



UNIVERSITÀ
DEGLI STUDI
FIRENZE

Scuola di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica

Tesi di Laurea

ANALISI CON TECNICHE DI DATA MINING
SU DATI RELATIVI A INSEGNAMENTI E
STUDENTI DEL CORSO DI LAUREA IN
INFORMATICA

MINING AND ANALYSIS OF COURSES AND
STUDENTS DATA CONCERNING THE
COMPUTER SCIENCE DEGREE

SIMONE CIPRIANI

Relatore: *Donatella Merlini*

Anno Accademico 2017-2018

Simone Cipriani: *Analisi con tecniche di Data Mining su dati relativi a insegnamenti e studenti del Corso di Laurea in Informatica*, Corso di Laurea in Informatica, © Anno Accademico 2017-2018

INDICE

1	INTRODUZIONE	11
2	DATI INIZIALI	13
2.1	Carriera degli Studenti	13
2.1.1	Formato e Rappresentazione	14
2.1.2	Mole di dati	15
2.2	Valutazione degli Insegnamenti	16
2.2.1	Formato e Rappresentazione	16
2.2.2	Mole di dati	18
2.3	Conclusioni dell'Analisi dei Dati Iniziali	18
3	TECHNOLOGY STACK	21
3.1	Strumenti Necessari	21
3.2	Data Base Management System: MongoDB	22
3.2.1	La scelta di MongoDB	22
3.2.2	Caratteristiche di MongoDB	23
3.3	Interazione col DBMS: Python	23
3.3.1	API per MongoDB: pymongo	24
3.4	Organizzazione: GNU Make	25
3.5	Algoritmi di Data Mining: Weka	26
3.6	Tecniche di Visualizzazione	27
3.6.1	Linguaggio R	27
3.6.2	Fogli di Calcolo	28
3.7	Redazione di questo documento	28
3.8	Conclusioni	29
4	DATA UNDERSTANDING PRELIMINARE	31
4.1	Introduzione al Data Understanding	31
4.1.1	Metodi e Strumenti	32
4.2	Data Set: Produttività degli Studenti	32
4.2.1	Prestazioni Generali degli Studenti	33
4.2.2	Prestazioni degli Studenti su Gruppi di Esami	39
4.2.3	Conclusioni	53
4.3	Data Set: Valutazioni degli Insegnamenti	53
4.4	Scelta delle Tecniche da Impiegare	54
5	PREPROCESSING	57
5.1	Preparazione dell'Ambiente di Lavoro	57

5.1.1	Ottenere gli strumenti necessari	57
5.1.2	Inizializzazione di un server MongoDB	58
5.1.3	Importazione dei Dati Grezzi	59
5.2	Aggregazione per Anno Accademico	60
5.2.1	Produttività degli Studenti	61
5.2.2	Valutazione degli Insegnamenti	62
5.3	Join dei due insiemi di dati	63
5.3.1	Join con valutazioni estese e attributi continui	63
5.3.2	Join con valutazioni aggregate e attributi continui	66
5.3.3	Attributi discreti	67
5.4	Sequenze ordinate di esami superati	67
5.5	Estrazione dei data set preprocessati	69
6	DATA UNDERSTANDING PER CLASSI	71
6.1	Valutazione dei Corsi	71
6.1.1	Panoramica Generale sulle Valutazioni dei Corsi	71
6.1.2	Dettaglio sulla Percentuale di Valutazioni Sufficienti	72
6.2	Produttività degli Studenti	72
6.2.1	Panoramica generale sulla Produttività degli Studenti	74
6.2.2	Relazione fra Test di Ingresso, Voto Medio e Ritardo	74
6.2.3	Percentuale di Studenti Laureati	77
7	CLUSTER ANALYSIS	79
7.1	Introduzione alla Cluster Analysis	79
7.2	Algoritmi di Clustering	80
7.2.1	Algoritmo K-Means	80
7.2.2	Algoritmo DBSCAN	81
7.3	Clustering sulle Valutazioni dei Corsi	83
7.3.1	Lancio di K-Means	83
7.4	Clustering sulla join dei due data set	86
7.4.1	Lancio di K-Means	86
7.4.2	Analisi dei risultati di K-Means	88
7.4.3	Tentativo di utilizzo di DBSCAN	97
8	ANALISI ASSOCIATIVA	101
8.1	Introduzione all'analisi associativa	101
8.1.1	Regole Associative	101
8.1.2	L'algoritmo Apriori di Weka	102
8.2	Apriori su dataset aggregato	104
8.2.1	Focus sul corso	105
8.2.2	Focus sul docente	109

8.2.3	Focus sulla Deviazione Standard	112
9	PATTERN SEQUENZIALI FREQUENTI	115
9.1	Algoritmo GSP	115
9.2	Scopo dell'Analisi	116
9.2.1	Esempio di Analisi di una Sequenza Frequente	117
9.3	Procedimento di Analisi	119
9.3.1	Lancio di GSP	120
9.3.2	Filtraggio delle Sequenze Inusuali	121
9.3.3	Sommario degli Esami Non Congruenti	121
9.4	Risultati Finali dell'Analisi	122
10	CONCLUSIONI E FUTURI SVILUPPI	125
	Appendici	129
A	MAKEFILE E CODICE PYTHON	129
A.1	Pulizia delle Collection	129
A.2	Aggregazione degli Attributi	134
A.3	Discretizzazione	142
A.4	Join	146
A.5	Produzione Data Set	149
A.5.1	Minimizzazione degli Attributi sul Data Set Unito	149
A.5.2	Condensazione delle Valutazioni sui Insegnamenti	150
A.5.3	Condensazione della Produttività degli Studenti	151
A.5.4	Sequenze Ordinate di Esami Superati	153
A.5.5	Pattern Sequenziali Inusuali	157
A.5.6	Esami più Frequentemente Fuori Posto	159
A.6	MakeFile	161

ELENCO DELLE FIGURE

Figura 1	Anni Accademici coperti dai dati a disposizione sugli studenti	14
Figura 2	logo del <i>data base management system</i> MongoDB . . .	22
Figura 3	logo del linguaggio di programmazione Python . .	23
Figura 4	schema riassuntivo dell'interazione fra un databa- se MongoDB e pymongo	24
Figura 5	logo della famiglia di software con licenza GNU . .	25
Figura 6	logo della suite di software per attività di <i>data mi- ning</i> Weka	26
Figura 7	logo del linguaggio di programmazione R	27
Figura 8	grafici di dispersione riguardanti attributi generali del data set degli studenti	34
Figura 9	grafico di dispersione sugli attributi "voto medio" e "test di ingresso"	35
Figura 10	grafico di dispersione sugli attributi "voto medio" e "CFU ottenuti"	36
Figura 11	diagramma a scatole e baffi relativo all'attributo "numero totale di crediti"	37
Figura 12	diagramma a scatole e baffi relativo all'attributo "voto medio"	38
Figura 13	grafici di dispersione riguardanti gli attributi rela- tivi agli esami del primo anno	40
Figura 14	matrice di correlazione fra le istanze degli studen- ti, le cui righe e colonne rappresentano i vettori composti dai voti sugli esami del primo anno	41
Figura 15	matrice dello scarto quadratico medio dei risultati degli esami da sostenere il primo anno	42
Figura 16	sommario dell'attributo "data" relativo agli esami del primo anno	43
Figura 17	grafico di dispersione fra gli attributi "voto ottenu- to a Matematica Discreta e Logica" e "CFU ottenuti nel periodo in esame"	44

