        gen = generalization_evaluator.apply(df_merged_caricato, net, initial_marking, final_marking)
        simp = simplicity_evaluator.apply(net)
    row = pd.DataFrame({'Threshold':k,'Fitness':fitness['average_trace_fitness'],
                        'Precision':prec, 'Generalization':gen, 'Semplicity':simp},index=[0])
    result = pd.concat([result, row])
```

Per la visualizzazione, invece il codice è il seguente:

```
tree = pm4py.discovery.discover_ heuristics_net (df_merged_caricato, k)
net, initial_marking, final_marking = pm4py.convert_to_petri_net(tree)
```

pm4py.view_petri_net(net, initial_marking, final_marking)

Anche in questo caso, come per l'inductive, si andrà a implementare il codice per le tre differenti categorie di studenti e sostituendo a "k" il valore del k scelto come migliore per la costruzione del modello di processo.

2.4 CONFORMANCE CHECKING

Dopo aver costruito i modelli di processo per entrambi gli event log,si è andati ad applicare la tecnica di Process mining del conformance checking sui percorsi degli studenti.

Innanzitutto, è stato costruito manualmente il modello di processo standard su cui effettuare il conformance checking per i vari processi, ossia quelli costruiti sulla base dell'event log degli esami senza gli stati, soprattutto perché l'analisi nel conformance checking si basa più sulla sequenzialità di quando gli esami vengono dati più che i risultati (quindi tenere in considerazione anche gli esami quando non vengono passati), che in questo caso porterebbero solamente a una maggiore complessità, conseguente rumore e minore chiarezza riguardo l'aderenza del modello di processo che abbiamo, i quali renderebbero il conformance checking meno efficace.

Quindi, per creare il modello di processo è stato utilizzato un software gratuito e facilmente installabile come PIPE, che permette di creare Petri nets nel formato XML da un file PNML, che è il formato necessario per poterle utilizzare e analizzare in python.

La rete di Petri definita per il conformance checking è una rete di Petri che riprende il programma didattico definito dall'università come percorso ideale da seguire per

poter concludere al meglio almeno il primo anno, che, come detto in precedenza, è quello più importante per una efficiente carriera accademica e su cui perciò si baserà l'analisi.

Per l'implementazione del conformance checking sono state scritte le seguenti righe di codice:

```
net, initial_marking, final_marking = pnml_importer.apply(pnml_path)
gviz = pn_visualizer.apply(net, initial_marking, final_marking)
pn_visualizer.view(gviz)
replayed_traces =
pm4py.conformance_diagnostics_token_based_replay(df_esami_filtrato, net,
initial_marking, final_marking)
aligned_traces = pm4py.conformance_diagnostics_alignments(df_esami_filtrato, net,
initial_marking, final_marking).
```

Quindi dopo aver visualizzato la Petri net importata, vengono utilizzate due differenti tecniche di conformance checking: "token-based replay" e "alignments".

La prima è una tecnica di conformance checking che permette di verificare la conformance andando ad effettuare un "*replay*", ossia una ripetizione delle tracce dell'event log sulla petri Net standard.

Le metriche che si avranno come risultati dall'applicazione di questa tecnica saranno: *fitness*, che va da 0 a 1 e descrive quanto la traccia si adatta al modello; *tokens* mancanti, ossia che ci si aspettava fossero consumati ma non lo sono stati, che indica quindi eventuali deviazioni; *tokens* rimanenti che non sono stati consumati dalla traccia e poi i *tokens* prodotti e consumati durante la ripetizione della traccia.

Quella degli alignments è una tecnica di conformance checking più dettagliata che combina ogni possibile traccia nell'event log con la possibile esecuzione della Petri net più simile.

Le metriche risultanti da questa metodologia di conformance checking saranno invece costi dell'allineamento (minori costi corrispondono a migliori performance); movimenti del log, ossia gli eventi del log che non sono presenti nel percorso ideale definito (l'evento che determina la deviazione è presente nel log ma non nel modello standard); movimenti del modello, costruita sulla base degli eventi che sono invece presenti nel modello ideale, ma non nel log (l'evento che determina la deviazione è quindi previsto dal percorso ideale, ma non vi è riscontro nel log) e infine i movimenti sincroni, perciò gli eventi che corrispondono in entrambi.

I risultati derivanti dall'applicazione di queste tecniche di conformance checking danno risultati per ciascuno studente. Per questo motivo una volta ottenuti i risultati derivanti dalle applicazioni delle tecniche, questi vengono riassunti in dei grafici che permettono di avere una visione panoramica dei livelli di adattamento dei percorsi accademici degli studenti sulla rete di Petri standard. I risultati sono stati divisi sulla base delle tre categorie di studenti.

2.5 COSTRUZIONE GRAFI

In questo sottoparagrafo verrà invece introdotta la metodologia tramite la quale si sono inizialmente costruiti i grafi relativi i percorsi dei vari studenti, per poi, tramite lo strumento SUBDUE, estrarre i sottografi più significativi da utilizzare poi per le tecniche di machine learning relative alla clusterizzazione.

2.5.1 Directed Graphs

Il primo step consiste perciò nell'andare a costruire i Directed graphs per ogni studente, utilizzando la libreria concessa da Python NetworkX[15], che permette di creare, manipolare e analizzare la struttura, le dinamiche e le funzioni di reti complesse e grafi, oltre che la libreria Matplotlib per andare a mostrare visivamente il grafo.

Dato che la costruzione dei grafi si basa sul concetto di concorrenza[16] degli esami, la prima cosa che è stata fatta è stata definire, per ogni esame di ogni studente, quale fosse il periodo entro il quale due esami vengono considerati concorrenti.

Supponendo quindi che uno studente cominci a studiare per il determinato esame un mese prima, è stata definita per ogni evento del dataframe un'ulteriore colonna chiamata "DATA_START", ossia una data che corrisponde semplicemente alla sottrazione di un mese dalla data presente in "DATA_APP".

Grazie a ciò sarà possibile definire due esami come concorrenti nel momento in cui la data di inizio dello studio per un esame cada all'interno del periodo che va dall'inizio dello studio e la data dell'appello dell'altro.

Inoltre, per standardizzare l'inizio e la fine per ogni grafo, sono state definite due nuove attività "fittizie", ossia "Start" ed "End". Per fare ciò sono state quindi inizialmente individuate per ogni studente la prima e l'ultima attività (che è ovviamente sempre "fine primo anno") svolta nel primo anno e dopodiché è stato

[15] https://networkx.org/

[16] Nel caso del presente progetto una concorrenza tra due o più esami si ha nel momento in cui lo studente studia per i due o più esami parallelamente.

rispettivamente tolto e aggiunto un mese e un giorno in modo tale da evitare la concorrenza con le attività “reali”.

Per definire la concorrenza è stata definita la seguente funzione che implementa l’idea precedentemente introdotta:

```
def check_concurrency_merged(start1, end1, start2, end2, desc1, desc2):
  if desc1.split('-')[0] == desc2.split('-')[0]:
      return False
   return (start1 >= start2 and start1 <= end2) or (start2 >= start1 and start2 <= end1)
```

Si nota, nella seconda e terza riga di codice della funzione, come per la parte di analisi che si concentra sugli esami con gli stati è stato effettuato uno step ulteriore, che consiste nel non considerare mai la concorrenza se i due diversi eventi fanno riferimento allo stesso esame (*Esame*-Non Passato ed *Esame*_Passato), anche se la “DATA_START” di un evento cade all’interno del periodo di studio definito per l’altro, perché si fa comunque riferimento allo studio per lo stesso esame (vedremo poi un esempio reale nel capitolo riguardante i risultati derivanti dall’applicazione della metodologia nel progetto).

Questa funzione viene poi applicata all’interno di un’altra funzione, che è quella che va a definire nell’effettivo le coppie degli esami e quindi la costruzione del grafo, collegando tramite un arco solamente gli eventi tra di loro concorrenti (“Fine primo semestre”, “fine primo anno”, “Start” ed “End” non sono considerati all’interno della concorrenza):

```
def create_exam_pairs_merged(df_merged_caricato_early):
   student_data = {}
   for stu_id, group in df_merged_caricato_early.groupby('STU_ID'):
```

```
        group = group.sort_values('DATA_APP')

        exams = group['DESC_FINALE'].tolist()
        data_sup = group['DATA_APP'].apply(pd.to_datetime).tolist()
        study_starts = group['DATA_START'].apply(pd.to_datetime).tolist()

        graph = nx.DiGraph()

        for idx, exam in enumerate(exams):
            graph.add_node(idx, label=exam)

        non_restrictive_activities = ["Fine primo anno-Fine primo anno", "Fine primo
        semestre-Fine primo semestre", "Start-Start", "End-End"]

        for i in range(len(exams)):
            for j in range(i + 1, len(exams)):

                if exams[i] in non_restrictive_activities or exams[j] in
non_restrictive_activities:
                    graph.add_edge(i, j)
                elif not check_concurrency_merged(study_starts[i], data_sup[i],
study_starts[j], data_sup[j], exams[i], exams[j]):
                    graph.add_edge(i, j)
```

```
        reduced_graph = transitive_reduction(graph)



        label_dict = {node: graph.nodes[node]['label'] for node in graph.nodes()}
        nx.set_node_attributes(reduced_graph, label_dict, 'label')
        student_data[stu_id] = reduced_graph

    return student_data
```

Nello specifico, si nota nel codice, che, per rendere più veloce la costruzione del grafo, gli eventi vengono prima di tutto ordinati, tramite ID numerico, sulla base della data in cui sono avvenuti, per poi effettuare la verifica della concorrenza solamente con tutti gli eventi che avvengono dopo dello specifico evento.
Infine, una volta che è avvenuta la verifica della concorrenza, i nodi verranno rinominati con le etichette originali, dopo che, tramite la funzione di transitive reduction, permessa dalla libreria NetworkX, vengono eliminate tutte le coppie di nodi non significative in modo tale da semplificare il grafo finale.
Infine vengono definite altre due funzioni che hanno l'obiettivo di mostrare graficamente il grafo associato ad un determinato identificatore relativo allo specifico studente di cui vogliamo visualizzare il percorso (e nel caso in cui l'identificatore non esistesse, va a mostrare "No graph available for student ID {stu_id}") e di scrivere tutti i grafi che sono stati costruiti all'interno di un file di testo che verrà utilizzato per estrarre i sottografi tramite SUBDUE, il quale deve essere formattato in un determinato modo.

Nello specifico, ogni grafo deve essere preceduto dalla parola chiave XP; i nodi devono essere preceduti dalla parola chiave “v”; gli archi da “e”; mentre l’etichetta degli archi è così formattata: etichettaTarget_etichettaSource.

```
XP
v 1 CreateFine
v 2 SendFine
v 3 Notification
v 4 AppealToPrefecture
v 5 AddPenalty
v 6 SendAppeal
e 1 2  CreateFine__SendFine
e 2 3  SendFine__Notification
e 3 4  Notification__AppealToPrefecture
e 4 5  AppealToPrefecture__AddPenalty
e 5 6  AddPenalty__SendAppeal
```

Figura 2.III: Esempio di come deve essere formattato grafo all'interno del file di testo.

2.5.2 Subgraphs

Dopo aver costruito i Directed graphs relativi ai percorsi degli studenti, e successivamente averli riportati tutti all’interno di un file di testo formattato come si è visto, è il momento di estrarre i sottografi più significativi tramite l’applicazione di SUBDUE.

Quindi SUBDUE andrà a prendere in input i grafi che abbiamo costruito e come output darà i sottografi da poter utilizzare per portare avanti e concludere l’analisi.

2.6 CLUSTERIZZAZIONE

In questo sottocapitolo verrà introdotta la metodologia per la parte finale del nostro progetto, ossia l'applicazione di tecniche di *clustering* per arrivare ad ottenere informazioni importanti riguardo i pattern più significativi che dividono gli studenti tra coloro che si sono laureati in corso e chi no.

Prima di passare alla metodologia relativa al clustering, è necessario specificare che il dataset utilizzato in questo caso non sarà né "df_esami" né "df_merged", bensì un altro dataset chiamato "df_anagrafiche", che è stato costruito prendendo informazioni dal terzo file a disposizione inizialmente, come ad esempio il sesso dello studente e voto con cui si è diplomato alle superiori. Oltre a queste informazioni altre ne sono state calcolate, in modo tale da avere un dataset all'interno del quale ogni osservazione fa riferimento ad un singolo studente e le *features* costituiscono una serie di informazioni relative a quello studente come ad esempio l'anno di immatricolazione, la fitness calcolata tramite il conformance checking, gli esami dati e i crediti ottenuti sia al primo anno in totale, che divisi per i due semestri del primo anno, la base di laurea, la media voti, crediti ottenuti al secondo e terzo anno, ecc….

2.6.1 K-Means

Passando poi alla metodologia seguita per l'attuazione del K-Means nella nostra analisi, la prima cosa che è stata fatta è stato costruire due dizionari: "instance_graphs" e "subgraphs_graphs". Il primo è il dizionario costruito sulla base dei grafi relativi ai percorsi accademici degli studenti, e quindi ha come chiave l'ID dello studente (STU_ID) e come oggetto un oggetto DiGraph; mentre il secondo dizionario fa riferimento ai 100 sottografi: quindi, ciascuno di questi ha come chiave

il numero del sottografo e come oggetto un oggetto DiGraph che riporta la struttura dello specifico sottografo.

Sulla base di questi due dizionari si è effettuato un match per capire quali pattern comportamentali ogni studente presenta nel proprio percorso accademico tramite il seguente codice:

```
def identify_root_subgraphs(subgraphs):
    roots = set(subgraphs.keys())
    for key1, graph1 in subgraphs.items():
        for key2 in subgraphs.keys():
            if key1 != key2 and any(data.get('label') == key2 for _, data in
graph1.nodes(data=True)):
                roots.discard(key1)
                break
    return roots


# Funzione per verificare la presenza di un sottografo in un grafo
def is_subgraph_present(student_graph, subgraph):
    matcher = nx.algorithms.isomorphism.DiGraphMatcher(
        student_graph, subgraph,
        node_match=lambda n1, n2: n1 == n2
    )
    return matcher.subgraph_is_isomorphic()


root_subgraphs_keys = identify_root_subgraphs(subgraph_graphs)
```

```
root_subgraphs = {key: subgraph_graphs[key] for key in root_subgraphs_keys}


df_anagrafiche  = df_anagrafiche.dropna(subset=['STATO_LAUREA'])


# Aggiornare la tabella degli studenti
for subgraph_key in root_subgraphs:
    subgraph = root_subgraphs[subgraph_key]
    df_anagrafiche[subgraph_key] = [
        1 if is_subgraph_present(instance_graphs[student_id], subgraph) else 0
        for student_id in df_anagrafiche['STU_ID']
    ]
```

Come si può notare, dunque, per effettuare il match è stato effettuato tramite la definizione della funzione “is_subgraph_present”, che utilizza il “DiGraphMatcher” di “networkX” per eseguire un match di isomorfismo del sottografo, confrontando le etichette dei nodi del grafo dello studente con quelle dello specifico sottografo.
Il codice, quindi, itera sulla base della chiave di ciascun sottografo per verificare se è presente nel grafo dello studente, per poi andare ad aggiornare il dataset “df_anagrafiche”, aggiungendo una colonna per ogni chiave del sottografo. Se il sottografo è presente nel percorso dello studente allora quello specifico studente presenterà un valore pari a 1 nella colonna corrispondente a quel SUB, mentre se in caso contrario non lo presenta, il valore nella colonna sarà pari a 0.
Perciò, il dataset sarà aggiornato con 100 colonne binarie, una per ogni sottografo, che presenteranno, per ogni osservazione, valori 1 o 0.

Prima di ciò però, il dataset relativo alla totalità delle informazioni è stato filtrato eliminando tutti gli studenti che non erano presenti nel dataset “df_merged_caricato”, portando ad un dataset con 707 osservazioni e 139 variabili.

Di queste 139 variabili sono state eliminate quelle considerate non significative per l’analisi:

```
columns_to_remove = [
   'AA_IMM1', 'VOTO_MAX', 'ANNO_MATURITA', 'tempo_laurea',
'ES_ATTESI_S1', 'ES_ATTESI_1',
   'ES_ATTESI_S2', 'BASE_LAUREA',"ES_DATI_2","ES_DATI_3",
'ES_ATTESI_S1',
   'ES_ATTESI_S2', 'CREDS_Y2', 'CREDS_Y3', 'EXP_CREDS_Y1',
"ES_ATTESI_2", "ES_ATTESI_3",
   'EXP_CREDS_Y2', 'EXP_CREDS_Y3', "TIPO_TITOLO_COD",
"COD_SCUOLA", "SESSO","alligned_fitness1","CONSISTENCY_NORM",
   'tempo_laurea_AA', 'STU_ID', 'MATRICOLA',"ES_DATI_1",
]
```

costituendo così il dataset “data_updated”.

Per quelle rimanenti sono state effettuate diverse imputazioni per quanto riguarda i valori mancanti. Infatti per le variabili relative ai crediti ottenuti in totale al primo anno, primo semestre e secondo semestre, numero di esami non passati al primo anno ed esami passati al primo e secondo semestre i valori mancanti sono stati sostituiti ponendoli uguale a 0, mentre per le variabili relative al voto medio, la costanza degli intervalli temporali tra un esame e un altro, intervallo medio tra il sostenimento di un esame e un altro, la costanza nei voti ottenuti e il voto di maturità l’imputazione è

stata invece effettuata sulla base della media tra i valori delle osservazioni, grazie alla seguente riga di codice: imputer = SimpleImputer(strategy='mean').
Successivamente, è stato definito il dataset “X”, che semplicemente rispetto a “data_updated” non tiene in considerazione la colonna “STATO_LAUREA” riferita al gruppo a cui lo studente appartiene tra “*Early*”, “*One_year_late*” e “*Late*” e del quale le variabili continue sono state standardizzate tramite lo StandardScaler messo a disposizione dalla libreria sklearn messa a disposizione da Python.
In seguito, è stato scritto il seguente codice per scegliere il miglior k, ossia miglior numero di cluster, per eseguire il K-Means, dato che il numero di cluster deve essere definito a priori.

```
k_range = range(1, 10)
r_seed_range = range(1, 20)
sse = []
silhouette_scores = []
davies_bouldin_scores = []
best_i_for_k = []

# Calcolo SSE e Silhouette Score per ogni k
for k in k_range:
    min_sse = float('inf')
    best_i = None
    best_silhouette = None
    for i in r_seed_range:
        km = KMeans(n_clusters=k, random_state=i)
```