Figura 18	grafico di dispersione fra gli attributi "voto" e "data di superamento" relativi all'esame di Matematica Discreta e Logica	45
Figura 19	grafico di dispersione fra gli attributi "voto ottenuto a Architetture degli Elaboratori" e "CFU ottenuti nel periodo in esame"	46
Figura 20	grafici di dispersione relativi agli esami a contenuto prevalentemente informatico	47
Figura 21	matrice di correlazione fra gli studenti, le cui righe e colonne rappresentano i vettori composti dai voti sugli esami a contenuto informatico	48
Figura 22	matrice dello scarto quadratico medio dei risultati degli esami ad argomento informatico	49
Figura 23	grafici di dispersione relativi agli esami ad argomento prevalentemente matematico	50
Figura 24	matrice di correlazione fra gli studenti, le cui righe e colonne rappresentano i vettori composti dai voti sugli esami a contenuto matematico	51
Figura 25	matrice dello scarto quadratico medio dei risultati degli esami a contenuto matematico	52
Figura 26	porzione del makefile che specifica le ricette per eseguire il <i>join</i> fra le valutazioni dei corsi e i dati degli studenti	65
Figura 27	porzione del makefile che specifica le ricette per eseguire il <i>join</i> fra le valutazioni dei corsi e i dati degli studenti	70
Figura 28	istogramma mostrante una panoramica generale su tutti gli attributi del dataset descritto in Sezione 5.2.2	73
Figura 29	serie storica dell'attributo "percentuale di valutazioni sufficienti" attraverso gli Anni Accademici coperti dal data set delle valutazioni dei corsi	73
Figura 30	istogramma mostrante una panoramica generale su tutti gli attributi generali del dataset descritto in Sezione 5.2.1	75
Figura 31	serie temporale mostrante i valori degli attributi normalizzati "voto medio" e "valutazione media al test di ingresso"	76

Figura 32	serie temporale mo­strante l'andamento dell'at­tri­buto "ritardo medio" relativo al superamento degli esami di profitto	76
Figura 33	serie temporale mo­strante l'andamento dell'at­tri­buto "percentuale di studenti laureati entro tre An­ni Accademici dall'immatricolazione"	77
Figura 34	finestra che mostra i parametri impostabili dell'al­goritmo K-Means	82
Figura 35	finestra che mostra i parametri impostabili dell'im­plementazione di DBSCAN su Weka	83
Figura 36	sezione dello spazio di esistenza del data set del­le valutazione dei corsi lungo il piano definito da "valutazione complessiva" e "percentuale di valuta­zioni sufficienti", con evidenziato per ogni istanza il cluster di appartenenza	86
Figura 37	matrice di prossimità, matrice di incidenza e cor­relazione fra di esse per misurare la contà del clu­stering effettuato con K-Means	89
Figura 38	sezione dello spazio di esistenza del data set delle valutazione dei corsi lungo il piano definito da "vo­to medio" e "ritardo medio", con evidenziato per ogni istanza il cluster di appartenenza	90
Figura 39	sezione dello spazio di esistenza del data set del­le valutazione dei corsi lungo il piano definito da "ritardo medio" e "valutazione della didattica", con evidenziato per ogni istanza il cluster di apparte­nenza	91
Figura 40	sezione dello spazio di esistenza del data set del­le valutazione dei corsi lungo il piano definito da "voto medio" e "valutazione della didattica", con evidenziato per ogni istanza il cluster di apparte­nenza	92
Figura 41	composizione del Cluster 0 riguardo gli Anni Ac­cademici considerati dal data set	93
Figura 42	composizione del Cluster 1 riguardo gli Anni Ac­cademici considerati dal data set	94
Figura 43	tabella che mostra la composizione dei cluster in riferimento ai corsi di esame	96

Figura 44	istogramma che mostra la somma dell'attributo "valutazione della didattica" nei due cluster ottenuti, divisa per Anno Accademico	98
Figura 45	istogramma che mostra la somma dell'attributo "ritardo medio nel dare gli esami" nei due cluster ottenuti, divisa per Anno Accademico	98
Figura 46	istogramma che mostra la somma dell'attributo "voto medio conseguito negli esami di profitto" nei due cluster ottenuti, divisa per Anno Accademico	99
Figura 47	piano composto dalle dimensioni "Voto Medio negli Esami" e "Valutazione Media dell'Insegnamento", sul quale è stato tentato il clustering con l'algoritmo DBSCAN	100
Figura 48	pannello di scelta delle impostazioni per l'algoritmo Apriori di Weka	103
Figura 49	finestra che mostra i parametri impostabili dell'implementazione di GSP su Weka	116
Figura 50	File contenente le sequenze più interessanti estratte dai risultati dell'algoritmo GSP	121
Figura 51	Grafico a torta che riassume la composizione degli esami più spesso rimandati in favore di altri ad essi successivi	123

*"Di grazie assai sincere
la Famiglia ne ha diritto:
Madre, Padre, Nonno e Nonna
mai mancò l'Alloggio e il Vitto.*

*Sara mia, ce l'ho fatta!
Mai negasti la presenza
né le stampe generose
né la tua Santa Pazienza.*

*Grazie Specca, grazie Sarti
miei due unici compari,
il reciproco Supporto
ci aiutò a finire in pari.*

*Grazie amico ingegnere
il Sostegno non è poco:
mi prestasti un caro libro
a cui io, poi, detti fuoco.*

*Il pensiero ai professori,
chi ostacolo e chi aiuto,
va a coloro che alleviaron
di serenità il Tributo.*

*Grazie soprattutto a me:
con impegno e con fatica
affrontai un Sacrificio
per sperare nella vita.*

*Questo piccolo traguardo
senza guida, senza esempi
sfido molti a replicarlo
nei miei modi e nei miei tempi.*

*La mia breve incursione
in Accademia, è tutta qui.
Tornerò? Lo scopriremo
fra qualche anno, o giù di lì."*

— Simone Cipriani

INTRODUZIONE

In un mondo dove l'informatica si è espansa fino a rasentare l'onnipresenza, nuove informazioni vengono prodotte così velocemente e in quantità tale da sopraffare persino la potenza di calcolo dei computer. Perciò, è fondamentale conoscere e padroneggiare delle tecniche per affrontare efficientemente i *big data*, moli di dati così grandi da essere intrattabili con la forza bruta.

Questo studio è il naturale proseguimento del lavoro svolto come progetto per il corso di *Data Mining and Organization*¹. L'attività svolta in quella sede consisteva in una *cluster analysis* dei dati relativi alla carriera universitaria degli studenti del Corso di Laurea Triennale in informatica. Facendo tesoro di quanto realizzato in quell'occasione, ed ispirandosi in parte a quanto già realizzato in [12], è stata portata avanti un'analisi molto più ampia e approfondita sullo stesso ambito, integrando il materiale iniziale e spostando il focus dagli studenti agli insegnamenti.

Questo documento riporta la descrizione di ogni fase delle attività di *data understanding* e *data mining* svolte durante lo studio dei dati a disposizione:

- nel Capitolo 2, verrà analizzata e descritta la natura dei dati iniziali;
- nel Capitolo 3, si illustreranno le tecnologie che compongono la *technology stack* impiegata, con particolare enfasi sulla scelta di esse in relazione al risultato da raggiungere;

¹ Si tratta di un esame del Corso di Laurea Magistrale in informatica, curriculum *Data Science*, che l'autore di questo documento ha deciso di intraprendere per conseguire i crediti a libera scelta.

- nel Capitolo 4, si mostrerà il risultato di una fase preliminare di *data understanding* effettuata direttamente sui dati grezzi, allo scopo di intuire quali informazioni possono esserne estratte;
- nel Capitolo 5, si descriverà il procedimento di *preprocessing* atto a mutare la forma dei dati iniziali per rendere utilizzabili su di essi alcune tecniche di *data mining*;
- nel Capitolo 6, verranno mostrati e interpretati dei grafici ottenuti mediante alcune tecniche di visualizzazione, atti a evidenziare le tendenze degli attributi generali di ogni classe di record;
- nel Capitolo 7, sarà illustrata la fase di *cluster analysis*, volta a cercare delle zone di agglomerazione nello spazio definito dagli attributi dei dati;
- nel Capitolo 8, si descriverà il procedimento di ricerca di regole associative fra la produttività degli studenti e le valutazioni dei corsi;
- nel Capitolo 9, verrà mostrato l'utilizzo dell'algoritmo *Generalized Sequential Pattern* per la ricerca di pattern sequenziali frequenti nell'ordine di superamento degli esami;
- infine, nel Capitolo 10, si cercherà di riassumere quanto rilevato in tutte le analisi e si discuterà di eventuali nuovi possibili sviluppi di questo lavoro.

Completa l'opera l'Appendice 10, dove sono riportati il codice e i file significativi per qualche aspetto, ma che non hanno trovato spazio nei relativi capitoli.

DATI INIZIALI

In questo capitolo ci si soffermerà in quella che è una preliminare analisi dei dati iniziali a disposizione. Questi dati rappresentano il materiale grezzo dal quale *estrarre* — fare del *mining*, come appunto il nome dell'attività suggerisce — delle informazioni.

Usando come metafora una lavorazione meccanica, avere ben chiara la natura del materiale grezzo a disposizione consente di scegliere opportunamente gli utensili adatti per il lavoro da fare. Nel nostro caso, poter vantare di una comprensione generale di ciò che si ha a disposizione, potrà consentirci di scegliere le tecniche migliori per trarre il meglio dai dati iniziali.

2.1 CARRIERA DEGLI STUDENTI

Una parte fondamentale dell'analisi descritta in questo lavoro è basata sul seguente data set, che contiene i dati riguardanti la produttività di tre coorti d'immatricolazione di studenti in un periodo di quattro anni. Più nel dettaglio, il dataset si compone di:

- **coorte 2010**: studenti immatricolati nel 2010, carriera registrata fino agli appelli di **febbraio 2014**;
- **coorte 2011**: studenti immatricolati nel 2011, carriera registrata fino agli appelli di **febbraio 2015**;
- **coorte 2012**: studenti immatricolati nel 2012, carriera registrata fino agli appelli di **febbraio 2016**;

Figura 1: Anni Accademici coperti dai dati a disposizione sugli studenti

A.A.	2010-2011	2011-2012	2012-2013	2013-2014	2014-2015	2015-2016	2016-2017
	Coorte 2010						
		Coorte 2011					
			Coorte 2012				
				Coorte 2013			

- **coorte 2013**: studenti immatricolati nel 2013, carriera registrata fino agli appelli di **febbraio 2017**.

Quindi, si ha a disposizione una finestra temporale di risultati ottenuti negli esami composta approssimativamente come indicato in Figura 1. Questo fatto porta ovviamente ad avere una mole di informazioni più addensata nella parte centrale della nostra finestra temporale, avendo idealmente dati riguardanti gli esiti del maggior numero di corsi possibili solo per l'Anno Accademico 2013-2014. Tale aspetto sarà rilevante nell'interpretare i risultati di alcune analisi effettuate in seguito.

2.1.1 *Formato e Rappresentazione*

Il dataset è stato fornito in un unico file CSV, un formato *plain text* facilmente manipolabile e interpretabile da un'ampia gamma di software. Esso si compone di una unica tabella, nella quale ogni tupla identifica uno studente, per il quale sono presenti attributi che descrivono la sua carriera universitaria nel periodo preso in esame. Oltre alle informazioni generali, quali ad esempio i risultati conseguiti nel test d'ingresso e il numero di crediti totali ottenuti nella finestra temporale esaminata, sono presenti attributi relativi alla data e al voto ottenuto in ciascun esame sostenuto.

Nel dettaglio, per ogni tupla rappresentante uno studente sono presenti le seguenti informazioni:

- **Coorte** di immatricolazione:

{2010, 2011, 2012, 2013}

- Voto conseguito nel **test di ingresso**:

$$\{x \in \mathbb{N} \text{ tale che } 0 \leq x \leq 25\}$$

- Voto ottenuto all'**esame di maturità**:

$$\{x \in \mathbb{N} \text{ tale che } 60 \leq x \leq 100\}$$

- Tipo di **scuola superiore** frequentata:

$$\{LS, LC, IT, TC, IP, AL\}$$

che rappresentano rispettivamente le seguenti categorie di scuola superiore: Liceo Scientifico, Liceo Classico, Istituto Tecnico, Istituto Commerciale, Istituto Professionale, Altro

- **Crediti totali** ottenuti:

$$\{x \in \mathbb{N} \text{ tale che } 0 \leq x \leq 180\}$$

- **Crediti** ottenuti da esami **con voto**:

$$\{x \in \mathbb{N} \text{ tale che } 0 \leq x \leq 159\}$$

- **Voto medio** ottenuto negli esami:

$$\{x \in \mathbb{N} \text{ tale che } 18 \leq x \leq 31 \text{ con } 31 \text{ indicante il } 30 \text{ con lode}\}$$

- **Voto** ottenuto in un **certo esame**:

$$\{x \in \mathbb{N} \text{ tale che } 18 \leq x \leq 31 \text{ con } 31 \text{ indicante il } 30 \text{ con lode}\}$$

- **Data** in cui è stato sostenuto quell'esame:

$$\text{data in formato MM/GG/YYYY}$$

Di tutte queste informazioni, solo alcune sono state utilizzate in qualche analisi di *data mining*, come si vedrà meglio nelle sezioni successive.

2.1.2 Mole di dati

Il dataset si compone di 208 record, ognuno dei quali ha 47 attributi. Il file che lo memorizza pesa circa 45 kb. Non si può quindi parlare propriamente di *big data* in questo caso.