```
        labels = km.fit_predict(X)
        inertia = km.inertia_
        if inertia < min_sse:
            min_sse = inertia
            best_i = i
        if k > 1:  # Silhouette Score non è definito per k=1
            silhouette_avg = silhouette_score(X, labels)
            if best_silhouette is None or silhouette_avg > best_silhouette:
                best_silhouette = silhouette_avg
    sse.append(min_sse)
    best_i_for_k.append(best_i)
    if k > 1:
        silhouette_scores.append(best_silhouette)
        davies_bouldin_scores.append(davies_bouldin_score(X, labels))
    else:
        silhouette_scores.append(0)  # Placeholder per k=1
        davies_bouldin_scores.append(0)  # Placeholder per k=1
```

Sono stati dunque in prima battuta definiti gli intervalli per il k e per il seed da utilizzare per i cicli e inizializzate la SSE[17], valore della Silhouette, valore dell'indice Davies Bouldin e la lista relativa al miglior seed per ciascun k.

[17] Sum of Squared Errors, ossia la distanza Euclidea di ogni punto dal suo centroide più vicino.

Viene poi effettuato un ciclo per k che va da 1 a 9 all'interno del quale viene, per ogni k fatto partire un ulteriore loop per il seed da 1 a 19 e conseguentemente applicato il K-Means per prendere i valori relativi a sse e silhouette (solamente per k>1 dato che non può essere calcolata se vi è solamente un cluster), memorizzando come migliore il valore minimo per la prima e massimo per la seconda.
Si esce poi dal loop per calcolare, per ogni k il valore Davies Bouldin, che mette in relazioni la silhouette e la sse, ossia le misure inter-cluster e intra-cluster.
Questi risultati verranno poi visualizzati graficamente in modo tale da poter scegliere facilmente il numero di cluster sulla base del quale definire il K_Means.
Dopo aver scelto il numero ottimale di cluster i parametri di k e rispettivo seed migliore verranno impostati all'interno di un nuovo K-Means per poi assegnare ad ogni osservazione l'etichetta relativa al cluster in cui è contenuta.
Sono stati poi calcolati i centroidi e sulla base dei quali è stato costruito un dataset chiamato "centroids_df" che riporta quindi per ogni variabile i valori osservati per ciascun centroide. I valori delle variabili continue sono stati poi inversamente scalati in modo tale da avere delle informazioni più significative.
L'analisi dei cluster continua poi con il calcolo della dimensionalità di ogni cluster in modo tale da evidenziare in caso eventuali problemi di dimensionalità degli stessi e con la visualizzazione di istogrammi comparativi che confrontano i valori medi delle variabili continue tra i cluster per fornire un'idea di come queste differiscono tra i vari gruppi.
I cluster sono stati poi valutati sulla base di tre misure, quali silhouette, che definisce quanto i cluster sono distinti tra loro; varianza intra-cluster, che rappresenta

l'omogeneità tra le osservazioni all'interno di ciascun cluster e distanza tra centroidi rappresentati all'interno di una matrice di distanza.

Inoltre, sono state calcolate le matrici di confusione per rappresentare quanti elementi di una classe sono presenti all'interno di ogni cluster, per definire anche la rappresentatività del singolo cluster per ogni categoria di studenti, sia in termini di valori assoluti che percentuali.

Una volta individuata la qualità della segmentazione e i cluster che contengono gli studenti migliori e peggiori in termini di performance sulla base delle precedenti matrici di confusione, viene effettuata un'analisi specifica sulla base dei sottografi significativi nei vari cluster, in modo tale da poter individuare i pattern più comuni tra gli studenti più performanti e quelli più comuni per quelli meno performanti, in modo tale da ottenere informazioni fondamentali per Università e studenti in un'ottica di miglioramento generale dell'esperienza accademica dello studente.

Quest'analisi consiste nel prendere i valori assunti da ogni centroide per ogni singolo sottografo e sommarli tra di loro, per poi dividere il valore del singolo centroide per la somma, in modo tale da avere un valore percentuale che definisce quanto un sottografo è rappresentato tra i vari cluster, ossia la capacità di questo di distinguere tra i vari cluster.

I valori delle percentuali verranno riportate all'interno del dataset chiamato "percentage_df" che è stato poi filtrato tenendo in considerazione solamente i sottografi che hanno una rilevanza statistica, cercando di evitare di avere come SUB significative SUB che hanno una percentuale del 100% ma basata su una sola osservazione. Per fare ciò è stato innanzitutto effettuato, per ogni sottografo, il

conteggio[18] delle osservazioni che lo presentano e successivamente, posta una soglia pari a 5 osservazioni, sono stati considerati in un dataset chiamato "percentage_df_filtrato", le percentuali di solamente quei sottografi che superavano il valore soglia pari a 5 (5 non incluso).

Questo dataset filtrato verrà poi utilizzato per costruire, per ogni gruppo, un grafico a barre orizzontali ordinato in modo decrescente che rappresenta i primi dieci sottografi in ordine di percentuale, consentendo quindi di capire rapidamente quali siano quelli che caratterizzano di più ogni cluster.

Per concludere l'applicazione dell'algoritmo K-Means per il clustering, sono stati costruiti i modelli di processo per ciascun cluster, con l'obiettivo di confrontarli con i modelli precedenti, che presentavano una marcata variabilità. Tale confronto è finalizzato a verificare se, tramite questo approccio, si possa ottenere una riduzione della significativa variabilità osservata in precedenza.

[18] Dato che le SUB sono variabili binarie che presentano 1 o 0, il conteggio è stato effettuato grazie a una somma per ogni colonna riferita a sottografi.

Capitolo 3: STATISTICHE

In questo capitolo verranno presentati i risultati delle statistiche definite riguardo i percorsi accademici degli studenti, fornendo quindi le motivazioni alla base della nostra analisi.

Si partirà prima di tutto con il mostrare graficamente le statistiche più generali, per poi concentrarsi su quelle del primo anno e infine quelle messe a disposizione da Disco, con particolare dettaglio al dataset in cui anche gli stati relativi a ciascun esame sono tenuti in considerazione.

3.1 Risultati statistiche descrittive generali

Si parte quindi, come detto precedentemente, con la presentazione delle statistiche generali, ossia quelle che riguardano i percorsi accademici nella loro completezza dei vari studenti a disposizione.

Prima di tutto, per avere una panoramica della situazione degli studenti dell'università presa in analisi mostriamo la suddivisione percentuale degli studenti tra le tre categorie in cui sono stati divisi sulla base del tempo di laurea distribuiti sui vari anni accademici:

Anno imm.ne	Early	One_year_late	Late
2010	0.00%	47.83%	52.17%
2011	37.14%	31.43%	31.43%
2012	35.71%	26.79%	37.50%
2013	38.36%	38.36%	23.29%
2014	47.54%	16.39%	36.07%
2015	40.00%	38.82%	21.18%
2016	42.86%	30.95%	26.19%
2017	50.60%	30.12%	19.28%
2018	64.21%	23.16%	12.63%
2019	69.23%	30.77%	0.00%
2020	100.00%	0.00%	0.00%

Tabella 3.I: Percentuale studenti per categoria per anno accademico di iscrizione.

Come si può notare, la percentuale degli studenti che si laureano in tempo è quasi per ogni anno la superiore tra le tre categorie, ad eccezione degli studenti iscritti nel 2010, 2012 e 2013. Difatti, nel primo anno d'iscrizione abbiamo poche osservazioni (23) e tutte queste risultano essersi laureate dopo il periodo di tre anni e mezzo. Nel 2012 e 2013 invece la situazione è più bilanciata, con la percentuale degli studenti che si sono laureati in tempo che è rispettivamente più bassa degli studenti "*Late*" e della stessa percentuale degli studenti sono laureati entro l'anno di ritardo previsto. Per quanto riguarda gli altri anni l'unico dato particolare che troviamo è l'assenza di studenti che si sono laureati in ritardo per l'anno di iscrizione 2020, ma questo è normale perché è dovuto al fatto che gli studenti iscritti nel 2020 che non si sono laureati in corso devono ancora laurearsi; perciò, non abbiamo informazioni riguardo al loro percorso accademico. Questo spiega anche perché per l'anno 2019 non si hanno studenti che si sono laureati dopo quattro anni e mezzo, che porta ad avere una percentuale diversa da zero solo per le categorie "*Early*" e "*One_year_late*".

Dopo aver presentato quindi la distribuzione degli studenti tra le varie categorie per anno accademico, si passa a mostrare quanti crediti in media vengono dati per anno

andando ad evidenziare l'andamento nel dare gli esami tra i vari anni accademici, tenendo in considerazione anche il peso degli esami che vengono dati.

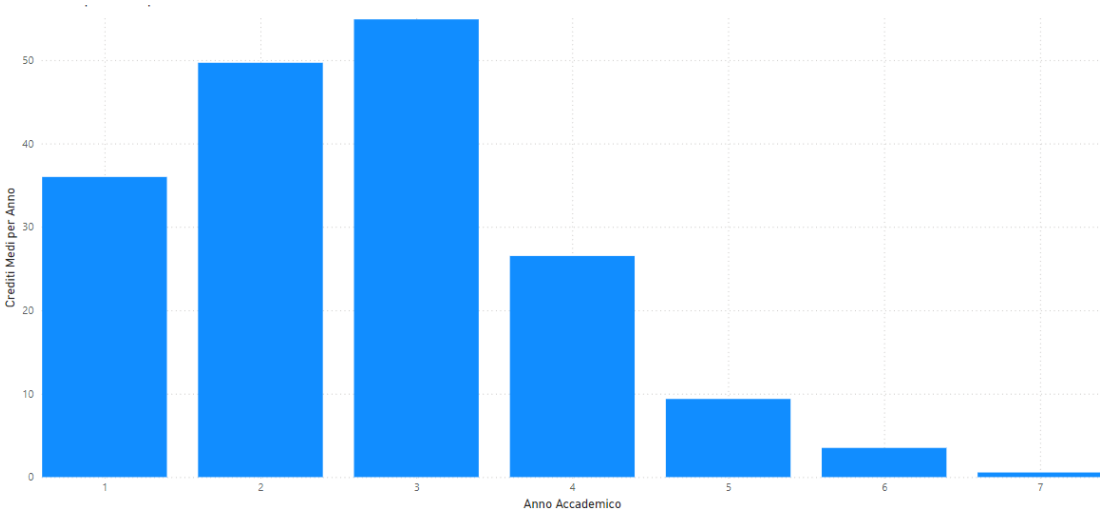

Figura 3.I: Distribuzione media crediti per anno accademico.

Nella figura soprastante vengono mostrati i valori relativi alla media di crediti dati in ogni anno accademico tenendo in considerazione tutti gli studenti, difatti si arriva fino al settimo anno accademico, dove si trova un valore molto basso, dovuto dal fatto che gli studenti che hanno impiegato sette anni per conseguire il titolo di laurea sono molto pochi, così come quelli che ne hanno impiegati sei.

Concentrandosi invece sugli altri anni, che garantiscono una base statistica di osservazioni più solida, si può notare come gli anni dove gli studenti sembrano impegnarsi di più sono il secondo e il terzo, in cui in media vengono dati intorno ai 50 crediti formativi in entrambi. Questo probabilmente dovuto al fatto che hanno più "libertà" nel dare gli esami, potendo dare anche quelli che non hanno sostenuto nel primo anno, dove invece gli studenti sono più limitati, dato che possono sostenere solamente i sette esami del primo anno, ottenendo al massimo 54 crediti.

Per avere una spiegazione più chiara di questa distribuzione è stata realizzata un'analisi specifica per ogni categoria di studenti. I risultati sono riportati nella seguente tabella:

Categoria	1°	2°	3°	4°	5°	6°	7°
Early	45.14	59.28	65.45	13.05	-	-	-
One_year_late	30.03	44.11	48.91	48.57	13.67	-	-
Late	21.76	33.38	36.75	36.70	33.54	21.69	9.88

Tabella 3.II: Crediti medi ottenuti per anno per categoria studenti.

La prima cosa che si può notare sono ovviamente i valori mancanti nelle categorie di studenti che non fanno parte di quelli laureatisi più in ritardo.

Andando più nello specifico nell'analisi dei crediti formativi ottenuti anno per anno, la tabella mostra come gli anni in cui gli studenti sembrano impegnarsi di più sono proprio quello centrali, come testimoniato anche dal grafico precedente della Figura 3.I. Inoltre, l'anno in cui gli studenti per ottengono in media più crediti è il terzo per tutte e tre le categorie con una differenza di addirittura 30 crediti tra gli studenti che si laureano in corso e quelli che si laureano dopo i quattro anni e mezzo.

Ciò che non viene evidenziato subito ma che ci dà un'informazione molto importante è la differenza percentuale tra i crediti ottenuti nel primo anno tra le varie categorie. Difatti si può vedere come la differenza percentuale tra i crediti dati nel primo anno tra gli studenti laureati in corso e non è del 50.32% con coloro che si sono laureati entro quattro anni e mezzo, ma soprattutto del 107.4% con gli studenti "*Late*", quindi più del doppio.

Guardando la stessa informazione, ma per il terzo anno, che, come abbiamo visto, è quello dove gli studenti danno più esami in media, le differenze percentuali sono diverse: rispettivamente 33.82% e del 78.10%, a testimonianza di come ad essere

importante è soprattutto il primo anno, ossia in che modo viene iniziato il percorso accademico.

A conferma di ciò sono stati anche costruiti tre istogrammi, uno per ogni anno, che rappresentano il conteggio degli studenti per ogni fascia percentuale di esami dati, dove la percentuale è il numero di esami dati su numero di esami previsti per quell'anno:

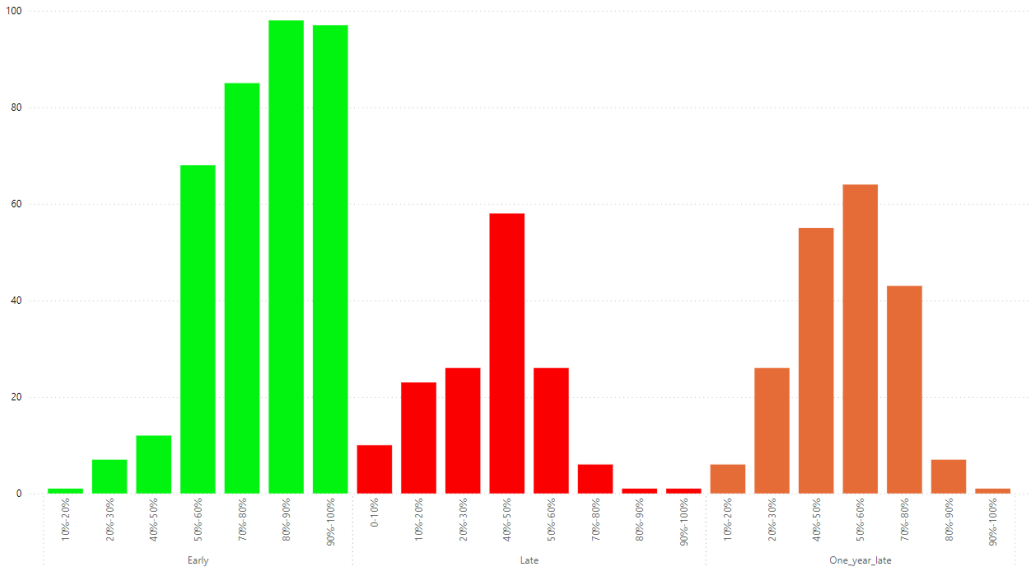

Figura 3.II:Istogramma fasce percentuali per primo anno.

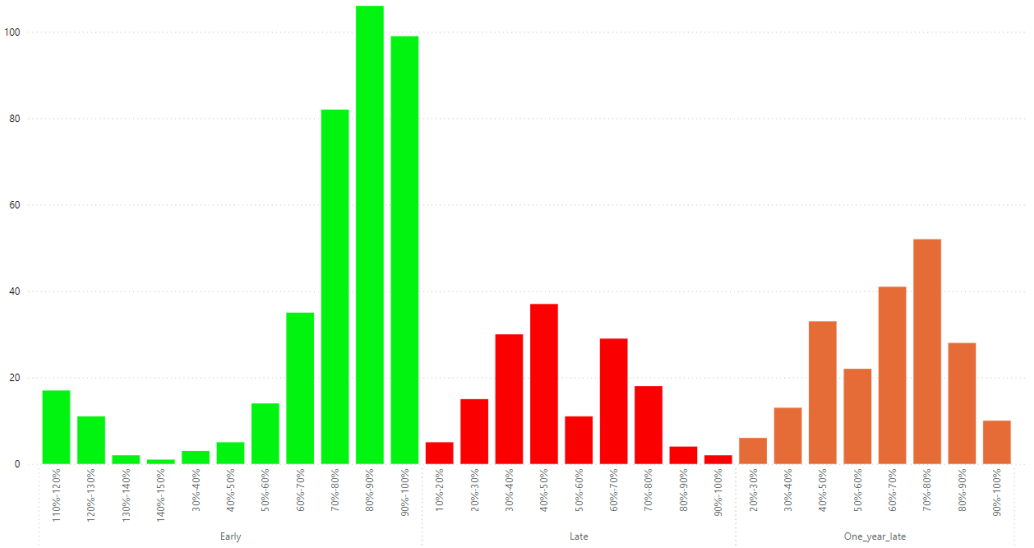

Figura 3.III: Istogramma fasce percentuali per secondo anno.

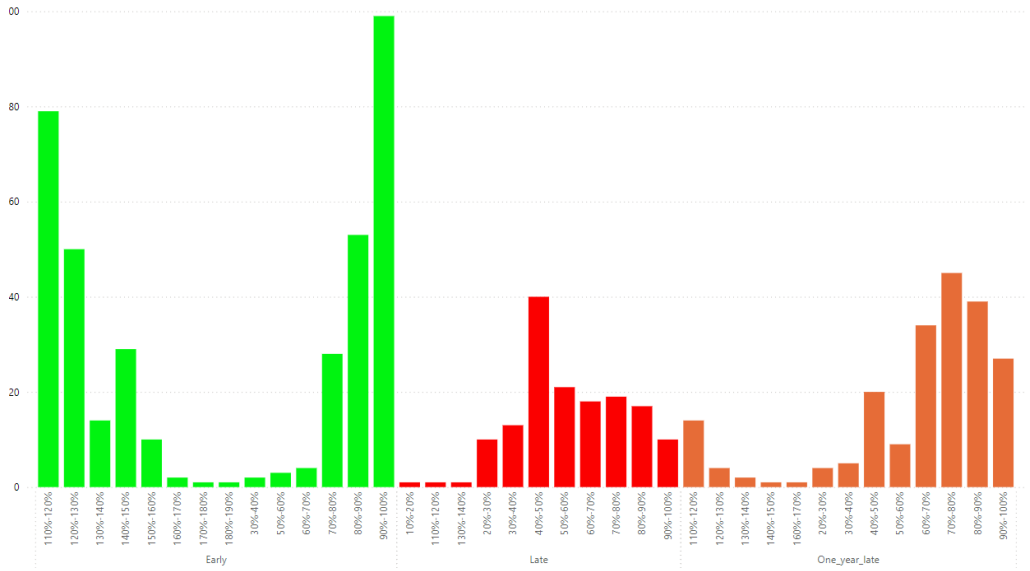

Figura 3.IV: Istogramma fasce percentuali per terzo anno.

Ciò su cui è importante concentrarsi in questi grafici non è solamente l'altezza della colonna, che rappresenta il numero di studenti appartenenti a quella fascia in numero assoluto, bensì il rapporto tra le colonne di ciascuna delle tre sezioni in cui sono divisi i grafici, che, come si può notare, rappresentano le categorie in cui sono stati divisi gli studenti.

Infatti, partendo dalla Figura III.2, che mostra la situazione per il primo anno, si vede come la distribuzione protenda in modo estremamente significativo verso le percentuali più alte, con solamente una minima parte degli studenti che si sono laureati in corso che danno meno del 50% degli esami al primo anno. La situazione cambia invece per le altre due categorie: la distribuzione è più o meno simile tra di loro, dove gli studenti "*Late*" presentano una situazione leggermente peggiore, infatti la fascia 0-10% è presente solamente per questi studenti, ed è addirittura superiore agli studenti che hanno dato più del 70% degli esami al primo anno.

Anche per quanto riguarda il secondo e terzo anno accademici, i cui istogrammi vengono riportati rispettivamente nella figura III.3 e III.4, la rappresentazione vede

una distribuzione nettamente migliore per gli studenti "*Early*", che toccano addirittura percentuali di esami dati su esami attesi per quell'anno anche del 170%/180%. Mentre per quanto riguarda gli studenti appartenenti a "*One_year_late*" e "*Late*" le distribuzioni sono simili nel secondo anno, il quale non risulta quindi essere significativo nel distinguere il percorso accademico tra categorie, mentre nel terzo si nota di più una differenza nel conteggio delle varie fasce percentuali.

3.2 Risultati statistiche con focus su primo anno (esami con stati)

Come è già stato evidenziato nelle sezioni precedenti, l'anno cruciale nel percorso accademico di uno studente è il primo, ossia, nella maggior parte dei casi risulta decisivo iniziare al meglio la carriera universitaria per riuscire poi a conseguire il titolo in tempo. Proprio per questo motivo, perciò, dopo aver mostrato le statistiche descrittive generali riferite all'intero percorso accademico degli studenti, verrà effettuato un focus dell'analisi proprio su questo primo anno.

Si comincerà innanzitutto con il mostrare gli esami dati nel primo e poi nel secondo semestre del primo anno, mettendoli a confronto con quelli che sono gli esami attesi per ciascuno dei due semestri:

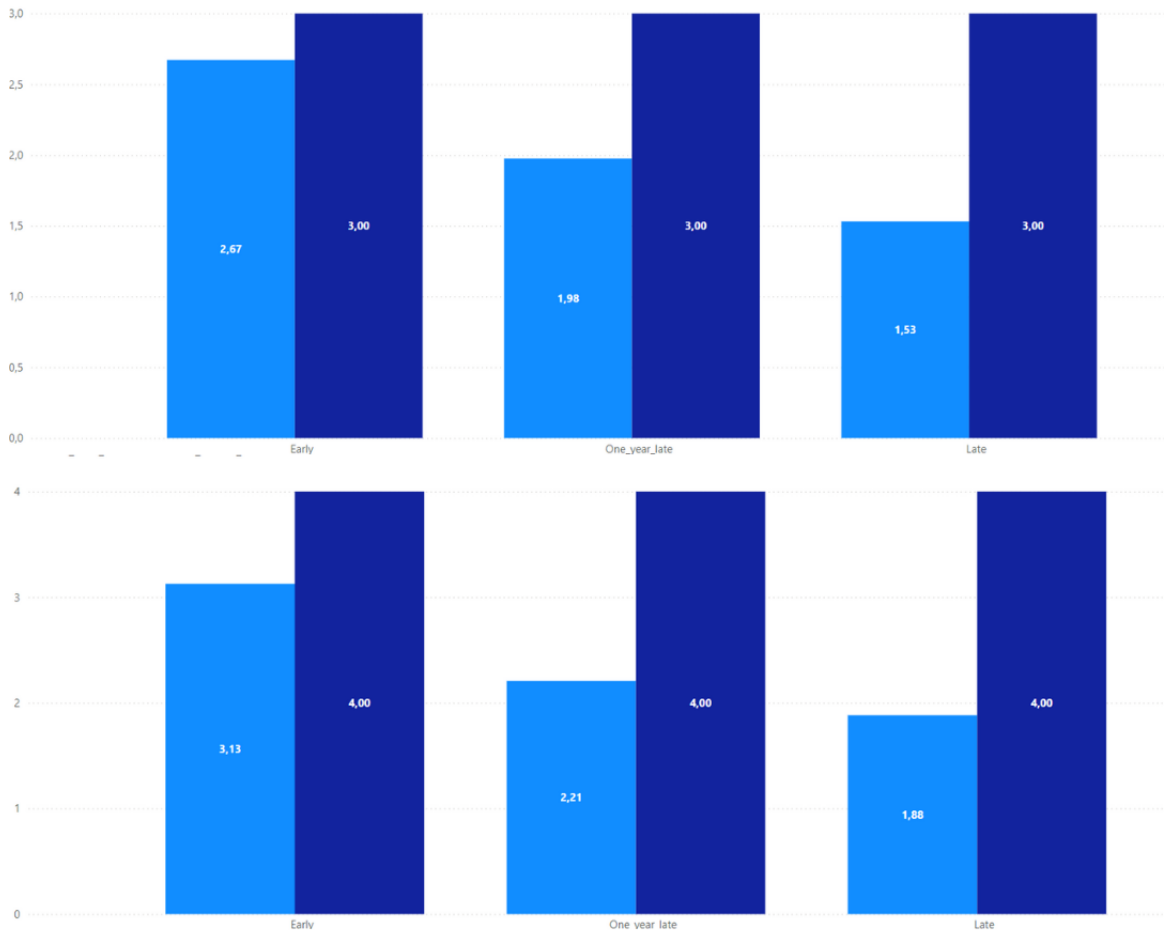

Figura 3.V: Rapporto tra esami dati ed esami attesi per entrambi i semestri del primo anno.

Prima di tutto bisogna sottolineare il fatto che tra questi due grafici, dove quello superiore rappresenta il primo semestre e quello inferiore il secondo, quello che è più limitato in termini di esami dati è il primo, perché non posso dare un esame del secondo semestre al primo semestre, mentre il contrario è possibile.

Dopo questa doverosa precisazione ci si può concentrare sui risultati, i quali evidenziano che gli studenti "*Early*" hanno una media di esami dati del 2.67 su 3 attesi: ciò vuol dire che quasi tutti gli studenti appartenenti a questa categoria hanno passato tutti e tre gli esami previsti per il primo semestre. La media degli esami dati

poi va diminuendo passando dagli studenti “One_year_late” (1.98) fino agli ultimi, ossia gli studenti gravemente in ritardo con una media pari circa alla metà degli esami attesi, 1.53.

Passando ora all'analisi dei dati relativi al secondo semestre, il grafico evidenzia che, anche dopo il primo semestre, gli studenti appartenenti al gruppo "Early" presentano una media di esami sostenuti significativamente elevata, pari a 3.13. Questo valore risulta nettamente superiore rispetto ai 2.21 degli studenti "One_year_late" e agli 1.88 degli studenti "Late". Tale dato assume una rilevanza ancora maggiore se si considera che la quasi totalità degli studenti "Early" ha sostenuto tutti e tre gli esami del primo semestre: ciò implica che, per la maggior parte degli studenti di questo gruppo, gli esami considerati nella seconda media sono con alta probabilità relativi al secondo semestre. Al contrario, per gli altri gruppi, è possibile che la media includa anche esami non sostenuti nel primo semestre.

Dato che si hanno a disposizione anche i dati riguardanti gli stati relativi ad uno specifico esame che definiscono se quest’ultimo è stato passato o meno, ci si concentrerà ora proprio su quelli che non sono stati passati, cercando di definire quelli che sono gli esami più provati e che di conseguenza necessitano, in teoria, di uno studio più approfondito. Anche in questo caso l’analisi sarà distinta tra le tre categorie di studenti per cercare di scovare analogie e differenze tra i percorsi accademici, in modo da mettere in risalto eventuali output che potrebbero risultare utili.

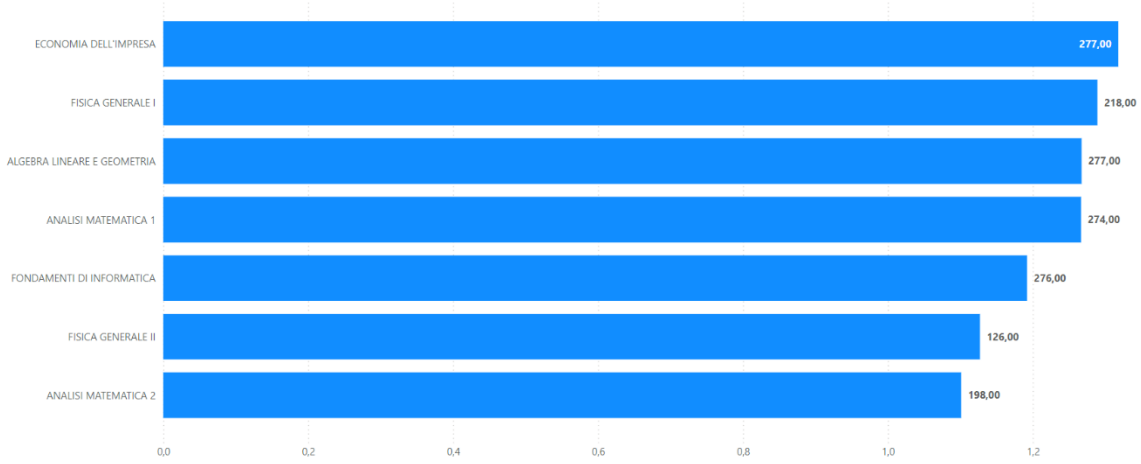

Figura 3.VI: Media di volte che un esame è stato provato prima di essere passato (Studenti "Early").

Partendo dagli studenti appartenenti al gruppo "*Early*" viene presentato il grafico a barre orizzontali che presenta la media tra gli studenti delle volte in cui uno studente ha provato uno specifico esame prima di passarlo che ne determina quindi la lunghezza, mentre i numeri alla fine della barra rappresenta il numero di studenti che quell'esame l'ha passato e quindi la base su cui è calcolata la media. Quindi l'esame più provato dagli studenti del gruppo "*Early*" si può notare sia "Economia dell'impresa", con una media di poco superiore all'1.2 su una base d 277 studenti, seguito da "Fisica Generale I" e "Algebra Lineare e Geometria" e "Analisi Matematica 1" a pari media. Il meno provato invece è "Fisica Generale II" (da 126 studenti), che potrebbe rappresentare quindi un esame a cui prestare particolare attenzione in fase di studio, nonostante come media di volte in cui è stato sostenuto sia inferiore ad altri.