2.2 VALUTAZIONE DEGLI INSEGNAMENTI

Al fine d'integrare i dati precedenti ponendo l'attenzione sui vari corsi che compongono il Corso di Laurea, sono stati forniti i dati relativi alla valutazione dei corsi di studi da parte degli studenti. Questi dati sono ottenuti da questionari anonimi, che devono essere obbligatoriamente compilati prima di potersi prenotare per un esame. I risultati sono poi divulgati in forma aggregata, garantendo così l'anonimato dello studente. A ogni domanda lo studente ha potuto rispondere indicando un valore compreso fra zero e dieci, con zero a indicare una risposta totalmente negativa e dieci a indicare invece una risposta totalmente positiva.

Volendo scendere in un maggior dettaglio, sono stati forniti i dati riguardanti i seguenti anni accademici:

- 2010-2011
- 2011-2012
- 2012-2013
- 2013-2014
- 2014-2015
- 2015-2016
- 2016-2017

Come si può facilmente notare, la finestra temporale coperta da questi dati va a combaciare con quella trattata dal dataset relativo alla carriera degli studenti. Questo aspetto fondamentale ha permesso di effettuare una operazione di *join* fra i due dataset a disposizione, che verrà in seguito descritta nella sezione dedicata al *preprocessing*.

2.2.1 Formato e Rappresentazione

Il dataset è stato fornito in sette diversi file CSV, uno per ogni anno accademico per il quale sono state espresse valutazioni dei relativi corsi. In ogni file si rappresenta una tabella le cui tuple identificano una valutazione relativa a un particolare aspetto di un corso, riportata ovviamente

in forma aggregata.

Nel particolare, ogni record di questo tipo di tabelle è identificato dai seguenti campi, che agiscono come chiave:

- **Codice identificativo:** stringa alfanumerica che identifica univocamente l'esame oggetto di valutazione all'interno del Corso di Laurea in esame.
- **Nome del corso:** stringa descrittiva che identifica il Corso di Laurea (*in questo caso, "INFORMATICA"*).
- **Tipo di corso:** stringa descrittiva che identifica il tipo di Corso di Laurea (*in questo caso, "Triennale"*).
- **Insegnamento:** stringa descrittiva che identifica l'esame oggetto di valutazione.
- **Docente/i:** stringa descrittiva che identifica il docente che ha tenuto il corso e svolto l'esame.
- **Paragrafo:** stringa descrittiva che identifica il paragrafo del questionario di valutazione.
- **Q:** stringa alfanumerica che identifica univocamente la domanda posta.
- **Quesito:** stringa descrittiva contenente il testo della domanda posta allo studente.

Si noti che il campo riguardante il docente ne riporta originariamente nome e cognome per esteso. Per salvaguardarne la privacy, i valori di questo campo sono stati sostituiti con le loro immagini tramite una funzione di *hash*, preservando l'unicità del valore ma nascondendo l'effettiva identità del docente.

A ogni tupla, identificata dai valori dei campi precedentemente descritti, corrispondono queste informazioni:

- **P1, P2:** $\{x \in \mathbb{R} \text{ tale che } 0 \leq x \leq 100\}$ percentuali rispettivamente di risposte sufficienti (≥ 6) e insufficienti (< 6).

- **Media:** $\{x \in \mathbb{R} \text{ tale che } 0 \leq x \leq 10\}$ media aritmetica delle valutazioni ottenute.
- **Deviazione Standard:** $\{x \in \mathbb{R} \text{ tale che } x \geq 0\}$ scarto quadratico medio delle singole valutazioni.
- **N:** $\{x \in \mathbb{N} \text{ tale che } x \geq 6\}$ quantità di valutazioni utilizzate per calcolarle e precedenti valori.

Come è possibile intuire, molti di questi attributi sono inutili o ridondanti. Il compito di valutarne l'utilità ed eventualmente di sfoltirli sarà svolto nella fase di *preprocessing*.

2.2.2 Mole di dati

I sette file forniti contengono complessivamente 2594 record, ognuno dei quali ha 13 attributi. Anche riguardo a questo dataset, non si può usare propriamente la denominazione *big data*.

In ogni caso, visto che le quantità in gioco non sono comunque piccole, in una eventuale *join* con il dataset precedente occorrerà fare particolare attenzione a non moltiplicare la quantità di record generando ridondanze, in quanto un simile errore potrebbe facilmente rendere l'insieme di dati risultante intrattabile.

2.3 CONCLUSIONI DELL'ANALISI DEI DATI INIZIALI

Dopo aver esaminato attentamente i due dataset a disposizione, si può immediatamente affermare che il focus principale dell'analisi dovrà essere posto sui singoli corsi, per i quali si hanno molte informazioni di vario genere.

Volendo quindi sintetizzare quanto è stato possibile capire dall'analisi presentata in questa sezione, elaborandolo nell'ottica appena acquisita, si può riassumere la descrizione del materiale a nostra disposizione in due semplici punti:

- risultati dei singoli studenti
- aggregazioni delle risposte ai questionari di valutazione dei corsi

Oltre a effettuare analisi sui singoli insiemi di dati, si potrà immaginare di doverli in qualche modo unire per incrociarne le informazioni e trovare, possibilmente, correlazioni interessanti. Si può quindi dire che la sfida più impegnativa della prossima fase, il *preprocessing*, riguardi la messa in relazione di dati aventi natura diversa.

Di pari passo a essa, sarà portata avanti una fase di *data understanding*. Sarà utile per affinare la comprensione di quanto si è appena mostrato e per decidere il tipo di tecniche di *data mining* da utilizzare sulla mole di dati a disposizione.

TECHNOLOGY STACK

La scelta della tecnologia da impiegare è un aspetto fondamentale di ogni progetto, e questo chiaramente non fa eccezione.

Molti testi — fra cui uno dei preferiti dall'autore¹ — hanno proposto analogie fra gli utensili di un artigiano e gli strumenti software di un qualunque utilizzatore informatico. Sull'onda di questa metafora, una lima non adatta al tipo di materiale che si intende lavorare non potrà mai garantire risultati eccellenti, indipendentemente dalla bravura dell'artigiano stesso.

Appare chiaro che una scelta sbagliata di tecnologie da utilizzare può condizionare negativamente l'esito di un lavoro, pertanto occorre prestare particolare attenzione nel decidere quale tecnologia impiegare per raggiungere lo scopo.

3.1 STRUMENTI NECESSARI

Messe da parte le metafore, all'atto pratico è evidente che gli strumenti di cui l'analisi ha bisogno sono in realtà pochi e piuttosto comuni:

- abbiamo a che fare con una certa quantità di dati → servirà un *Data Base Management System*.
- occorre avere una comprensione generale dei dati → occorrerà un software che permetta di usare *tecniche di visualizzazione*.

¹ Si tratta di [16], un libro estremamente interessante, oltre che essenziale come spunto di riflessione per lo sviluppo personale e professionale di ogni informatico che osi definirsi programmatore.

Figura 2: logo del *data base management system* MongoDB

- dobbiamo lanciare degli algoritmi di *data mining* → abbiamo bisogno di un software che li implementi.

Tutto questo è facilmente ottenibile impiegando gli strumenti che saranno descritti nelle prossime sezioni.

3.2 DATA BASE MANAGEMENT SYSTEM: MONGODB

La scelta del Data Base Management System è stata quella che ha avuto un impatto maggiore sul tipo di lavoro che è stato necessario fare.

3.2.1 *La scelta di MongoDB*

Sono state valutati principalmente due *DBMS* candidati, entrambi *open source*:

- **MySQL**, un *DBMS* relazionale, rodato e ormai *pietra miliare* nella manipolazione dati
- **MongoDB**, un *DBMS* di nuova generazione che adotta invece il paradigma *NoSQL*

I dati che si hanno a disposizione sono in forma ovviamente relazionale — cioè, sono tabelle. Utilizzare *MySQL* potrebbe sembrare quindi una scelta tanto solida quanto ovvia, ma sul piano prestazionale il *modello a documenti* di *MongoDB* risulta sensibilmente più efficiente nel manipolare grandi quantità di dati molto diversi fra loro.

La decisione che è stata infine presa — *utilizzare MongoDB* — ha tenuto conto anche del valore didattico che ha l'imparare un nuovo paradigma di memorizzazione dati che trascende il classico modello relazionale.

Figura 3: logo del linguaggio di programmazione Python



3.2.2 Caratteristiche di MongoDB

Come si può banalmente leggere in [2]:

"**MongoDB** (da "*humongous*", *enorme*) è un DBMS non relazionale, orientato ai documenti. Classificato come un database di tipo **NoSQL**, MongoDB si allontana dalla struttura tradizionale basata su tabelle dei database relazionali in favore di **documenti in stile JSON** con schema dinamico (MongoDB chiama il formato BSON), rendendo l'integrazione di dati di alcuni tipi di applicazioni più facile e veloce."

La società MongoDB Inc mette a disposizione liberamente e gratuitamente una buona documentazione in [1], la quale contiene tutte le informazioni necessarie per utilizzare al meglio il DBMS da loro sviluppato. Oltre alla sua *shell*, MongoDB mette a disposizione delle *API* per i principali linguaggi di programmazione. Questo aspetto è tornato molto utile per specificare le operazioni di *preprocessing* in un linguaggio familiare e più facilmente gestibile del dialetto di Javascript nativo di MongoDB.

3.3 INTERAZIONE COL DBMS: PYTHON

A proposito di quando detto alla fine della sezione precedente, il linguaggio scelto è stato **Python**. Come si può leggere direttamente da [5]:

"**Python** è un linguaggio multi-paradigma, che ha tra i principali obiettivi *dinamicità, semplicità e flessibilità*. Supporta il paradigma object oriented, la programmazione strutturata e molte caratteristiche di programma-

Figura 4: schema riassuntivo dell'interazione fra un database MongoDB e pymongo



zione funzionale e *Reflection*."

Il linguaggio Python ha una estesa e approfondita documentazione, messa a disposizione direttamente dalla Python Software Foundation in [4].

3.3.1 API per MongoDB: pymongo

Una sintetica ma tremendamente efficace descrizione di che cosa è pymongo si può trovare direttamente nella sua documentazione, in [3]:

"**PyMongo** is a Python distribution containing tools for working with MongoDB, and is the recommended way to work with MongoDB from Python."

Che tradotto in italiano significa:

"**PyMongo** è una distribuzione di Python che contiene degli strumenti per lavorare con MongoDB, ed è la maniera raccomandata per avere a che fare con MongoDB da Python."

Figura 5: logo della famiglia di software con licenza GNU



In altre parole, *pymongo* è una libreria per Python che fornisce oggetti e metodi per lanciare istruzioni di *Data Definition*² e *Data Manipulation*³.

3.4 ORGANIZZAZIONE: GNU MAKE

Data la complessità delle operazioni da eseguire, occorre un modo per dare e mantenere una struttura chiara nel lancio degli script necessari a manipolare opportunamente i dati. A questo proposito, è stato scelto il software Make, un'antica *utility* GNU eccezionalmente solida.

Citando direttamente da [6], "**GNU Make** is a tool which controls the generation of executables and other non-source files of a program from the program's source files.", che in italiano significa: "**GNU Make** è uno strumento che controlla la generazione di eseguibili ed altri file non sorgenti a partire dal codice sorgente."

Sebbene lo scopo più comune di Make sia la compilazione di programmi, è perfettamente malleabile per adattarsi a qualunque tipo di applicazione. Sempre da [6] si può leggere qualcosa proprio a questo riguardo:

"Make is not limited to building a package. You can also use Make to control installing or deinstalling a package, generate tags tables for it, or

² Data Definition Language: sottoinsieme di un linguaggio comprendente le istruzioni per definire e manipolare le strutture (in un DBMS relazionale, le *tabelle*, in MongoDB le *collezioni* di documenti). Ad esempio, in **SQL** comprende le istruzioni **create table**, **alter table**, ...

³ Data Manipulation Language: sottoinsieme di un linguaggio comprendente le istruzioni per manipolare i dati. Ad esempio, in **SQL** sono ad esempio **insert**, **update**, ...

Figura 6: logo della suite di software per attività di *data mining* Weka

anything else you want to do often enough to make it worth while writing down how to do it."

Ovvero, traducendo e parafrasando:

"Make non è limitato alla compilazione di un programma. Si può usare Make per controllare l'installazione o la disinstallazione di un pacchetto software, generare tabelle di tag, o qualunque altra cosa ci sia bisogno di fare spesso abbastanza da far valere la pena di automatizzarla."

Pertanto, è stato scritto un makefile — mostrato per intero nell'Appendice A.6 — per automatizzare il lancio dei vari script necessari a realizzare le operazioni di *preprocessing*.

3.5 ALGORITMI DI DATA MINING: WEKA

Come si può leggere direttamente da [8], "**Weka**, acronimo di **Waikato Environment for Knowledge Analysis**, è un software per l'apprendimento automatico sviluppato nell'Università di Waikato in Nuova Zelanda. È open source e viene distribuito con licenza *GNU General Public License*. Curiosamente la sigla corrisponde al nome di un simpatico animale simile al Kiwi, presente solo nelle isole della Nuova Zelanda."

Figura 7: logo del linguaggio di programmazione R



Sempre secondo [8], "Weka è un ambiente software interamente scritto in Java. Un semplice metodo per utilizzare questo software consiste nell'applicare dei metodi di apprendimento automatici (*learning methods*) ad un set di dati (*dataset*), e analizzarne il risultato".

Data la natura *I/O intensive* dell'attività di data mining, è da notare il fatto che l'implementazione di Weka in un linguaggio con un discreto *overhead* come Java non è da considerarsi rilevante nell'impatto sulla effettiva velocità degli algoritmi lanciabili su questa suite.