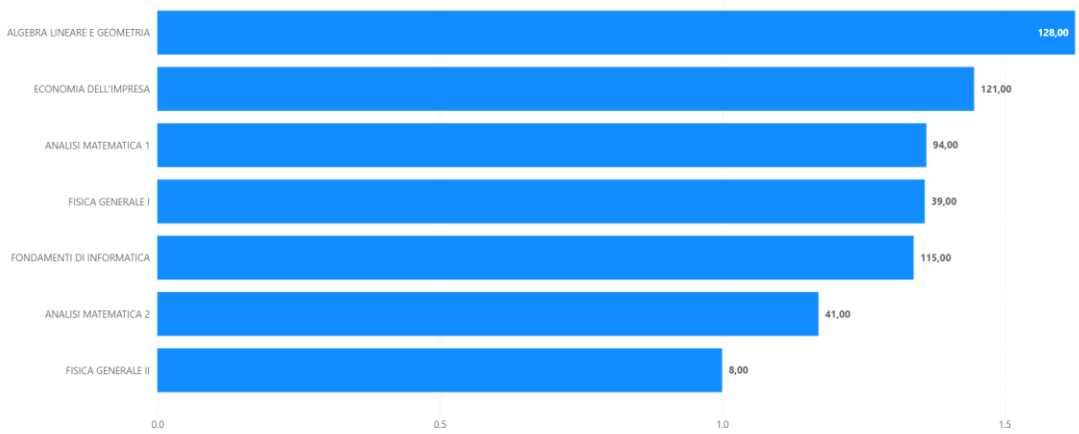

Figura 3.VII: Media di volte che un esame è stato provato prima di essere passato (Studenti "One_Year_Late").

Passando agli studenti appartenenti alla categoria “One_Year_Late” si può notare immediatamente un calo di almeno più della metà di studenti che l’hanno passato per tutti gli esami, fino ad arrivare all’esame “Fisica Generale II” che è stato passato da soli 8 studenti su 205 del gruppo.

Spostando l’attenzione invece agli esami più provati prima di essere passati, i primi sono sempre gli stessi esami ma con un differente ordine, difatti al primo posto non vi è più “Economia dell’Impresa”, che ora è al secondo, bensì “Algebra Lineare e Geometria” con una media di 1.67.

Per ritrovare una media simile a quella dell’esame più provato per gli studenti “Early” bisogna arrivare fino al quindi posto di quest’ultimo grafico, “Fondamenti di Informatica” con 1.34.

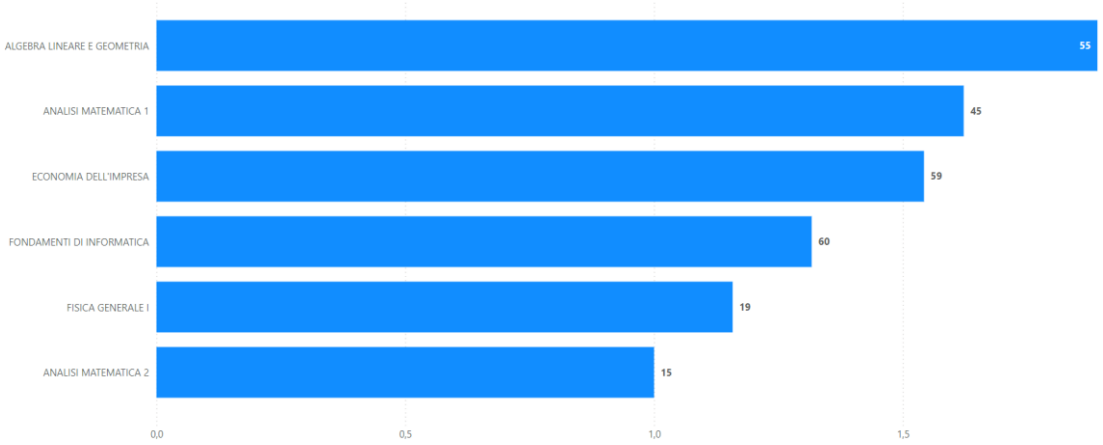

Figura 3.VIII: Media di volte che un esame è stato provato prima di essere passato (Studenti "Late").

Infine questa parte dell'analisi si conclude con gli studenti che si sono laureati in ritardo: innanzitutto sono presenti meno esami nel grafico, dato che "Fisica Generale II" non è stata passata da nessuno studente nel primo anno accademico, a ulteriore testimonianza di come sia un esame importante che necessita di una ottima preparazione e di come gli studenti "*Late*" devono probabilmente questo ritardo nella laurea a uno scarso impegno nel primo anno accademico, guardando anche i numeri relativi a quanti studenti hanno passato i vari esami, i quali risultano particolarmente bassi.

Inoltre, l'esame che è stato provando più volte, il quale rimane "Algebra Lineare e Geometria" anche in questo caso, presenta una media di 1.89 volte per essere passato, seguito questa volta da "Analisi Matematica 1" e poi "Economia dell'Impresa".

Un'ulteriore informazione che può essere dedotta dal dataset riguardo ai percorsi accademici degli studenti su cui basiamo l'analisi è quella riguardante la distribuzione della media dei voti ottenuti dal sostenimento degli esami nel primo anno.

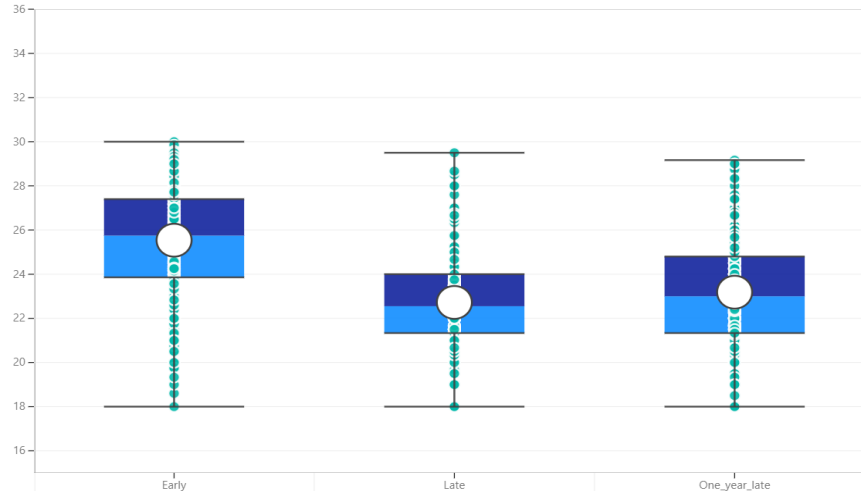

Figura 3.IX: Distribuzione medie voti del primo anno per categoria di studenti.

La distribuzione mostra perciò come per gli studenti appartenenti al primo gruppo la media dei voti dello specifico studente può andare da 18 a 30: quindi viene coperto l'intero range di voti possibili. Per quanto riguarda gli ultimi due gruppi, non vi è alcuno studente con una media voti per il primo anno pari a 30, sebbene alcuni si avvicinino a tale valore. Tuttavia, il limite inferiore della media voti per queste categorie rimane invariato, attestandosi su un valore di 18.

Quindi, per quanto riguarda il range di valori della distribuzione non sono evidenti differenze significative. Tuttavia, ciò che risulta immediatamente evidente, anche a livello visivo, è la differenza tra i valori di alcune statistiche descrittive, come la media, la mediana e i vari quartili, osservabile attraverso il posizionamento dei box colorati.

Per un'analisi più accurata nella seguente tabella verranno riportati i valori relativi a queste statistiche per poter effettuare un confronto più accurato:

Categoria	Primo quartile	Mediana	Terzo Quartile	Media
Early	23.86	25.75	27.40	25.54
One_year_late	21.33	23.00	24.80	23.18
Late	21.33	22.55	24.00	22.72

Tabella 3.III: Statistiche distribuzioni medie voti del primo anno per categoria di studenti.

La tabella mostra quindi che oltre a delle performance migliori in termini di tempo impiegato per laurearsi, gli studenti che si laureano in corso presentano anche delle performance significativamente migliori in termini di voti per il primo anno accademico. Difatti non solo la media è molto più alta rispetto agli altri due gruppi, ma un'informazione che può essere ancora più significativa è data dal confronto tra primo quartile del gruppo "*Early*" con il terzo quartile degli altri gruppi di studenti. Tale confronto mette in luce come gli studenti con le performance accademiche più basse nel gruppo di coloro che si sono laureati in tempo ottengano comunque risultati paragonabili a quelli degli studenti più performanti delle altre categorie.

I risultati ottenuti nel primo anno vengono poi messi a confronto con quelli relativi alla media voto dell'intero percorso accademico, per cercare di comprendere se una proiezione può essere fatta sulle basi delle performance del primo anno.

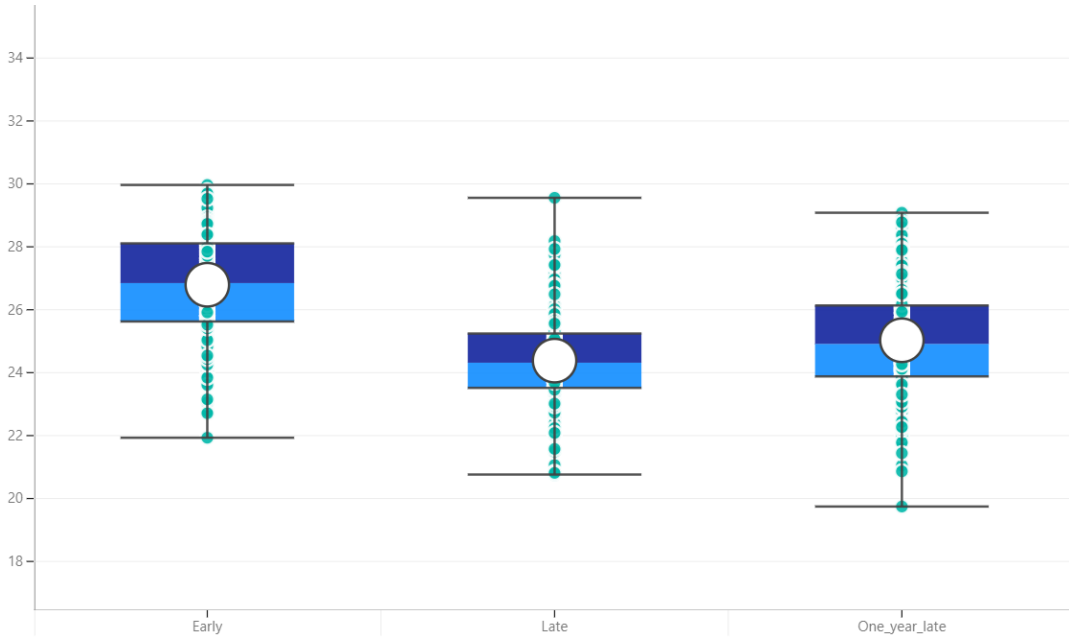

Figura 3.X: Distribuzione medie voti intero percorso accademico per categoria di studenti.

Partendo, come per l'analisi della media voti del primo anno, dal range all'interno del quale la distribuzione è definita, vediamo come, a fronte di un limite superiore

che è rimasto più o meno lo stesso, con un outlier che aumenta di molto quello della categoria "*Late*", i limiti inferiori sono tutti più alti rispetto al primo anno, a fronte di un miglioramento generale delle performance degli studenti negli altri anni.
Addentrandosi poi nell'analisi dettagliata delle statistiche, anche in questo caso una tabella verrà presentata per effettuare un confronto più chiaro:

Categoria	Primo quartile	Mediana	Terzo Quartile	Media
Early	25.63	26.85	28.11	26.79
One_year_late	23.88	24.92	26.14	25.03
Late	23.52	24.32	25.24	24.38

Tabella 3.IV: Statistiche distribuzioni medie voti dell'intero percorso accademico per categoria di studenti.

La tabella evidenzia delle statistiche simili a quelle mostrate nella prima, con il gruppo degli studenti che si sono laureati all'interno del periodo di tre anni e mezzo che rimane il più performante anche in termini di voti ottenuti e quindi conseguentemente di voto di laurea con cui hanno conseguito il titolo.
Si può notare, tuttavia, un miglioramento significativo delle prestazioni degli studenti "*One_year_late*" e "*Late*", che potrebbero però essere dovuti al fatto che avendo avuto più tempo a disposizione si siano concentrati sul prendere voti migliori.
Dal confronto delle due tabelle emerge dunque che, in generale, una partenza performante in termini di voti medi ottenuti coincide poi con una base di voto di laurea finale alto, a testimonianza di come il primo anno ci dia già delle informazioni estremamente significative riguardo i percorsi accademici degli studenti.

Capitolo 4: MODELLI DI PROCESSO

Dopo aver mostrato le statistiche, sia generali che specifiche del primo anno, la corrente analisi prosegue con la costruzione dei modelli di processo tramite due algoritmi di discovery Process mining, quali l'inductive miner e l'heuristic miner, per poi andare a confrontare i risultati da essi derivanti.

4.1 Costruzione modelli di processo

Nel presente sottoparagrafo verranno implementati quindi gli algoritmi introdotti precedentemente e analizzeremo i risultati prima per l'inductive miner e poi quelli dell'heuristic miner.

4.1.1 Inductive miner

L'inductive miner è stato applicato sia per la creazione dei modelli processo relativi all'event log con solamente gli esami una volta passati che in quello che presenta anche gli esami con gli stati, che è quello su cui verrà effettuato il focus in questa analisi.

Per creare i vari modelli di processo è stato preso quindi il dataframe con gli stati, già diviso nelle tre differenti categorie di studenti, relative a chi si è laureato in tempo, chi si è laureato con un anno in ritardo e chi, invece, ha impiegato più anni per il conseguimento della laurea, in modo da poter avere qualche informazione riguardo al perché uno studente si laurea prima rispetto ad un altro, sulla base dei percorsi fatti al primo anno.

Si è iniziato con l'implementazione dell'algoritmo inductive per gli studenti che si sono laureati in tempo, ossia coloro che costituiscono il dataset "df_merged_caricato_early". I risultati relativi alle quattro metriche utilizzate per valutare quale tra i 10 valori di *threshold* utilizzati possa essere considerato il migliore sono i seguenti:

Threshold	Fitness	Precision	Generalization	Semplicity
0	1	0.344348	0.864076	0.610738
0.1	1	0.397652	0.86295	0.605442
0.2	0.992433	0.480466	0.880777	0.62963
0.3	0.967122	0.461998	0.766045	0.621622
0.4	0.751758	0.47976	0.791356	0.6
0.5	0.713149	0.511254	0.781062	0.642857
0.6	0.5944	0.43036	0.750683	0.632653
0.7	0.644313	0.557911	0.771107	0.670886
0.8	0.735025	0.678607	0.784054	0.634409
0.9	0.636634	0.506331	0.707531	0.642105
1	0.891628	0.615261	0.577099	0.652174

Tabella 4.I: Valori metriche modelli di processo tramite inductive miner al variare del valore k per studenti Early.

Dati questi risultati il *threshold* migliore viene identificato come quel valore che riporta il miglior *trade-off* tra le quattro metriche calcolate.

Perciò quello che risulta essere il migliore in questo caso è il threshold con un valore pari a 0.2, che costruisce il seguente modello di processo:

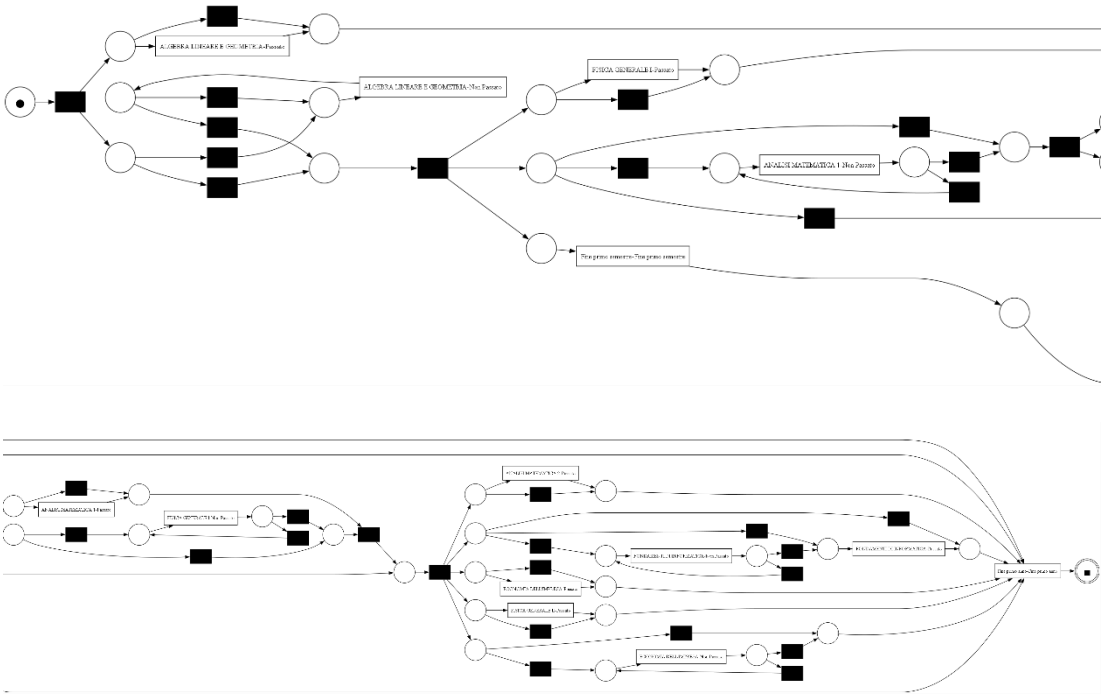

Figura 4.I: Modello di processo per Early students con k=0.2. Rappresentazione tramite rete di Petri.

Questo valore di k però rende il modello troppo complesso a causa dello scarso filtraggio dato dal valore basso assunto, per questo motivo se si volesse definire un modello di processo più semplice si potrebbe scegliere il valore pari a 0.8, che tra quelli più filtrati è il *threshold* che presenta le metriche migliori, similmente al valore 1, che però corrisponderebbe ad un estremo filtraggio e dunque risultati troppo semplificati.

Con un valore di k pari a 0.8 il modello di processo definito è invece il seguente:

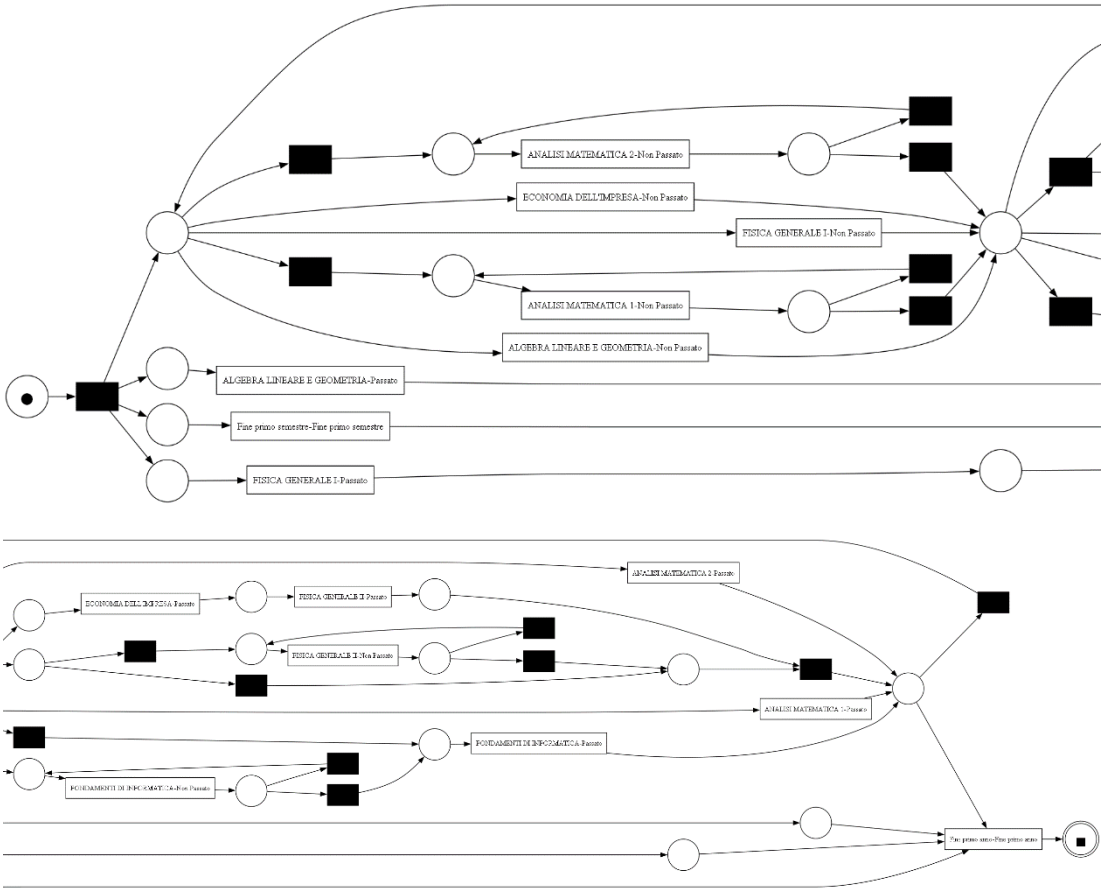

Figura 4.II: Modello di processo per Early students con k=0.8. Rappresentazione tramite rete di Petri.

Il modello appena mostrato evidenzia informazioni particolarmente importanti come la presenza di cicli su “Analisi Matematica 1”, “Analisi Matematica 2”, “Fondamenti di Informatica” e “Fisica Generale II”.

L’algoritmo dell’Inductive miner è stato poi utilizzato per definire il modello di processo più rilevante per i percorsi accademici degli studenti appartenenti al gruppo “One_year_late”. I risultati derivati dal *loop* sul k da 0.1 a 1 con salti di 0.1 sono stati riportati nella seguente tabella:

Threshold	Fitness	Precision	Generalization	Semplicity
0	1	0.380277	0.782739	0.622378
0.1	0.996415	0.493215	0.833996	0.623188
0.2	0.889084	0.810245	0.808098	0.625
0.3	0.954302	0.536138	0.795512	0.635036
0.4	0.951295	0.56729	0.78145	0.621622
0.5	0.7461	0.629281	0.786629	0.656566
0.6	0.697415	0.630555	0.778606	0.677419
0.7	0.783782	0.739444	0.771369	0.708333
0.8	0.685324	0.657535	0.679454	0.684211
0.9	0.700647	0.628668	0.66484	0.684211
1	0.606609	0.633242	0.64333	0.555556

Tabella 4.II: Valori metriche modelli di processo tramite inductive miner al variare del valore k per studenti One_year_late.

Come si può notare dai valori assunti dalle metriche al variare del *threshold*, anche in questo caso la scelta migliore sarebbe riconducibile a un valore della soglia pari a 0.2. Tuttavia, però, per lo stesso discorso fatto precedentemente è stato preferito un modello più filtrato e semplificato, corrispondente al valore 0.7.

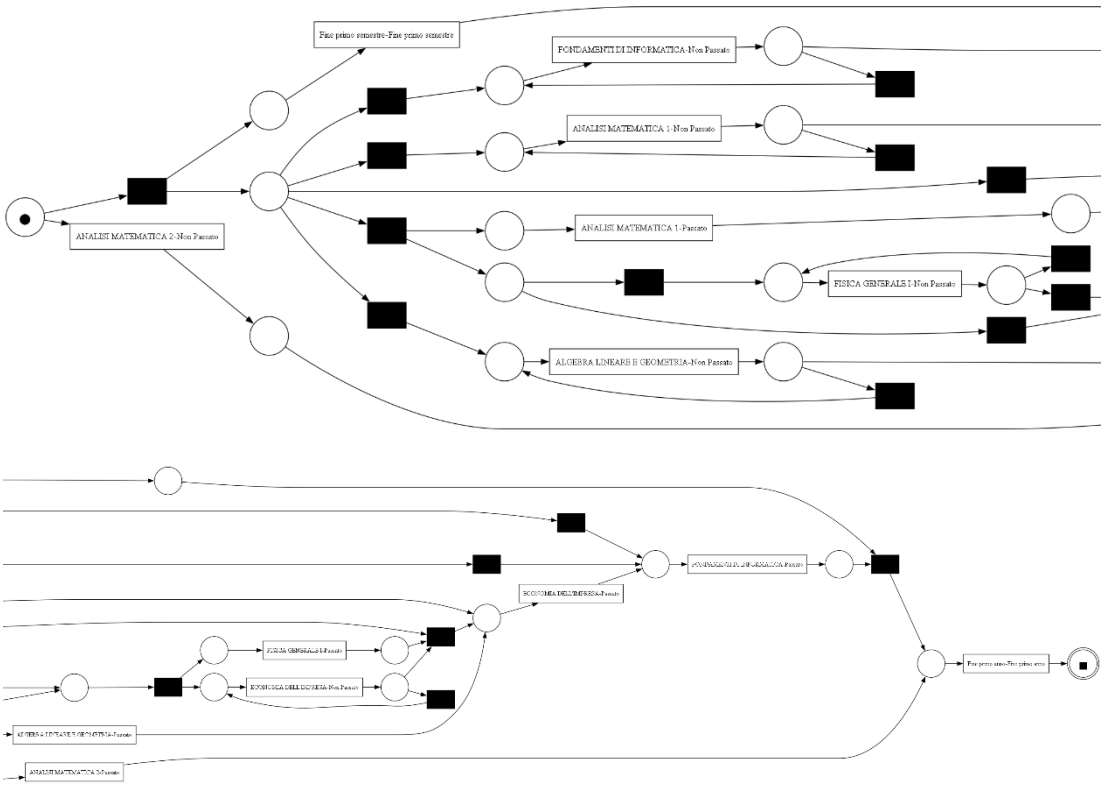

Figura 4.III: Modello di processo per One_year_late students con k=0.7.

Rappresentazione tramite rete di Petri.

Si può notare infatti come questo modello sia più lineare e semplice. Inoltre, sono presenti anche in questo caso dei loop per alcuni esami che solitamente necessitano di essere dati più volte per gli studenti "*One_year_late*", quali "Fondamenti di Informatica", "Analisi Matematica 1", "Algebra Lineare e Geometria", "Fisica Generale I" ed "Economia dell'Impresa".

Infine, si è scoperto il modello di processo migliore per gli studenti che hanno impiegato un periodo maggiore di quattro anni e mezzo per laurearsi, ossia i "*Late*".

Threshold	Fitness	Precision	Generalization	Semplicity
0	1	0.49607	0.799774	0.627586
0.1	0.999779	0.476484	0.803171	0.62963
0.2	0.974962	0.544266	0.805319	0.642857
0.3	0.916473	0.433536	0.663922	0.619048
0.4	0.791847	0.625689	0.777918	0.662338
0.5	0.781478	0.588044	0.744638	0.647059
0.6	0.740228	0.587872	0.729121	0.641026
0.7	0.740228	0.587872	0.729121	0.641026
0.8	0.715621	0.588677	0.701786	0.633803
0.9	0.729562	0.569256	0.679545	0.625
1	0.947564	0.717581	0.587378	0.692308

Tabella 4.III: Valori metriche modelli di processo tramite inductive miner al variare del valore k per studenti Late.

Analogamente alle altre due categorie di studenti, anche in questo caso la soglia che presenta le metriche più elevate è pari a 0.2. Tuttavia, il modello di processo adottato corrisponde a una soglia di k=0.7, il quale, pur avendo metriche uguali a quelle ottenute con una soglia di 0.6, è stato scelto considerando che entrambi i valori costruiscono lo stesso modello.

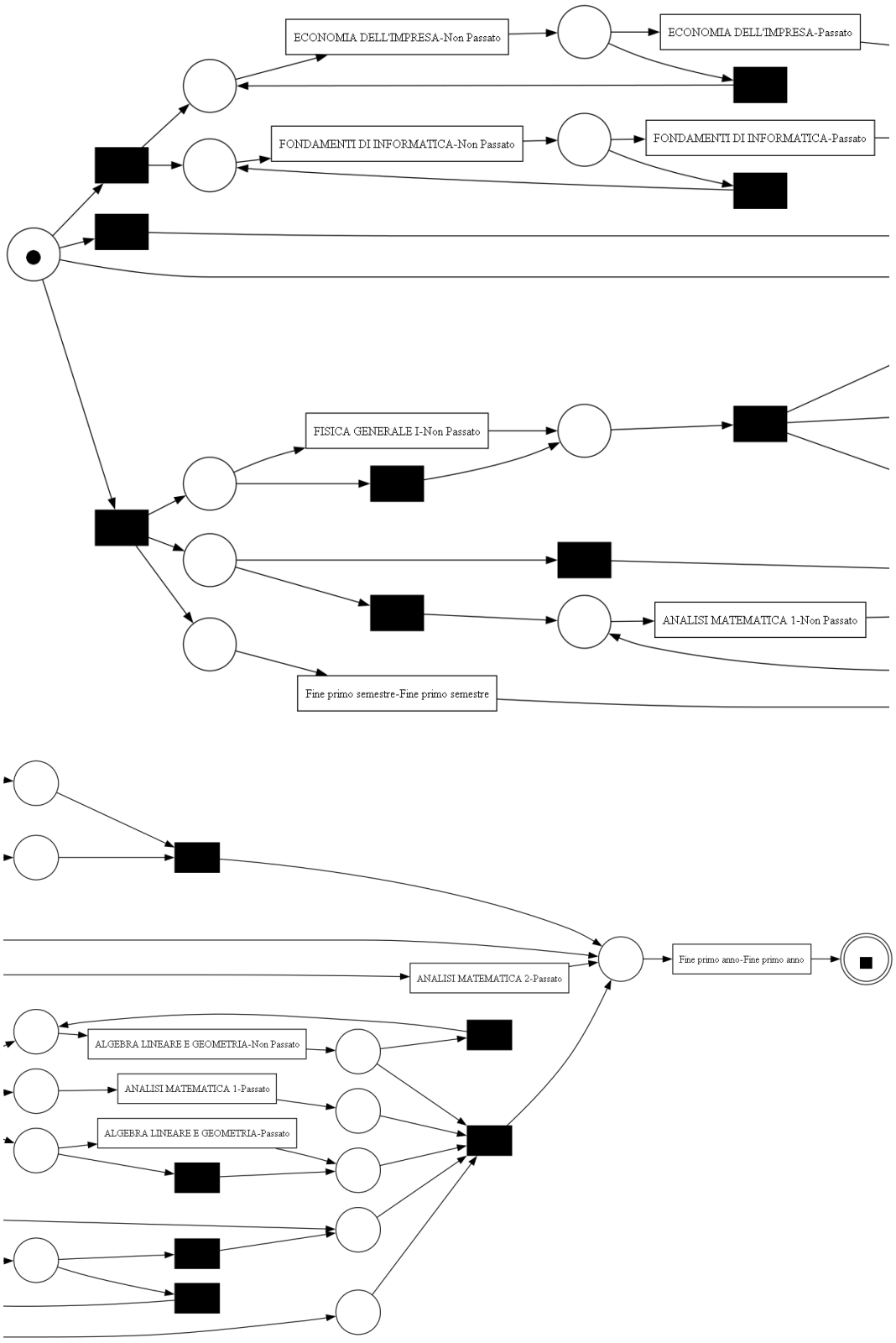

Figura 4.IV: Modello di processo per Late students con k=0.7. Rappresentazione tramite rete di Petri.

Come si può notare dalla rete di Petri appena mostrata in figura, il modello di processo relativo agli studenti appartenenti all'ultimo gruppo analizzato è particolarmente più semplice rispetto agli studenti "*One_year_late*" e ancora di più rispetto agli studenti "*Early*", nonostante la soglia di quest'ultimi sia stata scelta come 0.8, la quale rende quindi il modello più semplice.
Questa maggiore semplicità del modello corrisponde però, in questo caso, a una maggiore scarsità di attività svolte dagli studenti, a testimonianza quindi di come gli studenti che si sono laureati più in ritardo non si siano impegnati particolarmente nel loro primo anno accademico, tralasciando esami importanti, come "Fisica Generale II".

Anche questa rete presenta informazioni particolarmente importanti, quali ad esempio i cicli per gli esami di "Fondamenti di Informatica", "Analisi Matematica 1", "Algebra Lineare e Geometria", "Fisica Generale I" ed "Economia dell'Impresa", come per gli studenti "*One_year_late*".

4.1.2 Heuristic miner

Successivamente all'inductive miner, è stato poi implementato l'heuristic miner per scoprire i modelli di processo relativi agli studenti divisi per categoria, Come detto nella sezione relativa alla metodologia, per questo algoritmo il *threshold* k minimo viene settato più alto, perché anche filtrando non troppo potrebbe produrre dei modelli cosiddetti "*spaghetti-like*", ossia che hanno troppo rumore all'interno.

Threshold	Fitness	Precision	Generalization	Semplicity
0.80	0.708312	0.950575	0.734217	0.5
0.82	0.753161	0.956536	0.746536	0.514019
0.84	0.757266	0.960875	0.743272	0.511962
0.86	0.760567	0.960694	0.747643	0.523316
0.88	0.752776	0.83396	0.75553	0.530055
0.90	0.764228	0.814938	0.766808	0.536424
0.92	0.768082	0.817934	0.777648	0.544828
0.94	0.685857	0.724436	0.825666	0.643836
0.96	0.722825	0.591917	0.791655	0.698113
0.98	0.878192	0.456838	0.943393	1
1	0.746985	0.415232	0.896772	1

Tabella 4.IV: Valori metriche modelli di processo tramite heuristic miner al variare del valore k per studenti Early.

La tabella evidenzia come anche a valori relativamente alti della soglia, come ad esempio 0.8, la metrica relativa alla semplicità è decisamente bassa (0.5) e quindi corrisponde a un modello di processo già poco filtrato, a testimonianza di come aumentare la soglia minima sia la scelta più sensata.

Il valore del k per gli studenti "*Early*" che riporta delle metriche soddisfacenti, è pari a 0.92, nonostante anche ai valori seguenti corrispondano a dei buoni risultati, ma risulterebbero troppo filtrati.

Dunque, il modello di processo costruito tramite l'heuristic miner con un threshold pari a 0.92 è il seguente:

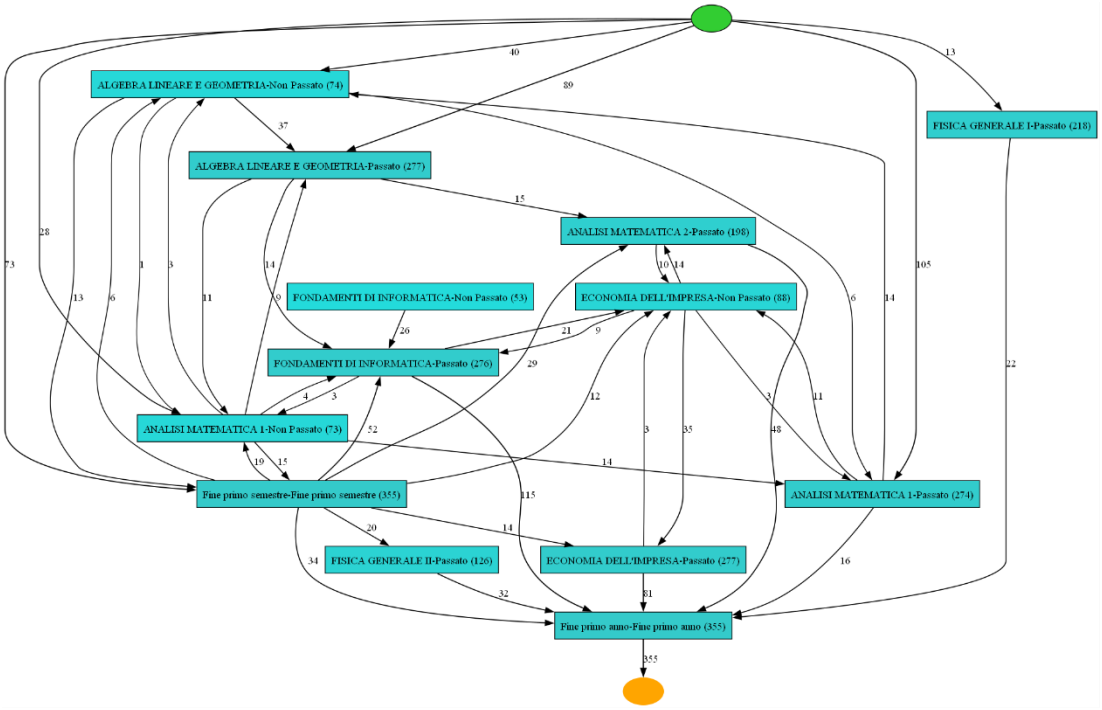

Figura 4.V: Modello di processo per Early students con k=0.92.

In primo luogo, si può vedere come per quanto riguarda l'Heuristic miner si è preferita una rappresentazione tramite la rete Heuristic piuttosto che tramite rete di Petri, data la maggior semplicità di visualizzazione e interpretazione, soprattutto per quanto riguarda i loop, oltre che a una maggiore chiarezza di come è costruito il modello di processo grazie al numero di studenti che hanno svolto un'attività riportato tra parentesi di fianco all'attività stessa, oltre che quelli sugli archi, che evidenziano quanti studenti sono passati dal primo nodo al secondo.

Passando all'analisi della rete, le informazioni più significative sono relative al fatto che gli esami più sostenuti sono "Algebra Lineare e Geometria" e "Economia dell'Impresa", entrambi passati da 277 studenti, tra i 355 che si sono laureati in tempo. Inoltre, si può notare che "Economia dell'Impresa" è stato passato non al

primo tentativo per diversi studenti, così come le altre materie a parte “Fisica Generale” sia I che II e “Analisi Matematica 2”.

Inoltre, è presente un loop tra “Fine primo semestre” e “Analisi Matematica 1-Non Passato” a testimonianza di come questo sia un esame particolarmente difficile e quindi provato diverse volte prima di essere passato in alcuni casi, mentre 14 lo hanno passato subito dopo essere stati bocciati la prima volta.

Passando ora al secondo gruppo, ossia gli studenti categorizzati come “One_year_late”, i risultati delle metriche relative all’applicazione dell’algoritmo *Heuristic miner* sono le seguenti:

Threshold	Fitness	Precision	Generalization	Semplicity
0.80	0.776691	0.93893	0.690955	0.534091
0.82	0.769016	0.948906	0.751695	0.558824
0.84	0.768198	0.0.943898	0.738297	0.578125
0.86	0.76623	0.79944	0.716188	0.567568
0.88	0.762481	0.809856	0.744023	0.588235
0.90	0.763415	0.809856	0.744568	0.6
0.92	0.779565	0.809856	0.744776	0.608247
0.94	0.775231	0.823088	0.748038	0.634146
0.96	0.777525	0.974774	0.745423	0.652174
0.98	0.690713	0.174428	0.918107	1
1	0.808229	0.454978	0.837476	1

Tabella 4.V: Valori metriche modelli di processo tramite heuristic miner al variare del valore k per studenti One_year_late.

In questo caso, dunque, la soglia a cui corrispondono i risultati migliori in termini di metriche è pari a 0.96, sulla base della quale il modello di processo costruito è il seguente:

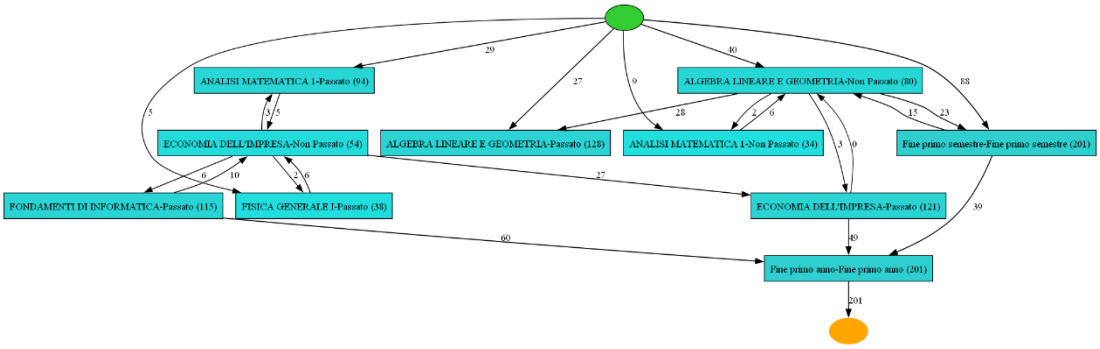

Figura 4.VI: Modello di processo per One_year_late students con k=0.96.

Le prime informazioni significative che questo modello evidenzia sono riferite al numero di studenti che dall'inizio del percorso accademico passa direttamente alla fine del primo semestre, quindi senza sostenere nessun esame, che è maggiore rispetto agli studenti "*Early*" che sono partiti nello stesso modo (88 i primi, 73 i secondi), nonostante il numero di studenti su cui è stato creato questo ultimo modello siano inferiori agli studenti totali che si sono laureati in corso (201 rispetto a 355).

Inoltre, si può vedere come il modello sia molto meno complicato rispetto a quello costruito per gli studenti del primo gruppo: questo perché gli esami dati sono di meno. Difatti gli esami che risultano passati nel primo anno in questo caso sono due in meno rispetto alla situazione precedente, dato che non sono presenti "Analisi Matematica 2- Passato" (che è stato passato da 198 studenti "*Early*") e "Fisica Generale II- Passato" (passato da 126 studenti "*Early*").

Inoltre, questi due esami non risultano nemmeno tentati in questo secondo modello, a testimonianza di come in linea generale vi sia anche un impegno minore, oltre che dei risultati peggiori in termini di performance.

Concludendo infine con gli studenti che si sono laureati più in ritardo, i valori delle metriche prodotti dal loop sulla soglia k sono riportati nella seguente tabella:

Threshold	Fitness	Precision	Generalization	Semplicity
0.80	0.868481	0.936765	0.631254	0.555556
0.82	0.883874	0.921867	0.62995	0.557522
0.84	0.876378	0.935756	0.618876	0.572816
0.86	0.867144	0.820136	0.729641	0.578947
0.88	0.861652	0.794366	0.72419	0.594203
0.90	0.852584	0.960543	0.791792	0.652174
0.92	0.88789	0.970918	0.78932	0.707317
0.94	0.88789	0.970918	0.78932	0.707317
0.96	0.914735	0.377286	0.894488	1
0.98	1	0.824734	0.918621	1
1	0.864217	0.546016	0.807004	1

Tabella 4.VI: Valori metriche modelli di processo tramite heuristic miner al variare del valore k per studenti Late.

In questo caso la tabella presenta le metriche migliori per due valori del k, quali 0.92 e 0.94, che quindi costruiscono lo stesso modello di processo, rendendo la scelta tra i due non influente.

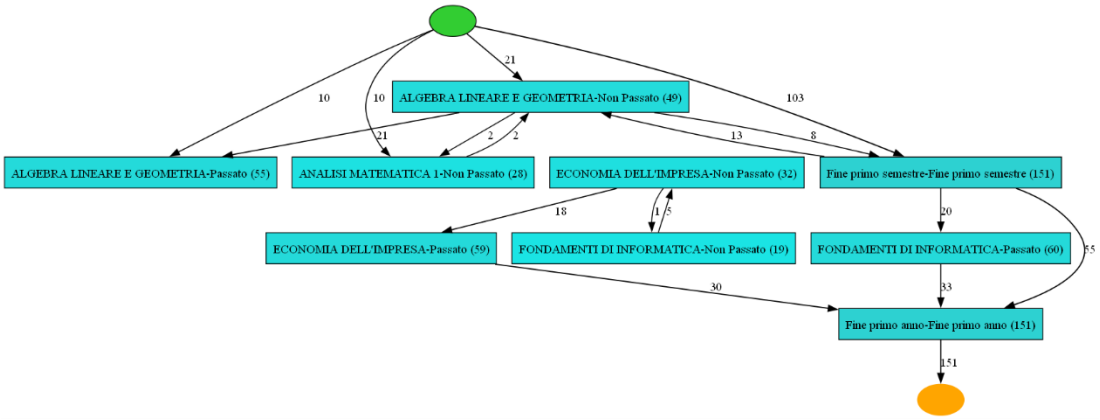

Figura 4.VII: Modello di processo per Late students con k=0.92 / k=0.94.

In questo modello si può notare come il modello sia abbastanza semplice, dato che gli esami sostenuti sono meno rispetto alle categorie di studenti precedenti. Inoltre, il numero di studenti che inizia senza dare nessun esame risulta essere ancora più elevato rispetto alla categoria “One_year_late”, nonostante la base su cui è stato costruito il modello sia ancora più inferiore, ossia 155 studenti. Inoltre, dalla fine del

primo semestre altri 55 studenti passano direttamente alla fine del primo anno, presupponendo che un numero consistente di studenti non ha sostenuto nessun esame nel corso del proprio primo anno accademico, sottolineando una situazione abbastanza grave comune all'interno di questo gruppo.

Gli unici esami passati sono "Algebra Lineare e Geometria", passato da 55 studenti; "Economia dell'Impresa" passato da 59 studenti e "Fondamenti di Informatica" passato da 60 studenti. Per quanto riguarda gli altri esami, a parte "Analisi Matematica 1", che conta 28 osservazioni per quanto riguarda la bocciatura all'esame, non sono presenti né in quanto a situazioni dove lo studente è stato bocciato all'esame, né in quanto a situazioni in cui lo studente è invece passato.

I modelli appena definiti tramite l'applicazione degli algoritmi inductive ed heuristic miner risultano molto importanti nel fornire delle informazioni relative ai vari percorsi intrapresi dagli studenti, divisi sulla base della categoria di appartenenza. Tutto ciò con l'obiettivo ultimo di analizzare il passato identificando gli esami più problematici ed impegnativi e poter poi intervenire e fornire adeguato supporto agli studenti futuri che lo necessitano, in modo tale da ottimizzare i percorsi accademici evitando significativi ritardi.

Difatti da questi modelli si può notare innanzitutto la differenza degli esami passati tra le tre diverse categorie, oltre all'individuazione di esami particolarmente critici e che vengono dunque ripetuti diverse volte, come possono essere, ad esempio, "Analisi Matematica 1" o "Algebra Lineare e Geometria".

Capitolo 5: CONFORMANCE CHECKING

Dopo aver applicato le tecniche di *discovery Process mining* per definire i modelli di processo più significativi per ogni categoria di studenti, sono state applicate le tecniche di *conformance checking* per calcolare metriche relative all'adattamento dei percorsi accademici degli studenti sulla base di una rete di Petri che è stata definita in modo standard seguendo il manifesto del corso universitario a cui gli studenti sono iscritti.

5.1 Costruzione rete di Petri Standard su cui basare conformance checking

La rete di Petri utilizzata come base per l'implementazione delle tecniche di conformance checking e per il calcolo delle metriche derivate è stata tratta dal manifesto del corso universitario a cui sono iscritti gli studenti oggetto di analisi[19]. Tale rete rappresenta il percorso "ideale" che ciascuno studente dovrebbe seguire durante il primo anno accademico per conseguire i migliori risultati possibili.

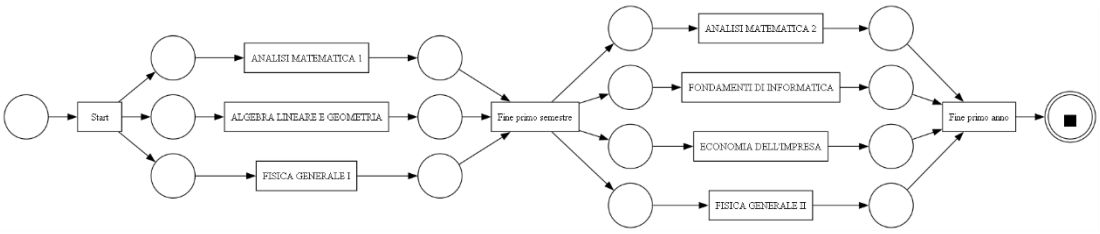

Figura 5.I: Petri net standard per il conformance checking.

[19] Il manifesto del primo anno è rimasto lo stesso per tutti gli anni accademici trattati nell'analisi.

La rete rappresenta quindi il percorso articolato nei due semestri, i quali presentano rispettivamente tre e quattro esami: nel primo si trovano “Analisi Matematica 1”, “Algebra Lineare e Geometria” e “Fisica Generale I”; mentre i quattro del secondo semestre sono “Analisi Matematica 2”, “Fondamenti di Informatica”, “Economia dell’Impresa” e “Fisica Generale II”.

Si può notare quindi, come, nel percorso riportato dalla rete di Petri standard, “Analisi Matematica 2” e “Fisica Generale II” vengono rispettivamente dopo “Analisi Matematica 1” e “Fisica Generale I”. Questo perché, nonostante da regolamento non vi sia propedeuticità tra gli esami, è ovviamente consigliabile e comune che vengano dati precedentemente i primi rispetto i secondi, dato che questi ultimi presentano alcuni argomenti che necessitano delle conoscenze di base fornite nei corsi iniziali per essere capiti.

Inoltre, sono state aggiunte tre attività cosiddette “artificiali” quali “Start”, “Fine primo semestre” e “Fine primo anno” che servono per marcare il percorso: definendo quindi quando il percorso inizia, la data per definire il passaggio da un semestre all’altro e la fine del primo anno. Tra queste però solamente le ultime due sono considerate all’interno del calcolo della fitness.

5.2 Applicazione conformance checking

In questo sottocapitolo verranno presentati i risultati derivanti dall’applicazione delle due tecniche di *conformance checking* applicate.

Si inizierà con l'applicazione della tecnica *token-based replay*, che riporta quindi come *output* un valore specifico per ogni studente, in un range che, come si può vedere ordinando prima in maniera crescente e poi decrescente, va da 0.222222 a 1, come illustrato nella seguente figura:

replayed_fitness	replayed_fitness
0.222222	1
0.222222	1
0.222222	1
0.222222	1
0.222222	1
0.222222	1
0.222222	1
0.222222	1
0.222222	1
0.222222	1
0.222222	1
0.3	1

Figura 5.II: Range di valori della fitness relativa alla tecnica token-based replay.

Se il valore 1 è normale perché significa che quello specifico studente ha sostenuto e passato tutti gli esami del primo anno entro la fine dello stesso, avere un valore minimo che non sia zero potrebbe risultare strano, dato che su più di 700 studenti è improbabile che nemmeno uno non abbia dato nessun esame al primo anno. La risposta a questo quesito è data dal fatto che tutti i percorsi accademici degli studenti, così come quello definito come standard, presentano all'interno le due attività

"artificiali" create per una maggiore completezza dell'analisi e che essendo presenti in tutti i percorsi rendono la *fitness* minima calcolabile pari a 0.222222[20].

Per ottenere una visione più ampia e significativa, i risultati sono stati sintetizzati all'interno di un *boxplot*, che rappresenta la distribuzione della *fitness* degli studenti. Questo consente di evidenziare in modo chiaro le differenze tra le varie categorie esaminate.

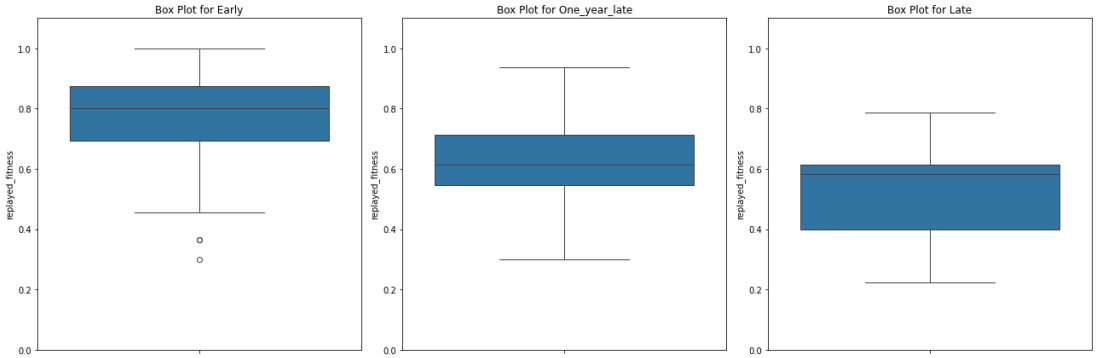

Figura 5.III: Boxplot relativo alle fitness calcolate tramite la tecnica token-based degli studenti divisi per categoria.

Come si può notare, le statistiche relative alla distribuzione tendono a diminuire partendo dagli studenti "*Early*" e arrivando ai "*Late*", passando per quelli del gruppo "*One_year_late*". Nel dettaglio, si nota una traslazione verso il basso sia del limite inferiore che superiore del range in cui i dati sono distribuiti. Solo gli studenti che si laureano in tempo raggiungono una percentuale di fitness pari al 100% rispetto al percorso standard, mentre solo gli studenti "Late" presentano un limite inferiore pari a 0,2222, il che indica che non hanno superato alcun esame del primo anno.

[20] Dato che la token-based è data dal rapporto tra token consumati su token prodotti; quindi, essendo le attività 9 (7 esami più "fine primo semestre" e "fine primo anno") e le attività sostenute solamente le due artificiali, la fitness è data da 2/9 = 0.2222.

Inoltre, la parte colorata del grafico, che rappresenta le osservazioni comprese tra il primo e il terzo quartile, evidenzia chiaramente come i valori migliori tendano a diminuire progressivamente passando dalla prima all'ultima categoria. L'unica eccezione riguarda la mediana, che nei gruppi "One_year_late" e "Late" si mantiene su valori simili, nonostante la generale tendenza di decrescita, a testimonianza di come tra gli studenti appartenenti al gruppo “Late” si trovino anche degli *outlier* che presentano dei valori molto negativi, come ad esempio proprio quelli che non hanno dato nessun esame.

Passando poi alla seconda tecnica di conformance checking utilizzata, ossia quella basata sugli *alignments*, si può notare, dalla tabella sottostante, come già la prima differenza rispetto alla prima è data dal fatto che il range di valori assunti dalle *fitness* degli studenti varia:

alligned_fitness1	alligned_fitness1
0.428571	1
0.428571	1
0.428571	1
0.428571	1
0.428571	1
0.428571	1
0.428571	1
0.428571	1
0.428571	1
0.428571	1
0.428571	1
0.428571	1
0.428571	1
0.428571	1
0.428571	1
0.428571	1
0.428571	1
0.428571	1
0.428571	1
0.428571	1
0.428571	1
0.461538	1

Figura 5.IV: Range di valori della fitness relativa alla tecnica alignments-based replay.

Difatti, si nota come, nonostante ovviamente il valore massimo rimanga 1, a cambiare è il valore minimo, che passa da 0.22222 nella prima tecnica utilizzata, a un valore decisamente più alto, pari a 0.428571. Questo perché la tecnica *token-based* penalizza molto l'assenza di token previsti, dato che si basa su un semplice rapporto, mentre la tecnica basata sugli *alignments* è più flessibile, dato che cerca di ottimizzare gli allineamenti e riducendo il costo complessivo delle deviazioni.
Anche in questo caso, per avere una quantità di informazioni maggiore riguardanti gli studenti in generale, sono state presentate, per ciascuna categoria di studenti, le distribuzioni dei risultati avuti in termini di *fitness* tramite tre diversi *boxplot*.

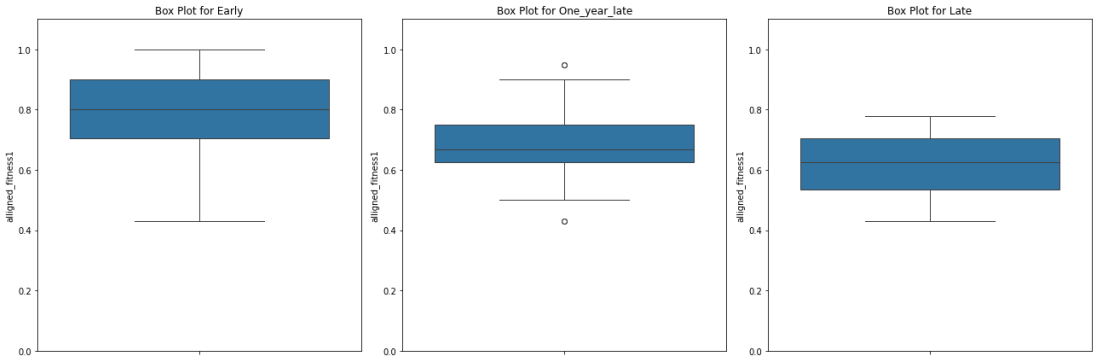

Figura 5.V: Boxplot relativo alle fitness calcolate tramite la tecnica aligned-based degli studenti divisi per categoria.

I grafici riportano più o meno le stesse informazioni che già si trovavano nei boxplot relativi alla tecnica basata su *token*, ossia che le statistiche della distribuzione tendono a diminuire a mano a mano che si passa dal gruppo di studenti che si sono laureati nel minor tempo fino ad arrivare a coloro che ci hanno messo di più.
Tuttavia, vi è una eccezione relativa al limite inferiore delle distribuzioni, difatti, stranamente, quello per il gruppo "*Early*" è intorno a 0.4 e quindi presumibilmente pari a 0.428571; mentre quello del secondo gruppo riporta un valore più alto, di circa 0.5, rendendo quindi la distribuzione estremamente concentrata, considerando anche

che il limite superiore di questa distribuzione non arriva a 1 come per i primi, ma bensì intorno a 0.9.

Capitolo 6: GRAFI

In questo capitolo verranno introdotti i risultati ottenuti riguardo ai grafi, partendo innanzitutto dai Directed graphs utilizzati per rappresentare i percorsi accademici per tutti gli studenti, ponendo particolare attenzione al dataset che contiene anche gli stati relativi all'esame sostenuto. Dopodiché si parlerà dei sottografi estratti tramite SUBDUE e ci come verranno utilizzati conseguentemente nell'utilizzo delle tecniche di machine learning per l'obiettivo della tesi.

6.1 Directed Graphs

La prima cosa che è stata definita per la costruzione dei grafi è la concorrenza tra gli esami sostenuti, definita come si può vedere nel capitolo relativo alla metodologia, sulla base di una colonna che è stata creata chiamata "DATA_START", che è calcolata sulla base della data in cui l'esame è stato sostenuto ("DATA_APP"), sottraendo un mese a quest'ultima informazione.

L'algoritmo definito agisce quindi come riportato nella figura sottostante:

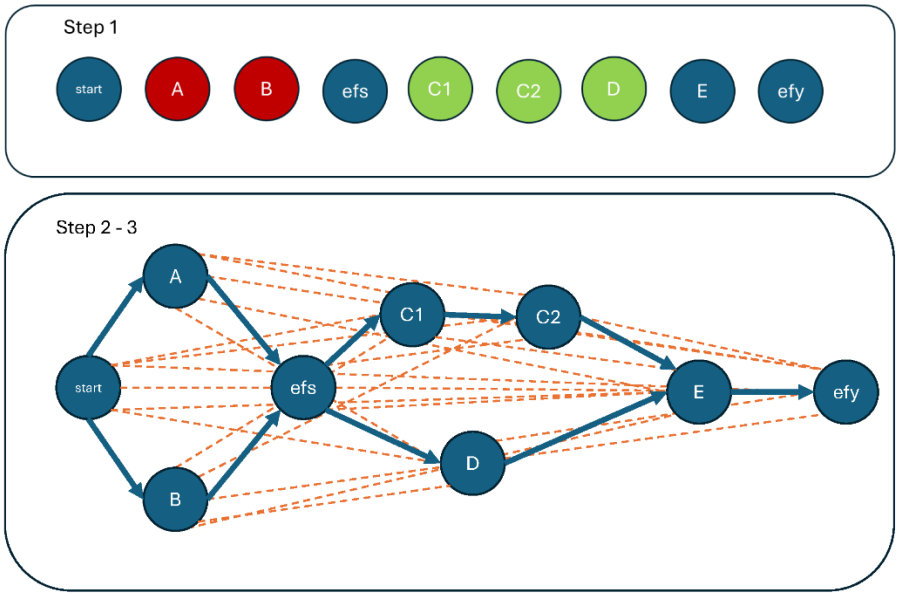

Figura 6.I: Come agisce l'algoritmo di concorrenza definito.

Come si vede nella figura l'algoritmo è articolate in tre step. Si parte dal primo in cui si hanno tutte le attività (comprese anche quelle artificiali *Start, efs* ed *efy)*, dove quelle colorate in rosso e in verde sono rispettivamente concorrenti tra di loro (C1 e C2 sono riferite alla stessa attività).

Nel secondo riguardo della figura sono rappresentati gli step 2 e 3: nel secondo passo le attività vengono collegate tutte tra di loro sulla base della sequenzialità degli identificatori dei nodi, mentre nel terzo, tramite la *transitive reduction*, vengono eliminate tutti gli archi considerati inutili, rimanendo quindi con il grafo che è rappresentato da nodi e archi di colore blu.

Dal grafo finale si può notare quindi come venga rappresentata la concorrenza e come per le attività che fanno riferimento allo stesso esame, come C1 e C2, nel grafo vi sarà comunque consequenzialità nonostante i periodi di studio determinati si sovrappongano.

Successivamente, nelle prossime pagine, verranno mostrati alcuni esempi relativi ai grafi costruiti per ogni categoria di studenti.

Partendo dal gruppo degli studenti che si sono laureati in tempo, un esempio significativo e rappresentativo è il seguente:

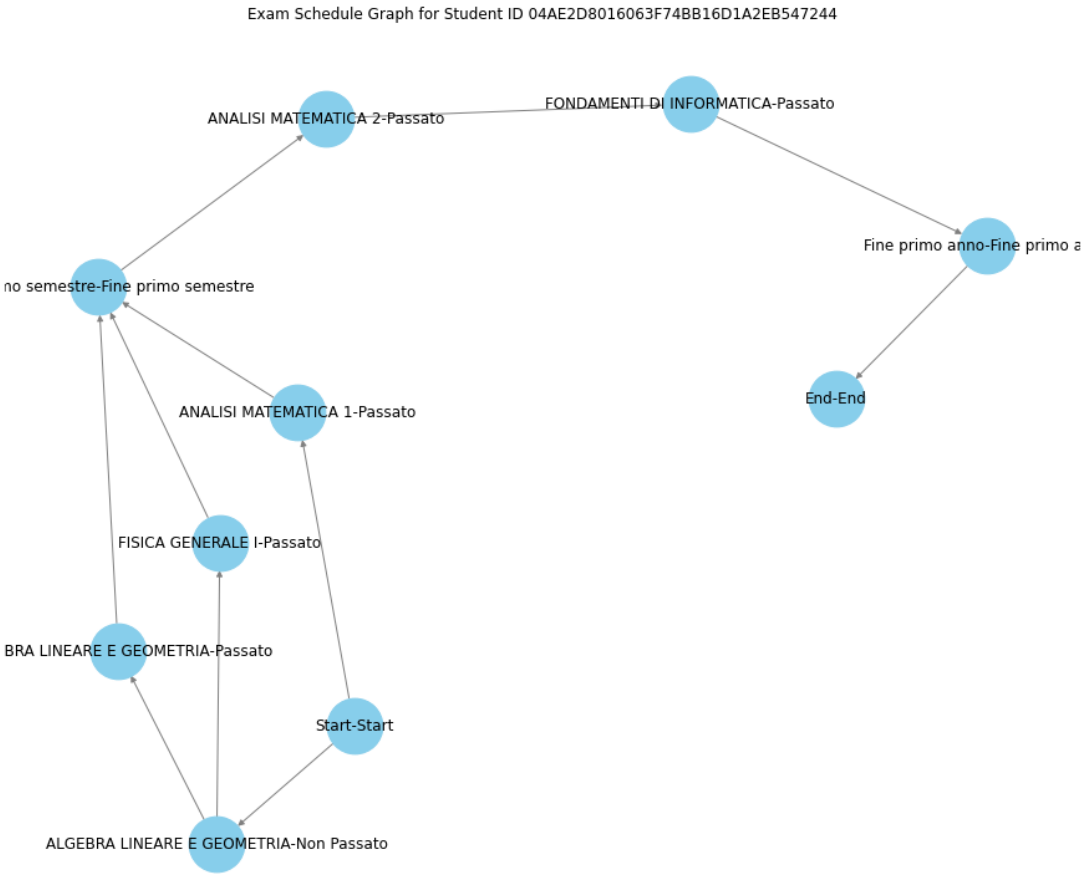

Figura 6.II: Esempio Grafo studente Early.

Si nota innanzitutto da cosa sono costituiti i grafi, ossia archi e nodi, che possono sia riferirsi ad esami passati che ad esami non passati, oltre che alle attività precedentemente create e che sono quindi presenti in ogni grafo, quali “Start”, “Fine primo semestre”, “Fine primo anno” ed “End”.

Dopo di che, passando all’analisi del grafo in sé, questo evidenzia una situazione molto comune negli studenti appartenenti a questa categoria, ossia la presenza di

tanti esami passati, ma soprattutto in concorrenza tra di loro, a testimonianza di come questi studenti si siano impegnati fin dall'inizio, studiando per più esami contemporaneamente. Nel caso specifico, infatti, lo studente parte con il sostenere "parallelamente" due esami, ossia "Analisi Matematica 1" e "Algebra Lineare e Geometria", passando il primo e non il secondo, che però è stato passato conseguentemente, concludendo quindi con tutti e tre gli esami del primo semestre che sono stati dati parallelamente prima della fine di quest'ultimo. Lo studente preso come esempio conclude poi con gli esami "Analisi Matematica 2" e "Fondamenti di Informatica", conseguendo ottimi risultati nel primo anno accademico.
Passando poi ad analizzare l'esempio preso in esame per la seconda categoria di studenti, si possono notare subito delle differenze tra i due grafi visti ad ora.

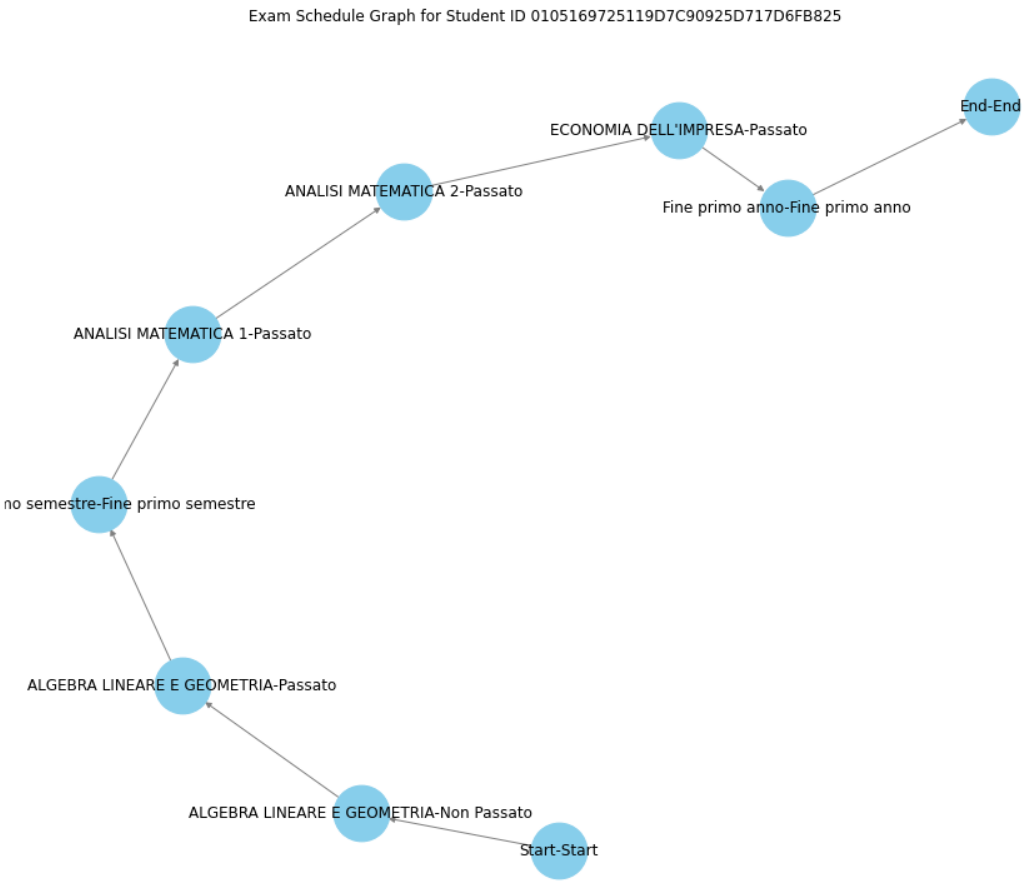

Figura 6.III: Esempio Grafo studente One_year_late.

Nonostante il numero di esami sostenuti e passati non sia così tanto inferiore rispetto a quelli dello studente "*Early*" visto precedentemente, questo grafo ha una differenza importante rispetto al primo: infatti non presenta nessuna concorrenza tra gli esami, pattern che nei gruppi di studenti che si sono laureati in ritardo (chi più gravemente, chi meno) è particolarmente presente. Questa è un'informazione significativa perché la non parallelizzazione degli esami porta lo studente inevitabilmente ad essere più lento nel percorso accademico, perché vuol dire che solitamente si prepara per un esame alla volta, o comunque anche nel caso parta con lo studiare per più di un esame, finirà solitamente per concentrarsi solamente su uno per poi sostenere gli altri in momenti successivi.

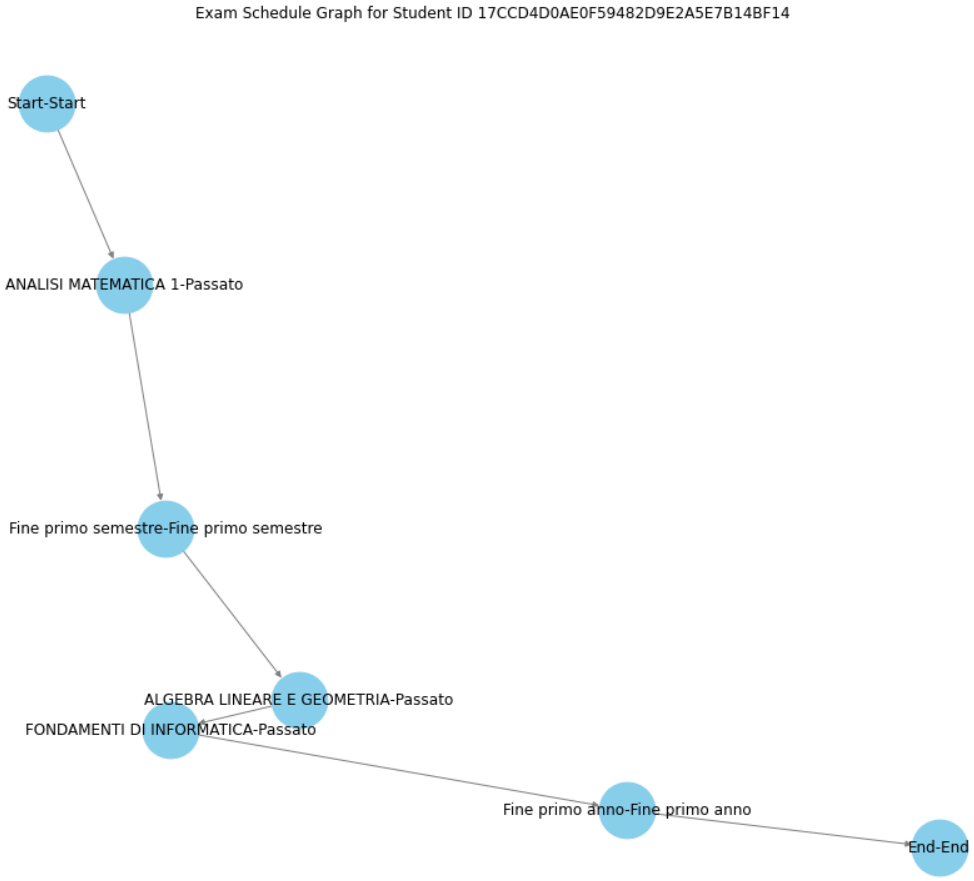

Figura 6.IV: Esempio Grafo studente Late.

Con il grafo rappresentato in quest'ultima figura viene portato l'esempio rappresentativo del percorso di uno studente "*Late*", in cui si può notare anche in questo caso la mancanza di esami sostenuti parallelamente, oltre che l'assenza anche solo di esami almeno tentati e non passati, a testimonianza di uno scarso impegno nel periodo iniziale del percorso accademico.

Concludendo la parte precedente all'estrazione dei sottografi tramite SUBDUE, i grafi costruiti vengono riportati in un file di testo nel formato che è già stato anticipato.