Nel merito del nostro lavoro, Weka è una *suite* di sottoprogrammi molto potente, che permetterebbe di effettuare ogni aspetto del processo di *data mining*, dal *preprocessing* al *postprocessing*. Ai fini del nostro scopo, questo software verrà utilizzato principalmente come ambiente per lanciare gli algoritmi implementati dalle sue librerie. Come si può vedere consultando la sua estesa documentazione in [7], Weka può essere integrato in un gran numero di ambienti; tuttavia, per questa applicazione è stato scelto di utilizzarlo in forma *stand alone*, preferendo concentrarsi sul merito delle analisi da fare e lasciando perdere il lavoro di programmazione che sarebbe seguito da scelte diverse.

3.6 TECNICHE DI VISUALIZZAZIONE

3.6.1 Linguaggio R

Direttamente dalla pagina Wikipedia relativa al software R (in [10]), "R è un linguaggio di programmazione e un ambiente di sviluppo specifico

per l'analisi statistica dei dati. [...] È un software libero in quanto viene distribuito con la licenza GNU GPL, ed è disponibile per diversi sistemi operativi (ad esempio Unix, GNU/Linux, macOS, Microsoft Windows). Il suo linguaggio orientato agli oggetti deriva direttamente dal pacchetto S [...]."

Un'estesa ed approfondita documentazione può essere trovata in [9]. Come si può notare consultandola, le potenzialità di questo software sono molto vaste; nel merito di questo studio è stato usato solamente una minuscola frazione delle funzioni offerte — ovvero la realizzazione di grafici a partire da un set di dati — perciò, analogamente a quanto osservato riguardo a Weka, è stato scelto di non spendere troppo tempo nell'integrazione di questo strumento in un flusso di automazione ma di usarlo soltanto in forma a sé stante.

3.6.2 Fogli di Calcolo

Per quanto banale possa sembrare, per alcuni dati molto semplici può essere utile non andare a scomodare strumenti troppo complessi, impiegando invece attrezzi dall'utilizzo estremamente semplice, come un foglio di calcolo. A questo proposito, è stato scelto di utilizzare Calc ([11]), facente parte della *suite da ufficio* LibreOffice, un progetto open source che si pone come alternativa a Microsoft Office.

Con esso è stato possibile realizzare semplici grafici di data set dalla mole ridotta, senza incorrere nell'*overhead* — inteso come carico di lavoro necessario a raggiungere un risultato simile — che avrebbe comportato l'utilizzo di R.

3.7 REDAZIONE DI QUESTO DOCUMENTO

Per quanto scontato e irrilevante possa sembrare, una parte integrante del lavoro svolto è stato quello di redarre questo documento. Vista in quest'ottica, appare chiaro che la scelta di uno strumento adatto possa condizionare in positivo la chiarezza e l'efficacia di questa produzione scritta.

Dato che lo standard accademico impone l'utilizzo del $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ per questo genere di lavori, è stata scelta una distribuzione aggiornata e supportata di esso, **TeX Live**, ottenuta direttamente dalle repository di Arch Linux.

3.8 CONCLUSIONI

Volgendo uno sguardo generale a quanto scritto fino a ora, possiamo dire che la nostra technology stack si compone di un mix di strumenti vario e tutto sommato moderno. Alcuni di essi sono di stampo generalista, non specifici per il nostro campo di applicazione, ma comunque sufficientemente malleabili per poter essere impiegati adeguatamente.

Riassumendo il tutto in pochi, semplici punti, la technology stack è la seguente:

- MongoDB come DBMS;
- Python (pymongo) per manipolare i dati su MongoDB;
- R e LibreOffice Calc per le tecniche di visualizzazione;
- Weka per gli algoritmi di data mining;
- TeX Live da Arch Linux come distribuzione $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$.

Ora che è ben chiaro cosa verrà utilizzato e a che scopo, si è in grado di cominciare finalmente il lavoro sui dati.

DATA UNDERSTANDING PRELIMINARE

La fase che verrà descritta in seguito è fra le più delicate di tutto il processo di *data mining*. Da essa dipende la buona riuscita o meno dell'intera attività di estrazione di informazioni. Come appare quindi naturale, è stata dedicata ad essa una particolare attenzione.

4.1 INTRODUZIONE AL DATA UNDERSTANDING

Perché occorre spendere del tempo per perpetrare un'estesa fase di *data understanding*?

Banalmente, l'obiettivo principale che ci si prefigge è quello di ottenere una chiara visione d'insieme sui dati che si hanno a disposizione. In questo modo, si sarà in grado di prendere decisioni oculate per quanto riguarda il da farsi nei successivi passi di questa analisi. Si prenda ad esempio l'attività di *preprocessing*: sintetizzando al massimo, possiamo dire che si tratta di lavorare dei dati grezzi al fine di renderli adatti a essere processati da certi algoritmi. L'ovvia implicazione è che occorre avere ben chiaro in anticipo quali algoritmi sarà opportuno lanciare, fatto che a sua volta implica la necessità di conoscere quali informazioni si intende estrarre dai dati grezzi a disposizione.

Questo aspetto rende necessario far precedere a quella che è, usando una metafora nell'ambito meccanico, l'attività di sgrossatura dei dati grezzi una fase di *data understanding*.

In questa sezione, quindi, si punterà a capire a fondo la natura dei dati al fine d'intuire che cosa sia possibile trarne.

4.1.1 Metodi e Strumenti

L'attività di *data understanding* può essere svolta in molti modi, non essendoci delle regole fisse che la disciplinano rigidamente. Si può azzardare che le risorse più importanti alle quali si può attingere in questo ambito sono, piuttosto romanticamente, l'intuito e la fantasia dei *data scientist* coinvolti nel progetto. Fuori di metafora, è chiaro che non esistono procedure che indichino univocamente le informazioni che è possibile ricavare su un certo insieme di dati. Perciò, quello che occorre fare è sfruttare dei metodi che riassumano i dati per poterli apprezzare nel loro insieme.

Gli strumenti più adatti per poter ottenere una visione d'insieme sull'intera mole di dati sono, senza ombra di dubbio, le varie *tecniche di visualizzazione* esistenti. Il loro utilizzo è stato permesso da software quali **R** e **Weka**, oltre che da più comuni *spreadsheet* forniti da suite di programmi da ufficio.¹

4.2 DATA SET: PRODUTTIVITÀ DEGLI STUDENTI

Senza ombra di dubbio questa porzione di dati rappresenta la più significativa dell'intero lotto a disposizione per questa analisi.

Si è quindi optato per utilizzare il software **R**, in quanto consente d'impiegare efficientemente un gran numero di approfondite tecniche di visualizzazione. Tuttavia, la potenza e la flessibilità offerte da questo tipo di strumento si pagano con la necessità di dedicare un minimo di attenzione all'importazione del data set in quell'ambiente di lavoro.

In senso pratico, questo si traduce nel lanciare del codice analogo — non identico: sono stati accorciati i percorsi — a quanto mostrato di seguito prima di poter effettuare qualunque altra operazione.

```
1 # set the repo root path before importing data!!!
2 setwd("C:/a/path/to/somewhere")
3 students <- read.csv("raw/prod_stud.csv")
4
```

¹ Per un maggior dettaglio, si consulti il capitolo riguardo alla *technology stack* impiegata per la realizzazione di questo progetto.

```

5 # gather some info about the imported data set
6 str(students)
7
8 # ADJUST DATA ATTRIBUTES
9 # convert "coorte" to nominal by making it a factor
10 students[, c(1)] <- sapply(students[, c(1)], factor)
11
12 # ok, let's take a look at it now
13 str(students)
14 summary(students)

```

In ogni caso, dato l'elevato numero degli attributi del data set, è stato necessario prendere delle decisioni relative a cosa visualizzare – e con quale tecnica. Le scelte sono state ovviamente compiute su base intuitiva, a seguito di un'analisi sommaria delle caratteristiche del data set.

È stato quindi deciso di effettuare una ricerca visiva di correlazioni fra valori di attributi relativi a:

- prestazioni generali durante tutto il periodo esaminato
- prestazioni nei singoli esami
- risultati di gruppi di esami

4.2.1 Prestazioni Generali degli Studenti

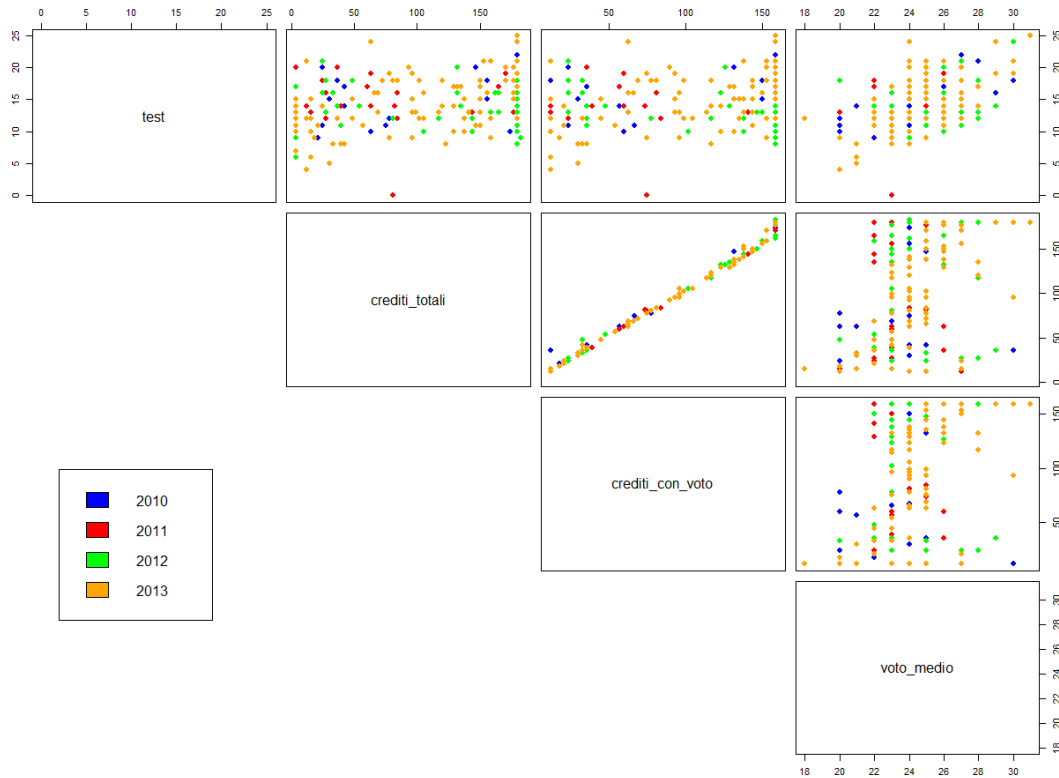
Lo script **R** che segue disegna dei grafici di dispersione su coppie di attributi relativi alle prestazioni generali degli studenti. Lo scopo è quello di individuare visivamente delle correlazioni significative fra questi attributi.

```

1 colors <- c("blue", "red", "green", "orange")
2 coorte_labels <- students[,1]
3 coorte_colors <- colors[as.numeric(coorte_labels)]
4 library(seriation)
5
6 # general attributes
7 students_subset1 <- students[, -c(1, 6 : 45)]
8 pairs(students_subset1, col = coorte_colors, lower.panel =
  NULL, cex.labelsiris=2, pch=19, cex = 1.2)

```

Figura 8: grafici di dispersione riguardanti attributi generali del data set degli studenti



```

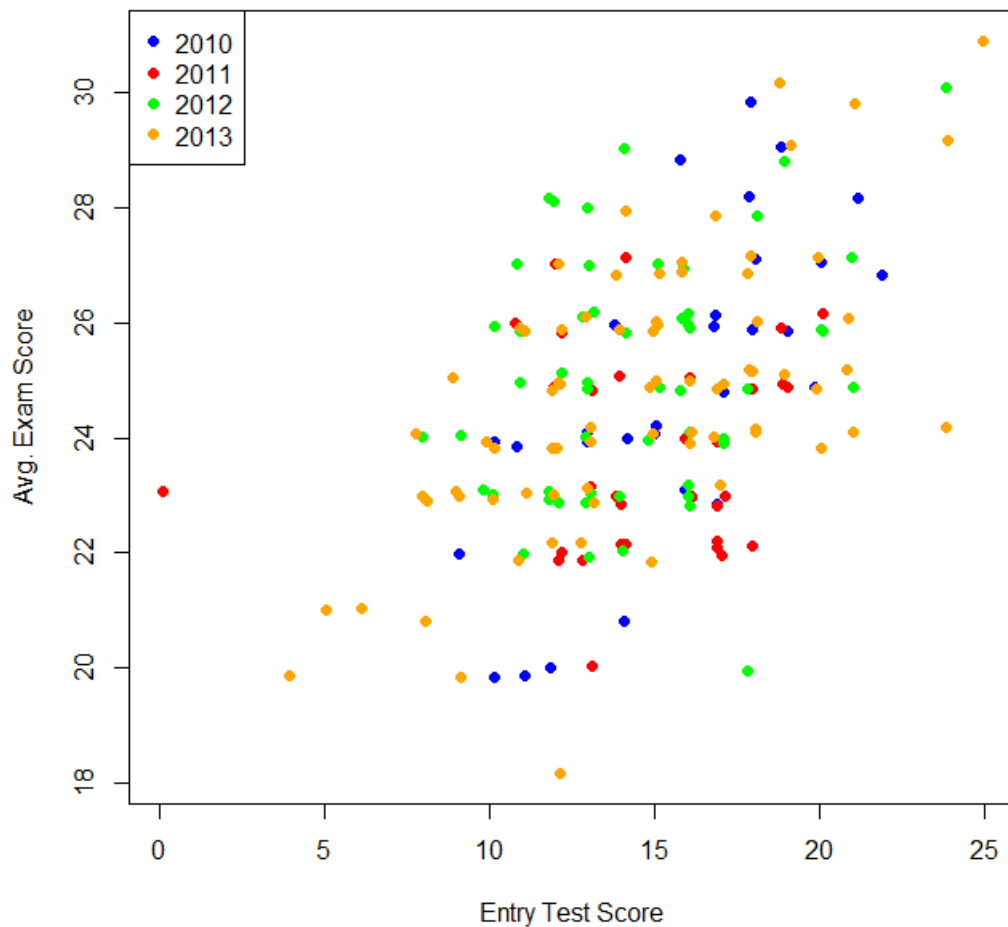
9 par(xpd = TRUE)
10 legend(x = 0.05, y = 0.4, cex = 1, legend =
    as.character(levels(coorte_labels)), fill = unique(coorte_colors))
11 par(xpd = NA)