```
XP
v 1 Start-Start
v 2 ALGEBRALINEAREEGEOMETRIA-Passato
v 3 ANALISIMATEMATICA1-Passato
v 4 Fineprimosemestre-Fineprimosemestre
v 5 FISICAGENERALEI-Passato
v 6 ECONOMIADELLIMPRESA-Passato
v 7 ANALISIMATEMATICA2-Passato
v 8 FONDAMENTIDIINFORMATICA-Passato
v 9 FISICAGENERALEII-Passato
v 10 Fineprimoanno-Fineprimoanno
v 11 End-End
e 1 2 Start-Start_ALGEBRALINEAREEGEOMETRIA-Passato
e 2 3 ALGEBRALINEAREEGEOMETRIA-Passato_ANALISIMATEMATICA1-Passato
e 3 4 ANALISIMATEMATICA1-Passato_Fineprimosemestre-Fineprimosemestre
e 4 5 Fineprimosemestre-Fineprimosemestre_FISICAGENERALEI-Passato
e 5 6 FISICAGENERALEI-Passato_ECONOMIADELLIMPRESA-Passato
e 5 7 FISICAGENERALEI-Passato_ANALISIMATEMATICA2-Passato
e 6 8 ECONOMIADELLIMPRESA-Passato_FONDAMENTIDIINFORMATICA-Passato
e 7 8 ANALISIMATEMATICA2-Passato_FONDAMENTIDIINFORMATICA-Passato
e 8 9 FONDAMENTIDIINFORMATICA-Passato_FISICAGENERALEII-Passato
e 9 10 FISICAGENERALEII-Passato_Fineprimoanno-Fineprimoanno
e 10 11 Fineprimoanno-Fineprimoanno_End-End

XP
v 1 Start-Start
v 2 FISICAGENERALEI-Passato
v 3 ANALISIMATEMATICA1-Passato
v 4 ALGEBRALINEAREEGEOMETRIA-NonPassato
v 5 Fineprimosemestre-Fineprimosemestre
v 6 ALGEBRALINEAREEGEOMETRIA-NonPassato
v 7 ALGEBRALINEAREEGEOMETRIA-Passato
v 8 ANALISIMATEMATICA2-Passato
v 9 Fineprimoanno-Fineprimoanno
v 10 End-End
e 1 2 Start-Start_FISICAGENERALEI-Passato
e 1 3 Start-Start_ANALISIMATEMATICA1-Passato
e 1 4 Start-Start_ALGEBRALINEAREEGEOMETRIA-NonPassato
e 2 5 FISICAGENERALEI-Passato_Fineprimosemestre-Fineprimosemestre
e 3 5 ANALISIMATEMATICA1-Passato_Fineprimosemestre-Fineprimosemestre
e 4 5 ALGEBRALINEAREEGEOMETRIA-NonPassato_Fineprimosemestre-Fineprimosemestre
```

Figura 6.V: Esempio di come sono formattati i grafi nel file di testo.

6.2 Subgraphs con SUBDUE

Dunque, dopo aver creato il file di testo con i grafi formattati come in figura, questo file viene utilizzato per implementare l'algoritmo definito in SUBDUE per estrarre i pattern più frequenti tra i percorsi accademici.

I risultati dell'estrazione sono quindi cento sottografi che riportano delle porzioni di percorso.

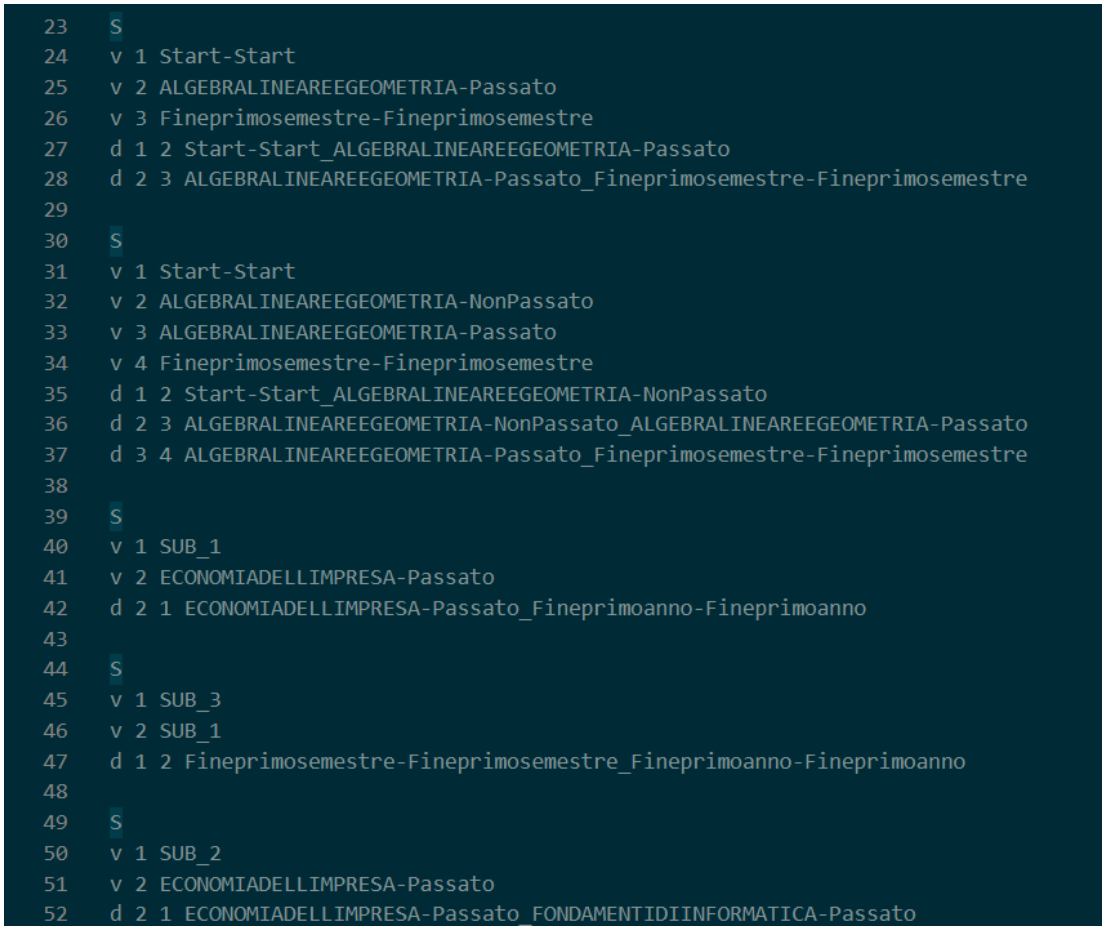

Figura 6.VI: Risultati dell'estrazione dei sottografi tramite SUBDUE.

Come si può notare dai risultati, innanzitutto il formato dei SUB sono diversi rispetto a quelli dei grafi presi in input; infatti, qua i sottografi sono divisi dalla lettera “S” e gli archi sono identificati dalla lettera “d”, piuttosto che “e”; mentre i nodi sono sempre rappresentati dalla lettera “v”, come nel caso precedente.

Successivamente si evidenzia una caratteristica di questi risultati ancora più significativa: dall’implementazione di SUBDUE sono stati estratti anche sottografi annidati, ossia che presentano al loro interno altri SUB, come possiamo notare negli ultimi tre sottografi dello screenshot, dove il penultimo presenta addirittura due sottografi annidati.

Questo comporta problemi con l'utilizzo di questi risultati per combinarli con i dataset che abbiamo già a disposizione e conseguentemente riuscire a tracciare i pattern seguiti dai singoli studenti.

Per questo tutti i sottografi annidati sono stati estesi, in modo tale da avere una situazione più chiara e lineare come riportato nella figura sottostante.

```
40
41  S
42  v 1 ECONOMIADELLIMPRESA-Passato
43  v 2 Fineprimoanno-Fineprimoanno
44  v 3 End-End
45  d 1 2 ECONOMIADELLIMPRESA-Passato_Fineprimoanno-Fineprimoanno
46  d 2 3 Fineprimoanno-Fineprimoanno_End-End
47
48  S
49  v 1 Start-Start
50  v 2 Fineprimosemestre-Fineprimosemestre
51  v 3 Fineprimoanno-Fineprimoanno
52  v 4 End-End
53  d 1 2 Start-Start_Fineprimosemestre-Fineprimosemestre
54  d 2 3 Fineprimosemestre-Fineprimosemestre_Fineprimoanno-Fineprimoanno
55  d 3 4 Fineprimoanno-Fineprimoanno_End-End
56
57  S
58  v 1 ECONOMIADELLIMPRESA-Passato
59  v 2 FONDAMENTIDIINFORMATICA-Passato
60  v 3 Fineprimoanno-Fineprimoanno
61  v 4 End-End
62  d 1 2 ECONOMIADELLIMPRESA-Passato_FONDAMENTIDIINFORMATICA-Passato
63  d 2 3 FONDAMENTIDIINFORMATICA-Passato_Fineprimoanno-Fineprimoanno
64  d 3 4 Fineprimoanno-Fineprimoanno_End-End
```

Figura 6.VII: Esempio di come sono stati estesi i sottografi.

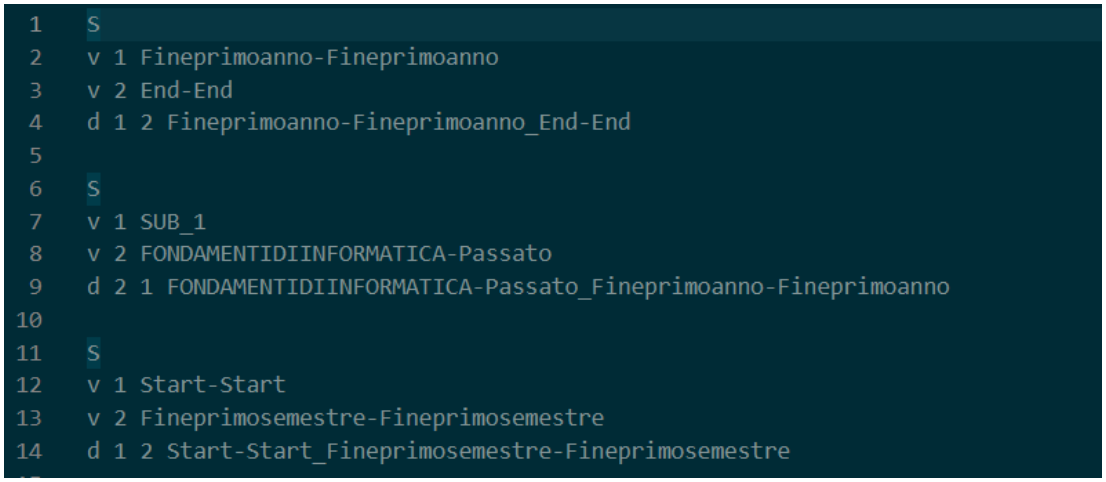

Figura 6.VIII: SUB_1, SUB_2, SUB_3 (dall'alto verso il basso) utilizzati per estendere i sottografi annidati precedenti.

Con questi risultati ora possiamo continuare e finalizzare la nostra analisi utilizzando i sottografi per l'implementazione di tecniche di machine learning, quale il K-Means per la clusterizzazione degli studenti.

Capitolo 7: CLUSTERIZZAZIONE

Nell'ultimo capitolo della tesi si andrà a mostrare, interpretare e discutere i risultati derivanti dalle tecniche di clusterizzazione applicate.

7.1 K-Means

Si parte con l'analizzare i risultati derivanti dall'applicazione del K-Means, che come è stato già detto nel capitolo relativo alla metodologia parte col definire il numero di k, ossia cluster, ottimale sulla base del quale definire l'analisi, tramite l'utilizzo di diverse metriche quali SSE, Silhouette e indice di Davies-Bouldin.

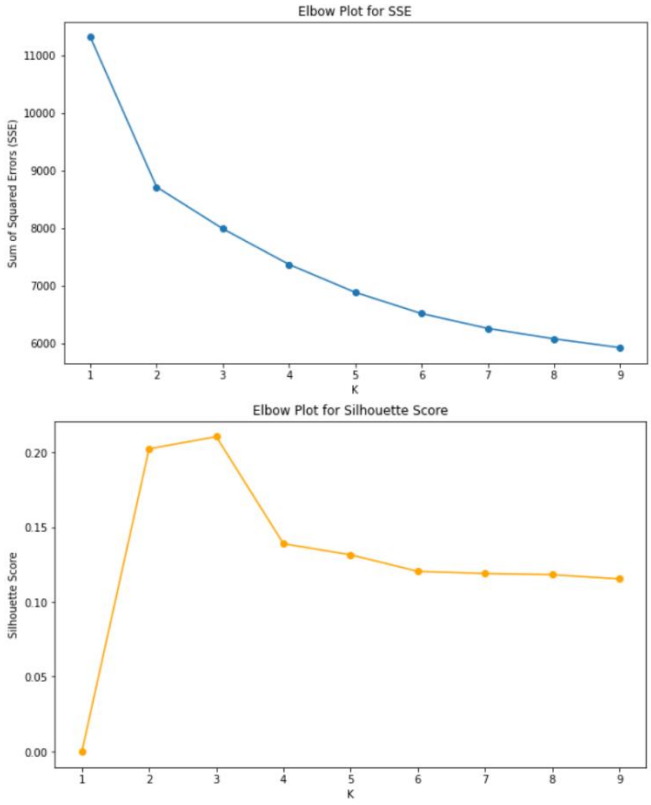

Figura 7.I: Elbow method per SSE e Silhouette insieme.

I due grafici precedenti ci mostrano l'applicazione del metodo conosciuto come "*elbow method*", ossia metodo del gomito in italiano. Si chiama in questo modo perché si andrà a scegliere il numero K ottimale sulla base della posizione si crea un "gomito" nel grafico. Infatti, questo gomito si crea quando si ha, al passaggio da un valore di K al successivo, una variazione di SSE, e quindi di omogeneità dei cluster meno netta per il primo grafico, o una variazione negativa più netta per quanto riguarda il grafico relativo alla Silhouette, e quindi della distanza tra cluster differenti.

Si può notare perciò che nel primo grafico il gomito evidente si nota per un valore di cluster pari a 2; mentre il secondo grafico, oltre a confermare che 2 è un valore molto buono su cui basare la clusterizzazione, dato che ha un valore di Silhouette alto, (che sottolinea una buona separazione dei cluster), suggerisce che un valore di K pari a 3 potrebbe essere ancora più efficiente.

Da una prima analisi si ottiene una risposta che tende all'utilizzare entrambi e vedere quale clusterizzazione risulta essere la migliore; perciò, sono stati utilizzati altri due grafici per avere una visione ancora più dettagliata riguardo questa scelta cruciale per l'analisi.

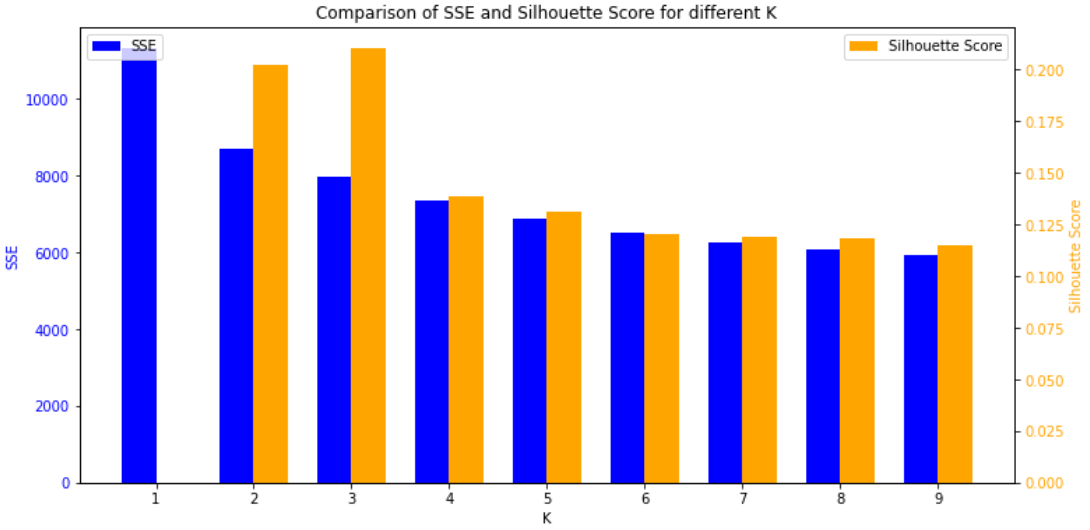

Figura 7.II: Comparazione valori SSE e Silhouette al variare di K.

Anche questo grafico evidenzia come i valori migliori per K siano 2 e 3, dato che vediamo come i valori della SSE vanno sempre più stabilizzandosi dopo questi valori e i valori della Silhouette sono nettamente superiori rispetto agli altri, soprattutto per il valore pari a 3.

Infine, è stato utilizzato l'indice Davies-Bouldin, il quale misura la qualità del clustering, sulla base della separazione tra cluster e la compattezza all'interno degli stessi. I calcoli dell'indice al variare di K portano ai seguenti risultati:

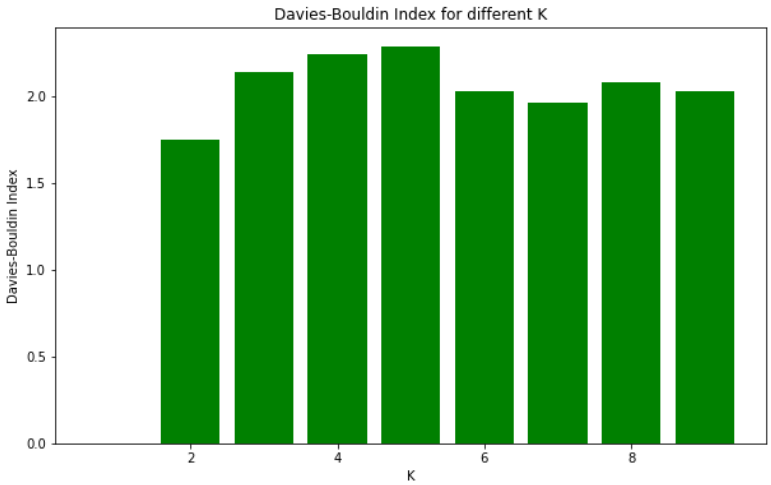

Figura 7.III: Risultati del Davies-Bouldin index.

Innanzitutto, si nota come per un valore pari a 1 non si hanno risultati dell'indice perché la Silhouette per un K pari a 1 non può essere calcolata, il cluster è solamente uno.

La scelta, dunque, viene effettuata sulla base dei valori che hanno un indice più basso, dato che corrisponde a una clusterizzazione più qualitativa, dato che i cluster sono più compatti internamente e separati tra di loro. Perciò sulla base di tutti questi risultati si può constatare come la scelta migliore sia applicare il K-Means con valori di K pari a 2 e 3.

Si parte perciò con l'implementazione del K-Means sulla base di due cluster, impostando come parametro relativo al seed per la clusterizzazione il valore ottimale trovato per questa situazione., e quindi un valore pari a 1, come si nota dalla figura sottostante.

Index	Type	Size	
0	int	1	1
1	int	1	1
2	int	1	19
3	int	1	9
4	int	1	1
5	int	1	4
6	int	1	2
7	int	1	10
8	int	1	10

Figura 7.IV: Tabella che riporta il valore ottimale del seed per ogni K.

Una volta applicata la tecnica di machine learning ci si concentra sull'analisi dei cluster definiti partendo dal conteggio delle osservazioni dei cluster, evidenziandone la dimensionalità.

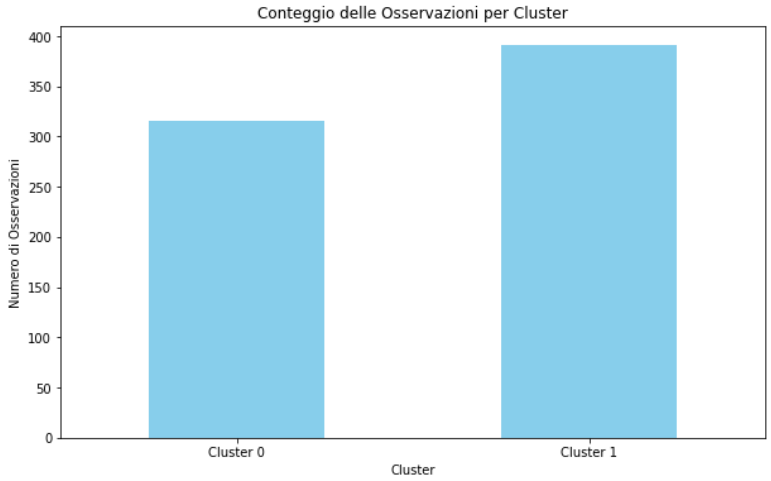

Figura 7.V: Dimensionalità dei cluster con K=2.

Si nota quindi come il cluster più grande, in termini di osservazioni, sia il secondo, ossia il cluster 1, con una dimensionalità di circa 400 osservazioni, mentre il cluster a sinistra presenta intorno alle 320 osservazioni.

Per definire le caratteristiche di questi cluster in modo tale da capire quale sia quello costituito dagli studenti “migliori” in termini di performance e quale quello costituito dai meno performanti, sono state messi a confronto i valori medi delle variabili continue calcolate sulla base delle osservazioni all’interno dei rispettivi cluster.

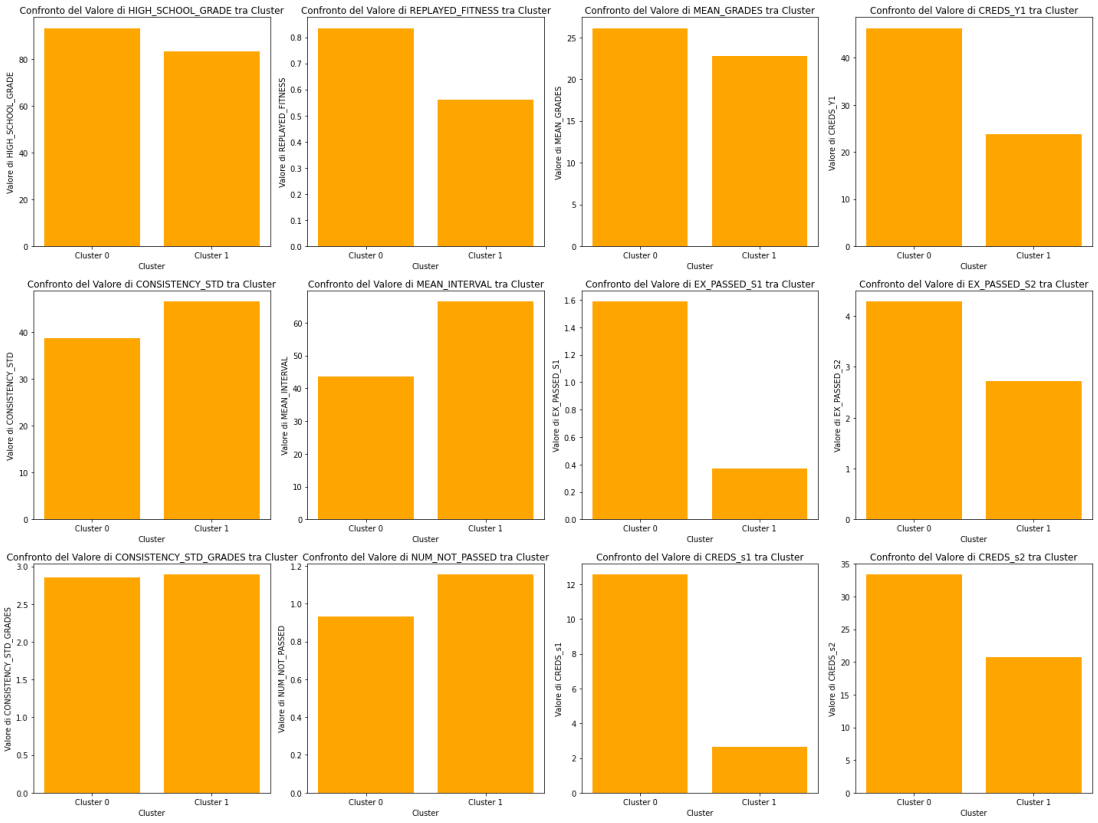

Figura 7.VI: Comparazione valori medi variabili tra cluster con K=2.

Da questo confronto si comprende dunque come il cluster che comprende gli studenti più performanti sia il cluster 0, difatti presenta valori migliori per voto di maturità, fitness calcolata dal conformance checking, media dei voti, crediti dati al primo anno, sia al primo semestre che al secondo, oltre che ovviamente in termini di esami passati in entrambi i semestri. Mentre invece il cluster 1 presenta valori medi più alti per quanto riguarda la consistenza nel dare gli esami e la media dell'intervallo temporale che hanno lasciato trascorre tra il sostenimento di un esame e un altro, oltre che per il numero volte in cui sono stati bocciati ad un esame.

Il fatto che il cluster 0 è composto dagli studenti più performanti è stato poi esplicitato attraverso due matrici di confusione, in cui sono riportati, sia in valori

assoluti, che in percentuale, il conteggio, per ogni cluster, delle osservazioni per ciascuna delle tre categorie di studenti "*Early*", "*One_year_late*", "*Late*".

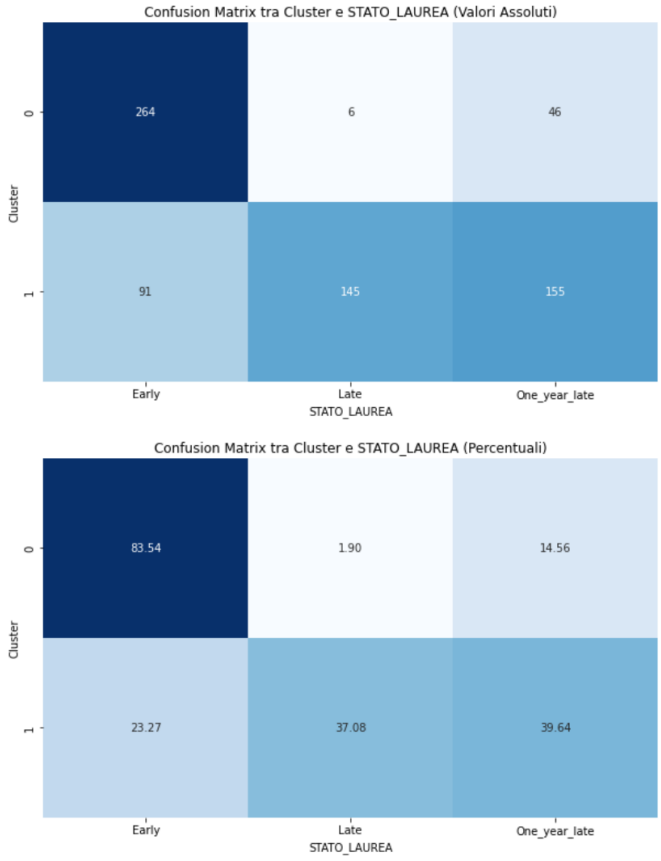

Figura 7.VII: Matrici di confusione Cluster/STATO_LAUREA per K=2.

Una volta dimostrato quale sia il cluster migliore si passa al focus sulla differenza tra pattern comportamentali tramite l'utilizzo dei sottografi.

Index	SUB_83	SUB_56	SUB_48	SUB_62	SUB_7	SUB_18	SUB_23	SUB_9	SUB_57	SUB_60	SUB_100	SUB_43	SUB_26
0	100	82.2696	50.766	29.2009	55.3041	24.5634	100	63.7462	100	100	55.3041	86.0854	48.6179
1	1.30191e-13	17.7304	49.234	70.7991	44.6959	75.4366	9.53344e-14	36.2538	-1.40089e-13	-8.77076e-14	44.6959	13.9146	51.3821

Figura 7.VIII: Dataframe percentuali di rappresentanza sottografi con K=2.

Sulla base del dataframe sopra presentato, che consiste in cento variabili (una per ogni sottografo) e due righe riferite rispettivamente ai cluster, vengono quindi evidenziate le percentuali calcolate sulla base della somma dei valori calcolati per i due centroidi dei cluster.

Da queste percentuali si sono poi considerati e conseguentemente visualizzati, per entrambi i cluster, solamente i primi dieci sottografi ordinati in ordine decrescente sulla base della percentuale di rappresentanza.

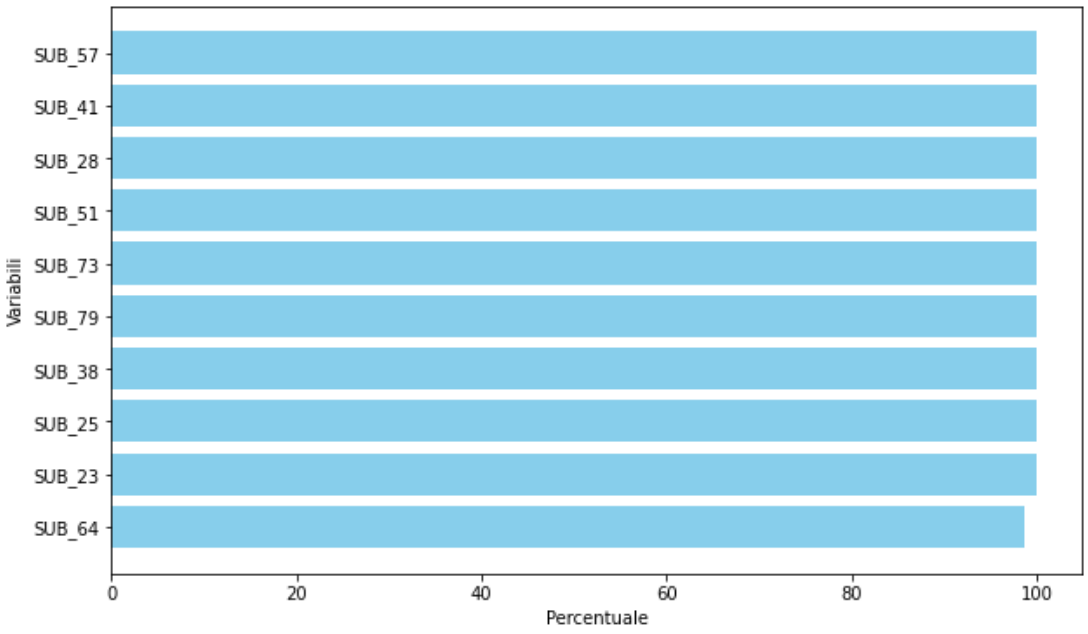

Figura 7.IX: Migliori 10 sottografi per percentuale cluster 0 con K=2.

Come si può notare il cluster 0 ha molti sottografi che presentano una percentuale pari al 100% e che quindi sono presenti solamente nelle osservazioni contenute in esso. Una percentuale pari al 100% per questi sottografi sottolinea l'importanza che questi hanno nel distinguere uno studente che si è laureato in tempi efficienti da uno che invece si è laureato più in ritardo.

Perciò prendendo in esame qualche SUB d'esempio si possono definire i pattern comportamentali più seguiti dagli studenti performanti, e quindi quelli che potrebbero essere seguiti anche da futuri studenti.

Utilizzando la tabella creata per contare le osservazioni per ogni SUB, ci si può concentrare su quelli con più osservazioni, ossia quelli più statisticamente rilevanti.

SUB	Cluster 0	Cluster 1	Totale
SUB_57	9	0	9
SUB_41	10	0	10
SUB_28	22	0	22
SUB_51	7	0	7
SUB_73	6	0	6
SUB_79	6	0	6
SUB_38	15	0	15
SUB_25	17	0	17
SUB_23	23	0	23
SUB_64	60	1	61

Tabella 7.I: Osservazioni Top 10 SUB cluster 0 con K=2.

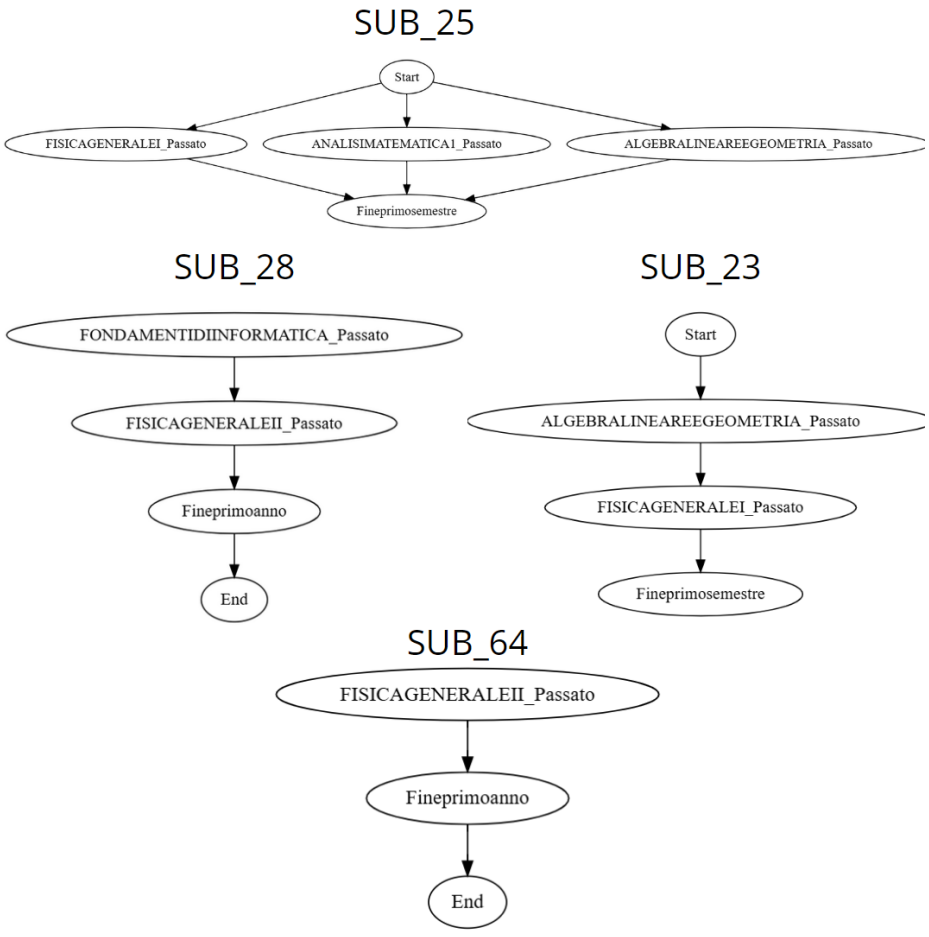

Figura 7.X: SUB più importanti per cluster 0 con K=2.

Analizzando i sottografi con il maggior numero di osservazioni, tra quelli che presentano le percentuali più elevate, è possibile discutere i risultati ottenuti. A partire dal sottografo più ricorrente, SUB_64, si osserva come esso sia in realtà piuttosto semplice, ma fornisca un'informazione di grande rilevanza: l'esame di Fisica Generale II risulta essere particolarmente difficile da superare, forse il più impegnativo tra tutti. Di conseguenza, il fatto che uno studente superi l'esame di Fisica Generale II già nel primo anno del suo percorso di studi, indica che ha affrontato e superato la prova più difficile al primo anno accademico.
Esame di Fisica Generale II che è presente anche nel SUB_28, che conteggia 22 ricorrenze, tutte nel cluster 0, all'interno del quale è inoltre preceduto da un altro esame, "Fondamenti di Informatica", il quale però è passato più frequentemente.
Il sottografo SUB_23 evidenzia invece come gli studenti che lo presentano abbiano passato due dei tre esami del primo semestre prima della fine di quest'ultimo, mentre infine il SUB_25 presenta l'inizio migliore che uno studente possa intraprendere, dato che i 17 studenti che presentano questo sottografo hanno passato, parallelamente e prima della fine del primo semestre, tutti e tre gli esami disponibili.

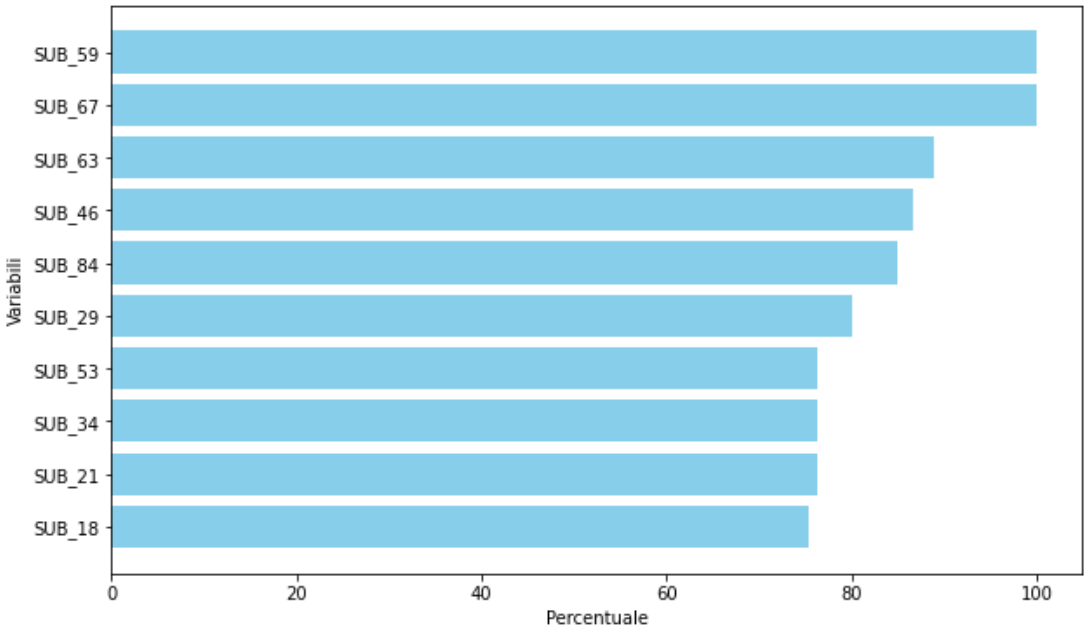

Figura 7.XI: Migliori 10 sottografi per percentuale cluster 1 con K=2.

Mentre per quanto riguarda il cluster 1, ossia quello degli studenti che hanno quindi avuto più problematiche in termini di tempo di laurea impiegato, si nota come i SUB che sono presenti esclusivamente nelle osservazioni appartenenti a questo cluster sono due, SUB_59 e SUB_67.

Tuttavia, anche quelle successive presentano percentuali significative, inizialmente poco inferiori al 90%, fino ad arrivare alle ultime a cui corrisponde una percentuale intorno al 75%, quindi comunque importante.

Anche del secondo cluster sono state calcolate le occorrenze dei dieci SUB più importanti:

SUB	Cluster 0	Cluster 1	Totale
SUB_59	0	9	9
SUB_67	0	16	16
SUB_63	1	10	11
SUB_46	1	8	9
SUB_84	1	7	8
SUB_29	6	30	36
SUB_53	2	8	10
SUB_34	3	12	15
SUB_21	6	24	30
SUB_18	15	57	72

Tabella 7.II: Osservazioni Top 10 SUB cluster 1 con K=2.

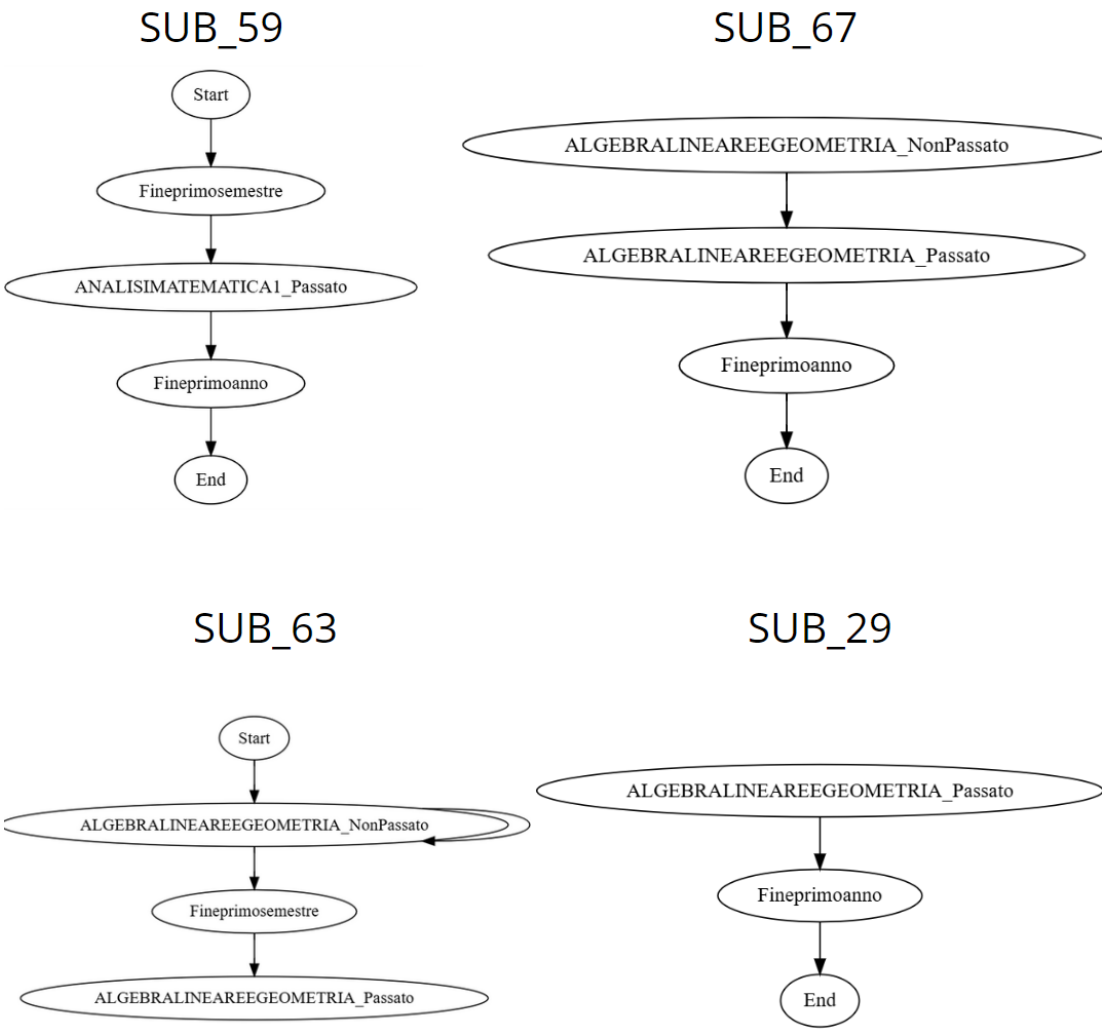

Figura 7.XII: SUB più importanti per cluster 1 con K=2.

Si parte dunque dai primi due SUB, 59 e 67, ossia gli unici due che hanno una percentuale di osservazioni del 100% nel cluster 1. Il primo rappresenta il fatto che, gli studenti che presentano questo pattern, hanno dato, nell'intero primo anno accademico, solamente "Analisi Matematica 1"; mentre il secondo dà un'informazione meno "grave", dato che non spiega cosa è successo nel primo semestre, ma solamente che, nel secondo semestre, è stato bocciato una volta, prima di passarlo, ad "Algebra Lineare e Geometria" per poi finire l'anno. Questa

informazione, seppur meno negativa della prima, ci dice comunque che lo studente ha dovuto sostenere almeno due volte nel secondo semestre un esame del primo. Passando poi ad analizzare il primo dei sottografi che non è presente, per la sua totalità, nelle osservazioni del cluster 1, ossia il SUB_63, si trovano informazioni altrettanto negative, dato che evidenzia come lo studente presenti un loop di bocciature nel primo semestre per "Algebra Lineare e Geometria", per poi passarlo nel secondo semestre, a testimonianza di come lo studente abbia potuto bloccare l'avanzamento del percorso accademico studiando troppo, e male, per un solo esame. Il SUB_29, infine, presenta solamente l'esame di "Algebra Lineare e Geometria" che si collega alla fine del primo anno, presupponendo dunque che sia stato dato al secondo semestre. Tra i quattro sottografi presentati è quello meno significativo, testimoniato dal fatto che divide comunque efficaciemente tra gli studenti dei cluster (6 per cluster 0 e 30 per cluster 1), ma non ottimamente come quelli analizzati precedentemente.

Dopo aver analizzato il clustering per K=2, si passa all'analisi per K=3, dato che come si era notato dall'utilizzo di misure intra e inter-cluster, era l'altro numero di cluster ottimale insieme a due.

Il seed, prendendo i risultati della Figura VII.4, sarà impostato a 19, definendo i seguenti risultati, partendo dalla dimensionalità dei cluster:

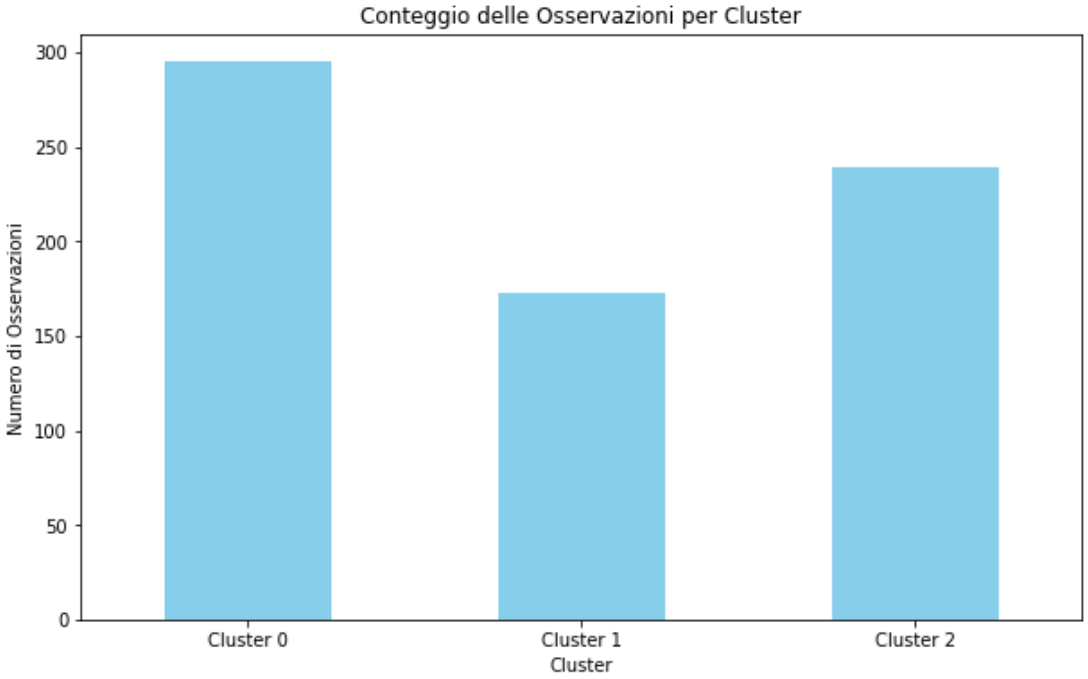

Figura 7.XIII: Dimensionalità dei cluster con K=3.

Il cluster più grande in termini di osservazioni che lo costituiscono è il primo, ossia il cluster 0, seguito da cluster 2 e cluster 1.

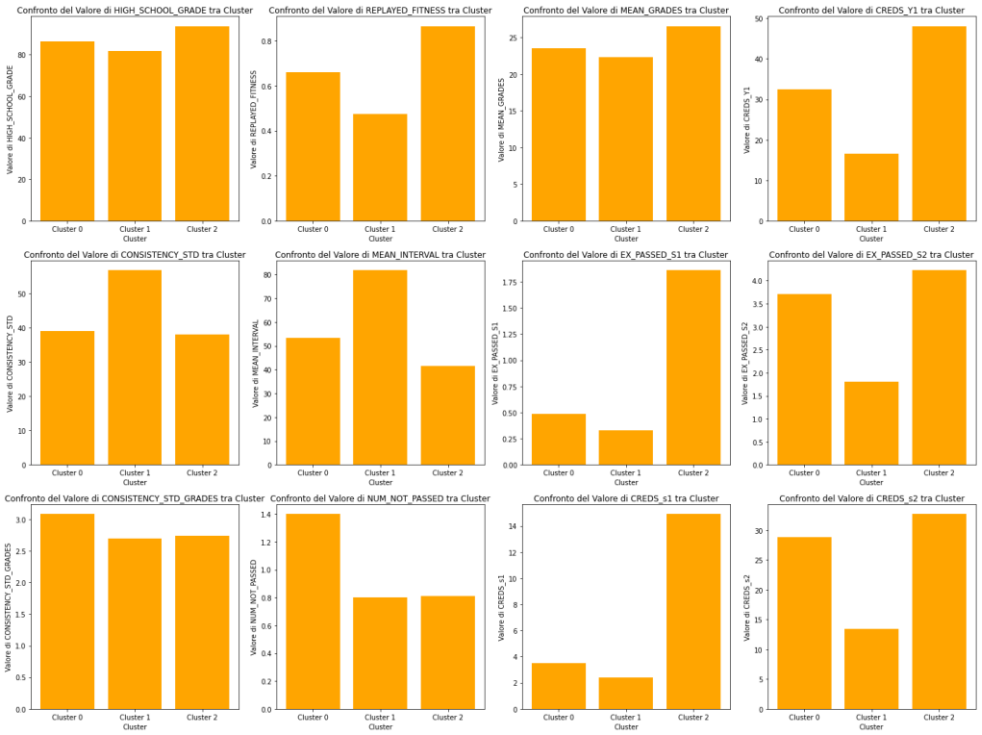

Figura 7.XIV: Comparazione valori medi variabili tra cluster con K=3.

Per definire quale cluster contenga gli studenti migliori si parte quindi con la comparazione, tra centroidi, dei valori assunti da questi per ogni variabile continua. Da questa visualizzazione si nota chiaramente come il cluster 2 sia quello che è composto dalla maggior parte degli studenti con valori positivi in termini di performance. Infatti, per quanto riguarda gli esami passati al primo anno e i relativi crediti ottenuti, sia nel primo che nel secondo semestre, riporta dei valori migliori, soprattutto per quanto riguarda il primo semestre. Inoltre, in media, hanno valori migliori per quanto riguarda il voto con cui sono usciti dalle superiori, media voto e valori di fitness derivanti dall'applicazione del conformance checking.
Guardando le stesse variabili possiamo definire anche il cluster che contiene più studenti appartenenti al gruppo "*Late*", ossia il cluster 1, che riporta dunque i valori medi minori. Si ritrova la stessa informazione anche per le due variabili che definiscono gli intervalli temporali tra un esame e un altro, quali "CONSISTENCY_STD" e "MEAN_INTERVAL". Le variabili invece per cui gli studenti del cluster 0 presentano valori migliori sono invece "CONSISTENCY_STD_GRADES" e "NUM_NOT_PASSED", probabilmente perché sono studenti che si prendono più tempo, ricercando dunque una maggiore costanza di voti, che, come definisce la votazione media, è comunque buona.
Per rafforzare le informazioni derivanti da questa comparazione, rappresentiamo, come nel primo clustering, le matrici di confusione che definiscono, per ciascun cluster, il numero di studenti per ogni categoria.

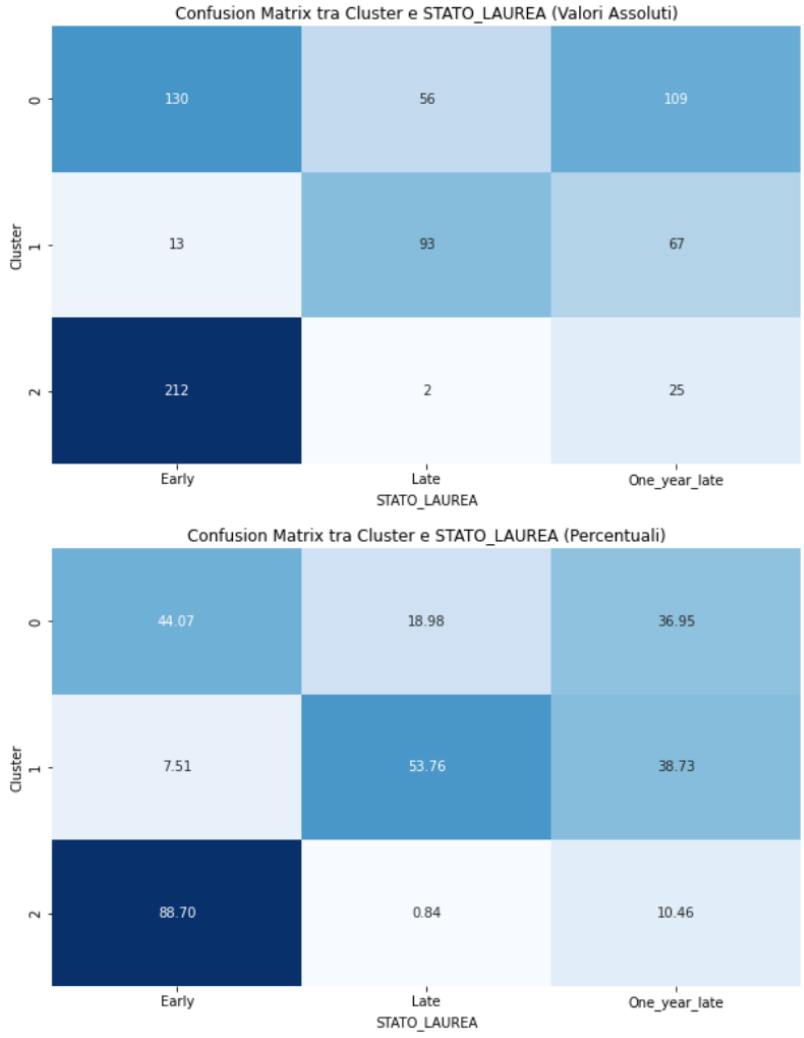

Figura 7.XV: Matrici di confusione Cluster/STATO_LAUREA per K=3.

Si evince infatti come nel Cluster 2 l'88.70% degli studenti si sia laureato in tempo, mentre la presenza degli studenti "*Late*" è pari solamente al 0.84%, che corrisponde a 2 osservazioni. Situazione contraria nel cluster 1, costituito per il 92.49% da studenti appartenenti ai gruppi "*Late*" e "*One_year_late*". Il cluster 0 invece risulta essere il più eterogeneo tra i tre.

Per il clustering con K=3 definiamo anche la matrice di distanza tra centroidi, tramite cui si può notare che i cluster 2 e 1 sono distanti tra di loro, a testimonianza di una

netta separazione tra questi; mentre il cluster 0, essendo quello intermedio presenta distanze non così elevate con entrambi i cluster più “estremi”.

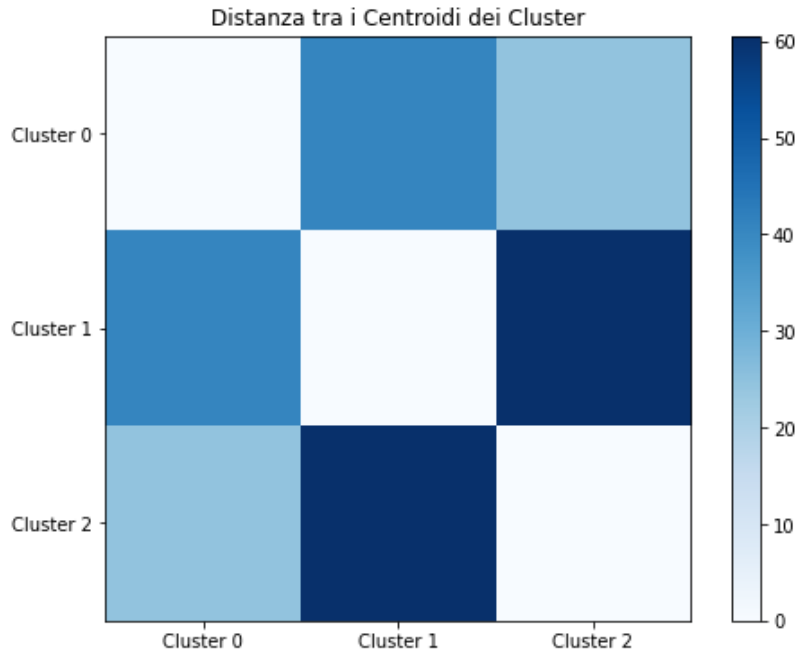

Figura 7.XVI: Distanza tra centroidi dei cluster con K=3.

Sulla base di un dataframe strutturato similarmente a quello presentato in Figura VII.8 per K=2 (con l’unica differenza che abbiamo tre righe su cui calcolare la percentuale in questo caso), vengono prima filtrati anche in questo caso i sottografi più rilevanti statisticamente per poi definirne i dieci migliori per ciascun cluster.

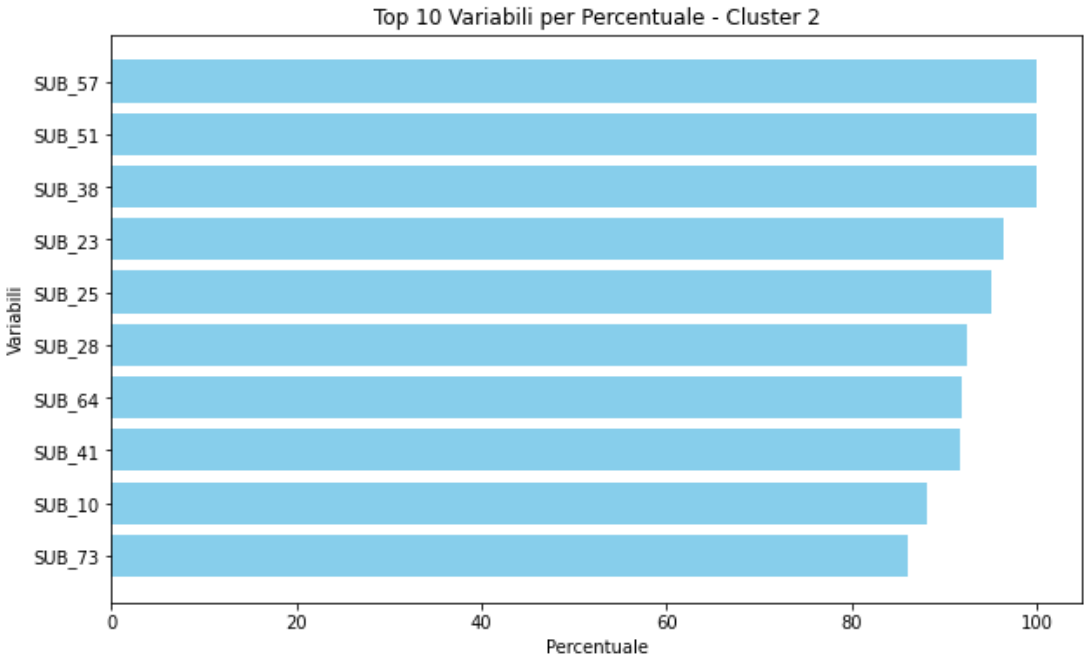

Figura 7.XVII: Migliori 10 sottografi per percentuale cluster 2 con K=3.

Dunque, il cluster 2 presenta tre sottografi che sono presenti solamente tra gli studenti appartenenti a questo gruppo, e perciò fortemente distintivi.

In generale però si nota subito come il fatto che i cluster ora sono tre diminuisca il numero di pattern comportamentali che sono specifici per un singolo cluster rispetto alla situazione in cui il K-Means è stato applicato con K=2. Difatti nel caso precedente per il cluster migliore i sottografi che presentavano una percentuale pari al 100% erano nove, con il decimo che presentava comunque una percentuale molto vicina al 100%.

Anche in questo caso si sono calcolate le occorrenze di ognuno tra i dieci SUB più rilevanti per ciascun cluster.

SUB	Cluster 0	Cluster 1	Cluster 2	Totale
SUB_57	0	0	9	9
SUB_51	0	0	7	7
SUB_38	0	0	15	15
SUB_23	1	0	22	23
SUB_25	1	0	16	17
SUB_28	2	0	20	22
SUB_64	6	0	55	61
SUB_41	1	0	9	10
SUB_10	8	0	48	56
SUB_73	1	0	5	6

Tabella 7.III: Osservazioni Top 10 SUB cluster 2 con K=3.

Confrontando innanzitutto questi risultati con quelli del cluster migliore risultante dal primo clustering, ritroviamo quasi tutti i sottografi, a parte SUB_73 e SUB_10, che è in realtà un sottoinsieme del SUB_25 e che quindi presenta solamente “Analisi Matematica 1-Passato” e “Algebra Lineare e Geometria-Passato” in parallelo.

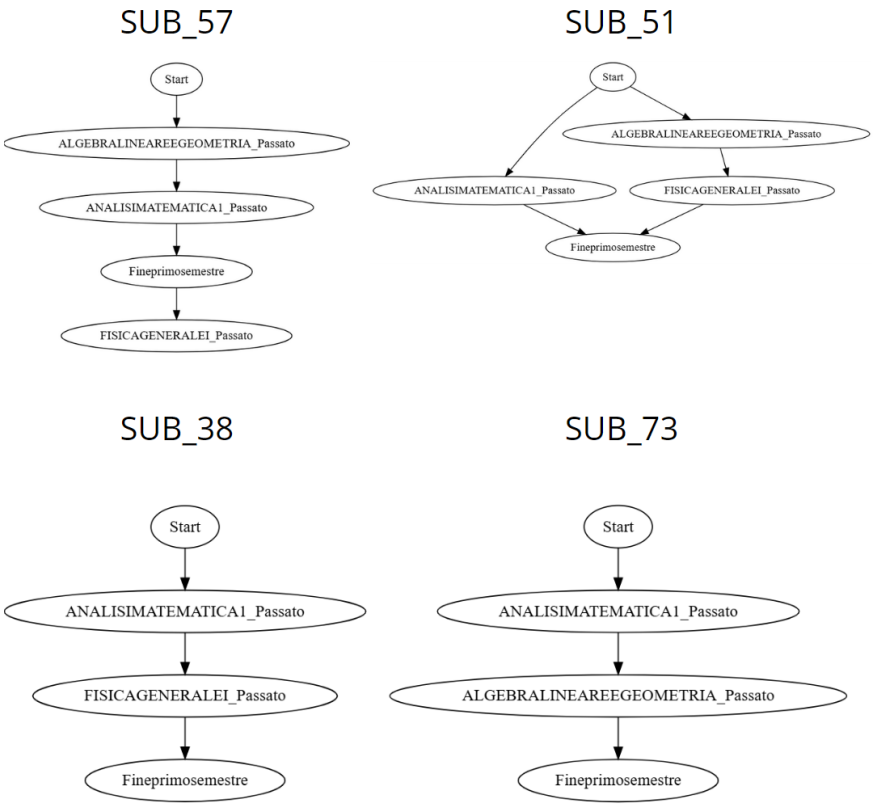

Figura 7.XVIII: SUB più importanti per cluster 2 con K=3.

Si può notare dunque che i SUB_38 e SUB_73 sono molto simili al SUB_23 della Figura VII.10. Difatti che entrambi, come SUB_23, presentano due esami passati al primo semestre, con l'unica differenza che nel SUB_38, al posto di "Algebra Lineare e Geometria" è presente "Analisi Matematica 1", mentre nel SUB_73 è presente "Analisi Matematica 1" al posto di "Fisica Generale I", con "Algebra Lineare e Geometria" che in questo caso viene data come seconda materia.

Una certa somiglianza vi è anche tra il SUB_51 e il SUB_25, dato che il primo, invece di essere costituito da tutti e tre gli esami disponibili al primo semestre passati in parallelo, è costituito da sempre tutti e tre gli esami, ma in parallelo troviamo solamente "Analisi Matematica 1" e "Fisica Generale I", con "Algebra Lineare e

Geometria che è stato passato leggermente prima di questi, dato che è in parallelo solamente con “Analisi Matematica 1”, ma viene prima di “Fisica Generale I”.

Infine, l’ultimo sottografo della figura, il 57, evidenzia come anche in questo caso, gli esami del primo semestre risultano tutti passati, di cui solamente “Fisica Generale I” è stato dato nel secondo semestre.

Passando dal grafico che presenta gli studenti più performanti a quello contenente gli studenti meno performanti, si mostrano i risultati dei dieci sottografi più rappresentativi del cluster, che risultano essere i seguenti:

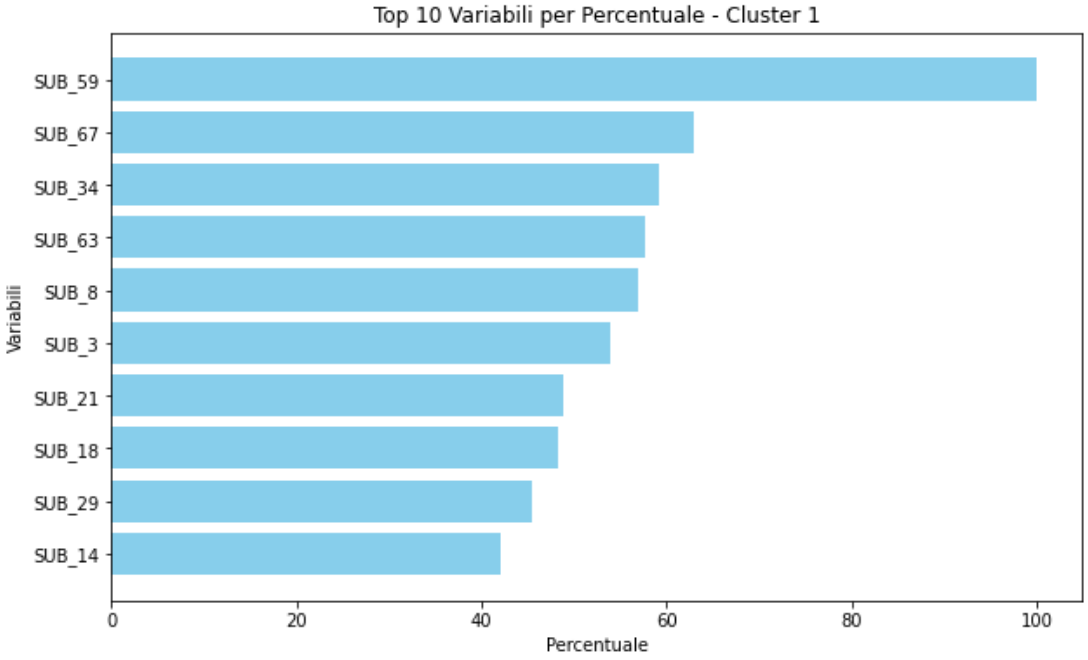

Figura 7.XIX: Migliori 10 sottografi per percentuale cluster 1 con K=3.

I risultati confermano immediatamente come con l’aumento dei cluster diminuiscono i sottografi con una percentuale del 100%. Difatti in questo caso per il cluster peggiore si trova solamente il SUB_59 con 100% di osservazioni all’interno di un singolo cluster, mentre già quelli successivi denotano un forte decremento nella percentuale, con il secondo che presenta poco più del 60%.

Anche in questo caso, come era prevedibile, ben sette tra i primi dieci sottografi sono comuni con quelli per il cluster peggiore con K=2, dato che a differire sono solamente il SUB_8, SUB_3 e SUB_14.

SUB	Cluster 0	Cluster 1	Cluster 2	Totale
SUB_59	0	9	0	9
SUB_67	8	8	0	16
SUB_34	7	7	1	15
SUB_63	5	5	1	11
SUB_8	39	58	29	126
SUB_3	102	112	50	264
SUB_21	16	11	3	30
SUB_18	39	26	7	72
SUB_29	22	12	2	36
SUB_14	21	10	2	33

Tabella 7.IV: Osservazioni Top 10 SUB cluster 1 con K=3.

Si può vedere anche dalla tabella come le osservazioni in questo caso siano più distribuite tra i vari cluster, sottintendendo quindi che trovare pattern comportamentali tipici degli studenti meno performanti risulti più difficile.
Dato che molti sono comuni col cluster 1 del clustering con due variabili, si è andati ad analizzare solamente i sottografi che si sono presentati solamente per questo cluster, oltre al SUB_59, anche se già rappresentato nella Figura 7.XI.

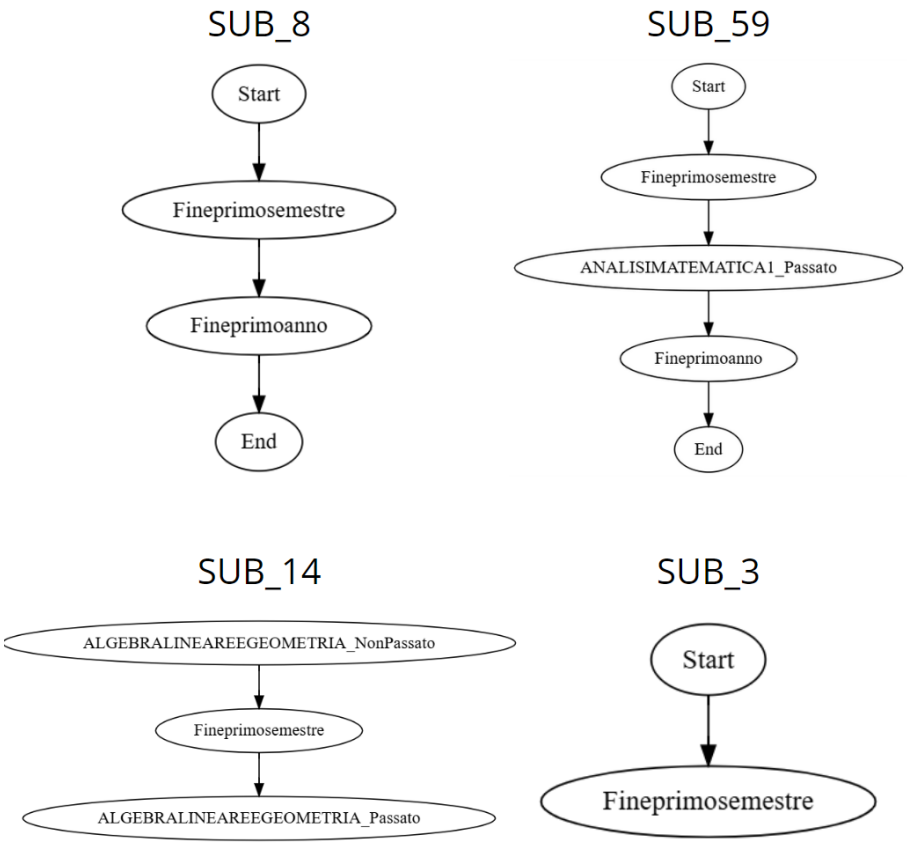

Figura 7.XX: SUB più importanti per cluster 1 con K=3.

Partendo dall'analisi del primo, si vede immediatamente come questo sia l'inizio del percorso accademico peggiore con cui uno studente possa iniziare, dato che consiste nel non aver passato, ma nemmeno provato, nessun esame ed è infatti presente solamente nel percorso di 29 studenti su 239 per il cluster 2, mentre sono precisamente il doppio per il cluster 1, nonostante la numerosità di questo sia molto più bassa del cluster 2, ossia 173 studenti.

Il SUB_3 consiste nella prima parte del SUB_8, ossia conteggia tutti quegli studenti che non hanno sostenuto nessun esame nel primo semestre del primo anno accademico; è quindi comunque un pattern negativo, ma non così grave come il primo, dato che gli studenti potrebbero comunque aver recuperato nel secondo.

Infine, il SUB_14 è molto simile al SUB_63, con la differenza relativa al fatto che in questo caso non vi è un *loop* per "Algebra Lineare e Geometria-NonPassato".

Concludendo poi l'analisi per il clustering con K=3, si presentano i dieci migliori sottografi in termini di percentuale di rappresentanza per il cluster intermedio, ossia il cluster 0.

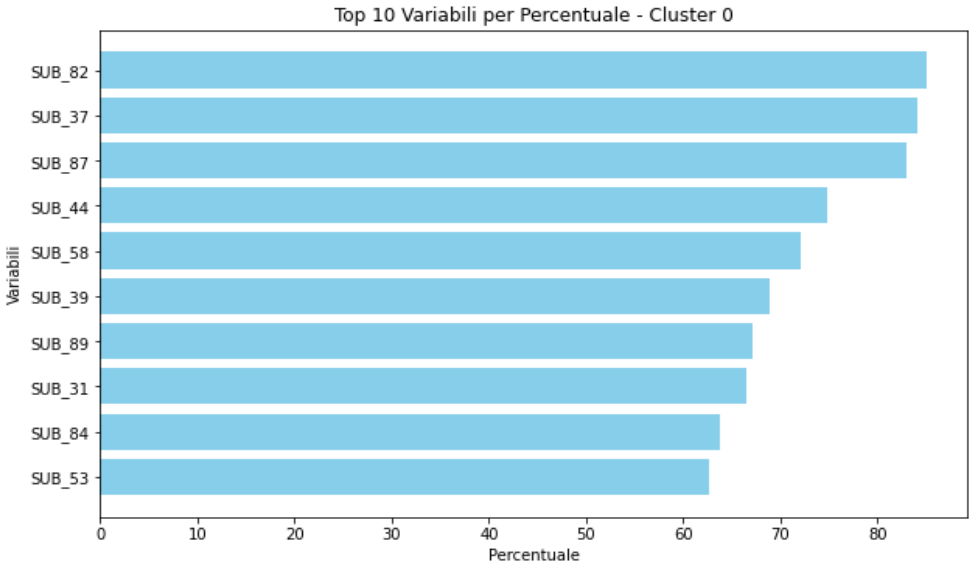

Figura 7.XXI: Migliori 10 sottografi per percentuale cluster 0 con K=3.

In questo caso, a testimonianza di come questo sia un cluster intermedio, non vi è nessun sottografo che riporta una percentuale pari al 100%, bensì il più rappresentativo del cluster ha una percentuale intorno all'85%. Tuttavia, a differenza del cluster 0, qua anche gli ultimi sottografi tra quelli visualizzati presentano delle percentuali buone, tanto che il decimo ha una percentuale superiore al 60%.

I sottografi presenti in quest'ultimo grafico sono quasi tutti nuovi, a parte SUB_84 e SUB_53 che sono presenti anche per il Cluster 1, ossia quello relativo agli studenti meno performanti, nel clustering con K=2(Figura 7.X). Questo potrebbe essere dato dal fatto che, a differenza di coloro che sono gli studenti meno performanti, gli

studenti che hanno impiegato meno a laurearsi presentano dei pattern che sono tipici di quella categoria e perciò meno presenti nelle osservazioni di altri cluster.

SUB	Cluster 0	Cluster 1	Cluster 2	Totale
SUB_82	7	0	1	8
SUB_37	13	0	2	15
SUB_87	6	0	1	7
SUB_44	11	0	3	14
SUB_58	12	2	1	15
SUB_39	12	1	3	16
SUB_89	6	1	1	8
SUB_31	16	4	1	21
SUB_84	6	2	0	8
SUB_53	7	1	2	10

Tabella 7.V: Osservazioni Top 10 SUB cluster 0 con K=3.

Dato che in questo caso le percentuali dei sottografi sono molto compatte, il criterio per la scelta dei sottografi da visualizzare come esempi cluster è basato sul numero di osservazioni di quel sottografo; verranno dunque visualizzati quelli più numerosi.

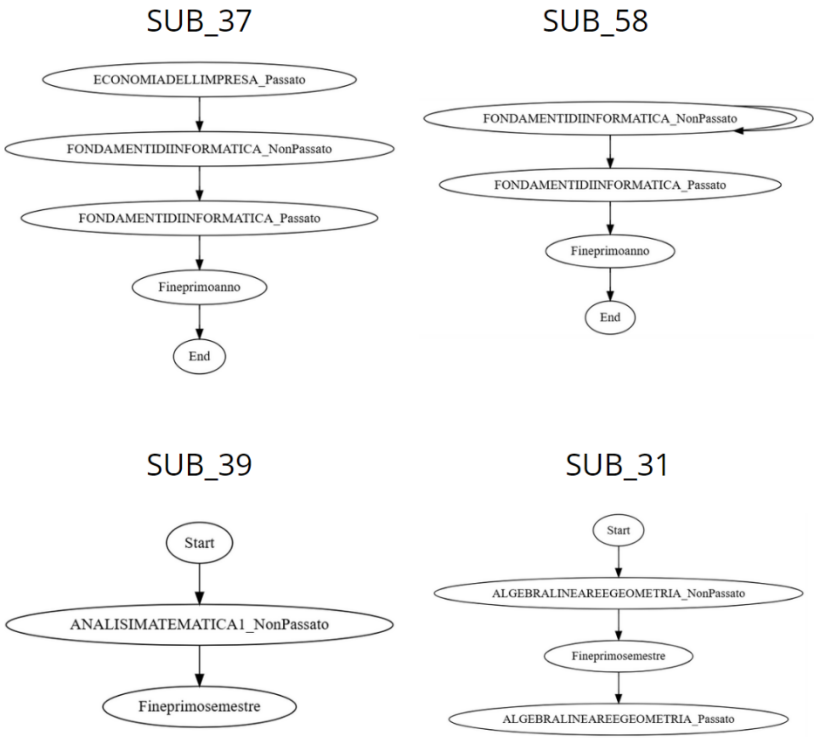

Figura 7.XXII: SUB più importanti per cluster 0 con K=3.

I risultai esplicitati, ci confermano come questo sia il cluster che contiene gli studenti intermedi in termini di performance, dato che, anche gli stessi sottografi che risultano essere tra i più rappresentativi, non danno informazioni polarizzanti, bensì sono tutti pattern che ci danno informazioni che non possono essere utilizzare per eseguire una netta distinzione tra gli studenti.

Come conclusione dell'analisi si sono quindi definiti i modelli di processo per ciascun cluster, scegliendo il *threshold* sulla base dei valori delle metriche calcolati dall'inductive miner.

Il primo modello costruito è quello relativo al cluster 2, ossia quello contenente gli studenti che presentano le performance migliori.

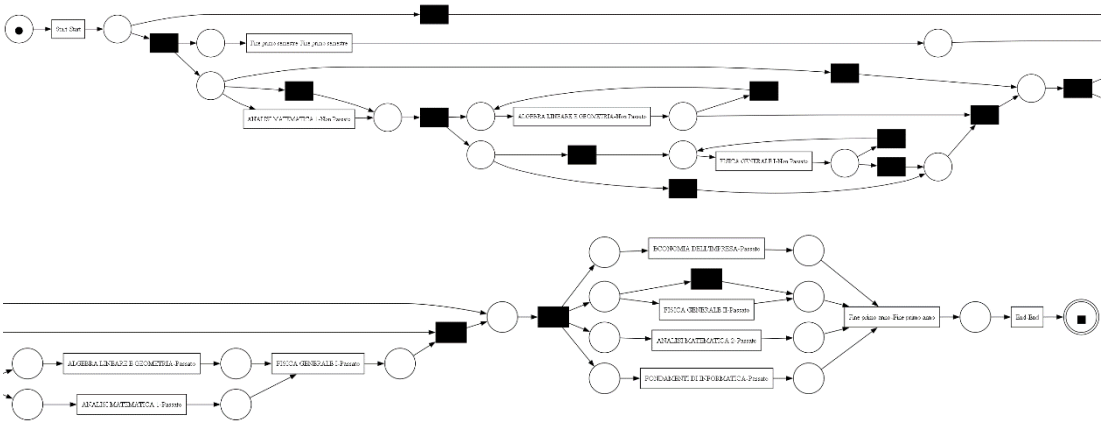

Figura 7.XXIII: Rete di Petri inductive miner con K=0.6 per cluster 2.

Dal confronto di questo modello con il modello degli studenti migliori basandosi solamente sulla variabile categorica riferita al tempo di laurea impiegato (Figura 4.II), si può notare come, nonostante un valore di K più basso come 0.6 e dunque con un filtraggio meno stringente, il modello risulta decisamente più pulito e chiaro.
Inoltre, il percorso risultante rispecchia molto di più quello che potrebbe essere il percorso di studenti che hanno iniziato l'anno al meglio, tanto che si creano dei cicli basati sulle bocciature agli esami solamente per tre materie ("Analisi Matematica 1, "Fisica Generale I" e "Algebra Lineare e Geometria") e non vi sono percorsi che presuppongono che nessun esame sia stato dato, a differenza del modello per gli studenti del gruppo "*Early*".
Successivamente è stato utilizzato l'inductive miner per costruire il modello di processo per gli studenti appartenenti al cluster intermedio, cluster 0, per poterlo confrontare con quello degli studenti "*One_year_late*".

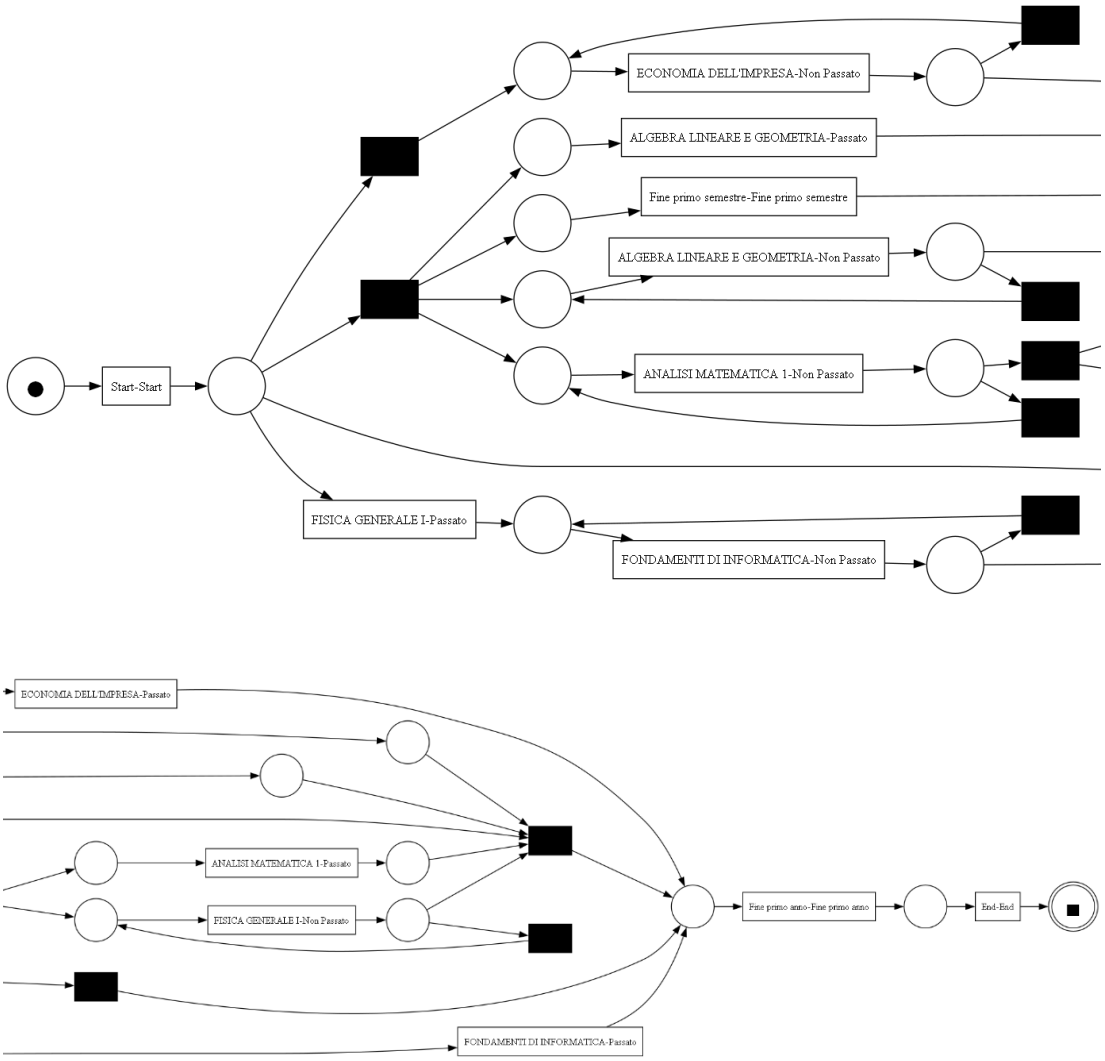

Figura 7.XXIV:Rete di Petri inductive miner con K=0.8 per cluster 0.

In questo caso il valore della soglia che presentava le metriche migliori era pari a 0.8, quindi il modello di processo costruito è leggermente più filtrato rispetto al modello per gli studenti "One_year_late", il quale era stato costruito sulla base di un valore K uguale a 0.7 (Figura 4.III). Nonostante ciò, il modello in Figura 7.XXIV risulta molto più chiaro rispetto al precedente.

Per concludere, è stato costruito il modello del cluster che verrà messo a confronto con quello degli studenti appartenenti alla categoria "*Late*", ossia il cluster 1.

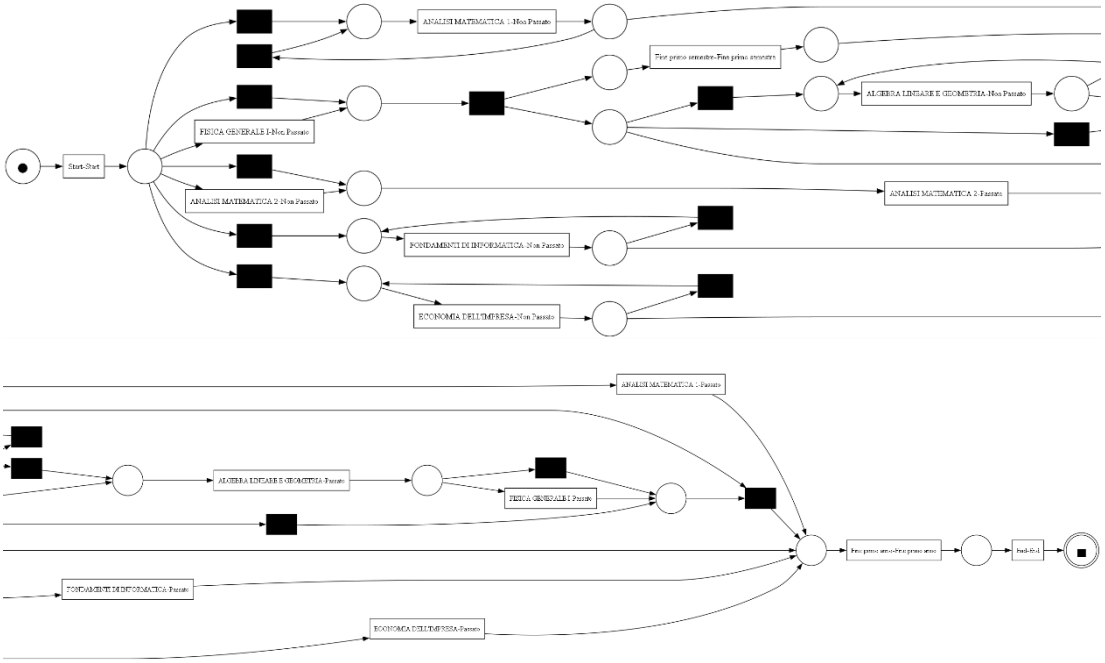

Figura 7.XXV: Rete di Petri inductive miner con K=0.6 per cluster 1.

In questo caso la differenze tra quest'ultimo modello e quello a Figura 4.IV si notano meno, con l'unica differenza significativa legata al fatto che in quest'ultimo modello viene riportato il superamento dell'esame "Fisica Generale I", probabilmente dettato anche dal minor filtraggio applicato.

Capitolo 8: CONCLUSIONI

L'analisi sui processi educativi analizzati è stata effettuata con l'obiettivo di fornire informazioni rilevanti per migliorare la capacità da parte dell'università di intervenire per aiutare gli studenti e di conseguenza migliorare l'esperienza universitaria di quest'ultimi.

Gli studi precedentemente effettuati hanno dunque illustrato i modelli di comportamento comuni tra gli studenti nel processo relativo al sostenimento degli esami, sia per quanto riguarda l'eventuale superamento dell'esame che per quanto riguarda la preparazione degli stessi.

Questo perché l'obiettivo ultimo è proprio quello di analizzare quali pattern sono più comuni tra gli studenti più performanti e quelli meno, in modo tale da poter nel primo caso definire dei percorsi consigliati per raggiungere la laurea nelle modalità migliori e intercettare ed intervenire il più presto possibile per aiutare gli studenti che presentano comportamenti individuati nel secondo caso.

Inoltre, tramite le statistiche presentate nel capitolo 4 sono stati individuati gli esami più sostenuti e con più bocciature, individuando esami quali ad esempio "Analisi Matematica 1", "Algebra Lineare e Geometria" e "Fisica Generali II" come quelli che necessitano una quantità di studio maggiore per essere passati.

Dunque, una volta costruiti e messi a confronto i percorsi tipici degli studenti appartenenti alle tre diverse categorie si è entrati nella parte più significativa in termini di risultati, ossia quella relativa alla costruzione dei grafi con conseguente

estrazione dei sottografi che sono stati utilizzati per il clustering, tramite l'applicazione del K-Means.

Per l'applicazione dell'algoritmo sono stati scelti i valori di K migliori per determinare il numero di cluster sulla base dell'*elbow method,* ottenendo quindi dei risultati diversi, per quanto riguarda comportamenti e modelli di processo costruiti tramite inductive e heuristic tra i diversi cluster ottenuti.

In particolare, i modelli di processo costruiti sulla base dei cluster sono stati poi confrontati con quelli costruiti per le categorie degli studenti definite inizialmente ("*Early*", "*One_year_late*" e "*Late*"), ottenendo dei risultati molto importanti a favore dell'ottima segmentazione effettuata dal K-Means.

In conclusione, dunque, l'analisi ha sottolineato come vi siano degli esami cruciali per l'andamento del percorso che necessitano priorità e maggiore impegno nello studio, oltre al fatto che iniziare al meglio il primo anno, ossia l'inizio del percorso accademico dello studente, risulti particolarmente importante per il proseguo dello stesso.

Si ritiene perciò che questo studio sia molto importante come già detto per entrambe le parti interessate, sia studenti che università, per cercare di migliorare il più possibile l'esperienza accademica del singolo studente, che andrà ad apportare benefici anche all'università stessa in termini di prestigio.

L'analisi lascia comunque spunti per eventuali progetti futuri che potrebbero consistere nell'adottare altre metodologie per definire i pattern comportamentali, come ad esempio l'utilizzo degli alberi di processo, i quali mostrano le sequenze di attività in modo chiaro e strutturato, evidenziando le dipendenze tra attività, parallelismi, loop e decisioni.

Inoltre, si potrebbe anche modificare l'intervallo di tempo utilizzato per definire la concorrenza tra due esami paralleli, aumentando o diminuendo il periodo pari ad un mese utilizzato per la corrente analisi.

BIBLIOGRAFIA

1. WIL VAN DER AALS, Process mining: Data science in action (2nd ed.). Springer, 2016.
2. WIL VAN DER AALST, Process Mining Manifesto, 2016.
3. DESEL, J., & REISIG, W., The concepts of Petri nets. Software and Systems Modeling, 2015.
4. Slides: Wil van der Aalst (www.vdaalst.com).
5. BAR, J., CAVERLEE, J., & ZHANG, Z., *Process Mining, Discovery, and Integration using Distance Measures*. Georgia Institute of Technology Repository, 2006.
6. K-means Clustering Algorithm: Applications, Types, & How Does It Work?, Simplilearn, https://www.simplilearn.com/tutorials/machine-learning-tutorial/k-means-clustering-algorithm#:~:text=K%2DMeans%20clustering%20is%20an,objects%20belonging%20to%20another%20cluster.
7. https://fluxicon.com/disco/.
8. Documentazione pm4py:
9. Documentazione network: https://networkx.org/.
10. Documentazione scikit-learn: https://scikit-learn.org/stable/.