```

Come si può vedere sul grafico generato dallo script — mostrato in Figura 8 — esistono varie correlazioni fra gli attributi considerati. Nel merito, la relazione fra le quantità *crediti totali* e *crediti con voto* è tanto palese quanto banale: il primo ammontare contiene il secondo, e la loro differenza è minima. Appare invece più interessante quello che accade fra altre coppie di attributi:

- punteggio del test di ingresso e valore atteso del voto
- quantità di crediti ottenuti e valore atteso del voto

Figura 9: grafico di dispersione sugli attributi "voto medio" e "test di ingresso"



Si ritiene quindi opportuno realizzare dei grafici che abbiano un maggior livello di dettaglio su questi aspetti.

Come risulta dalla Figura 9, si potrebbe speculare che esista una correlazione lineare fra i due attributi: gli studenti che conseguono un punteggio alto nel test di ingresso sono più propensi ad ottenere voti alti negli esami.

Per quanto riguarda invece la Figura 10, fra gli studenti che hanno conseguito tutti i crediti si può notare una consistente densità nella fascia che va circa dal ventidue al ventinove. Si può anche notare un'assenza di voti medi inferiori al 22 fra coloro che hanno conseguito più di 100 CFU.

Figura 10: grafico di dispersione sugli attributi "voto medio" e "CFU ottenuti"

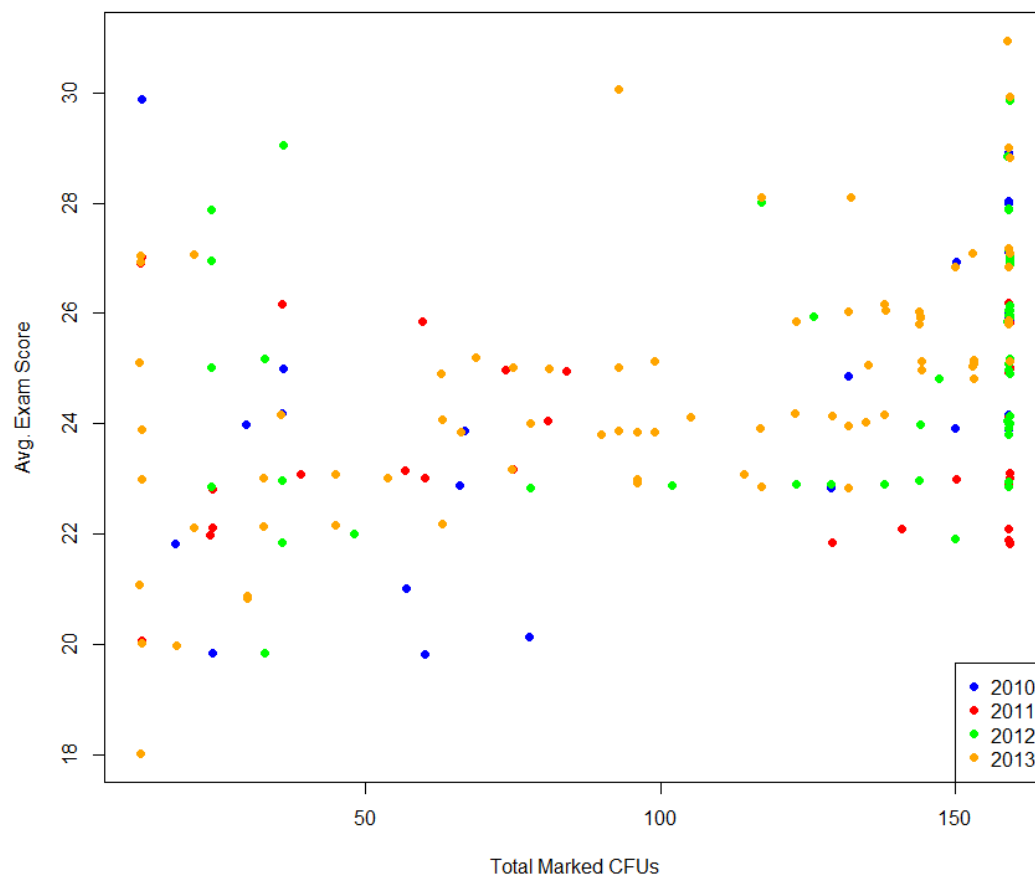
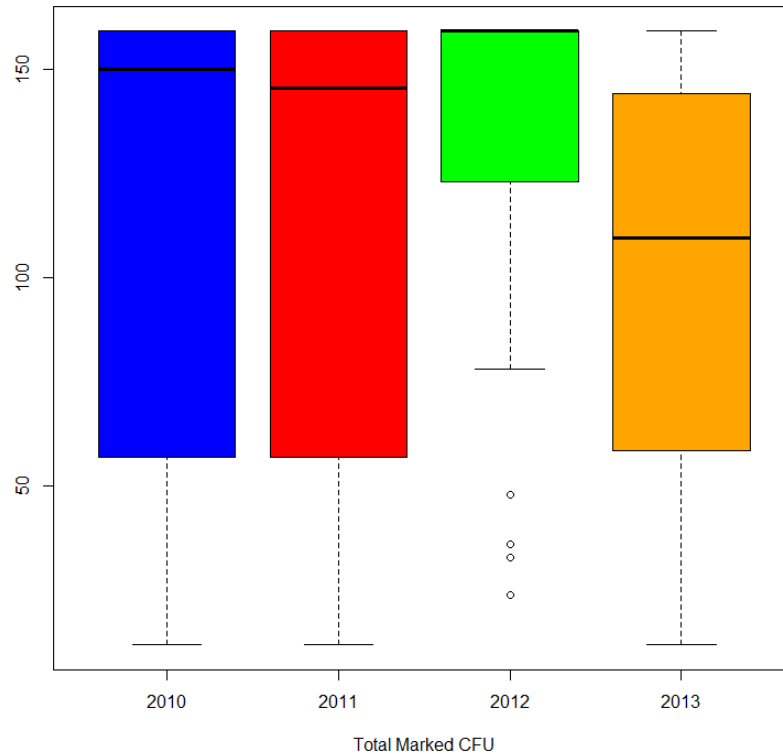


Figura 11: diagramma a scatole e baffi relativo all'attributo "numero totale di crediti"



Un altro aspetto che potrebbe rivelarsi interessante è che, fra la coorte di studenti del 2013, vari studenti hanno conseguito un solo esame con un voto superiore al 20.

Viste le possibili informazioni che sembra possibile estrarre, si è deciso d'insistere sull'analisi visiva degli attributi *voto medio* e *numero totale di crediti*. Sono stati quindi realizzati dei diagrammi a scatola e baffi con il seguente script **R**:

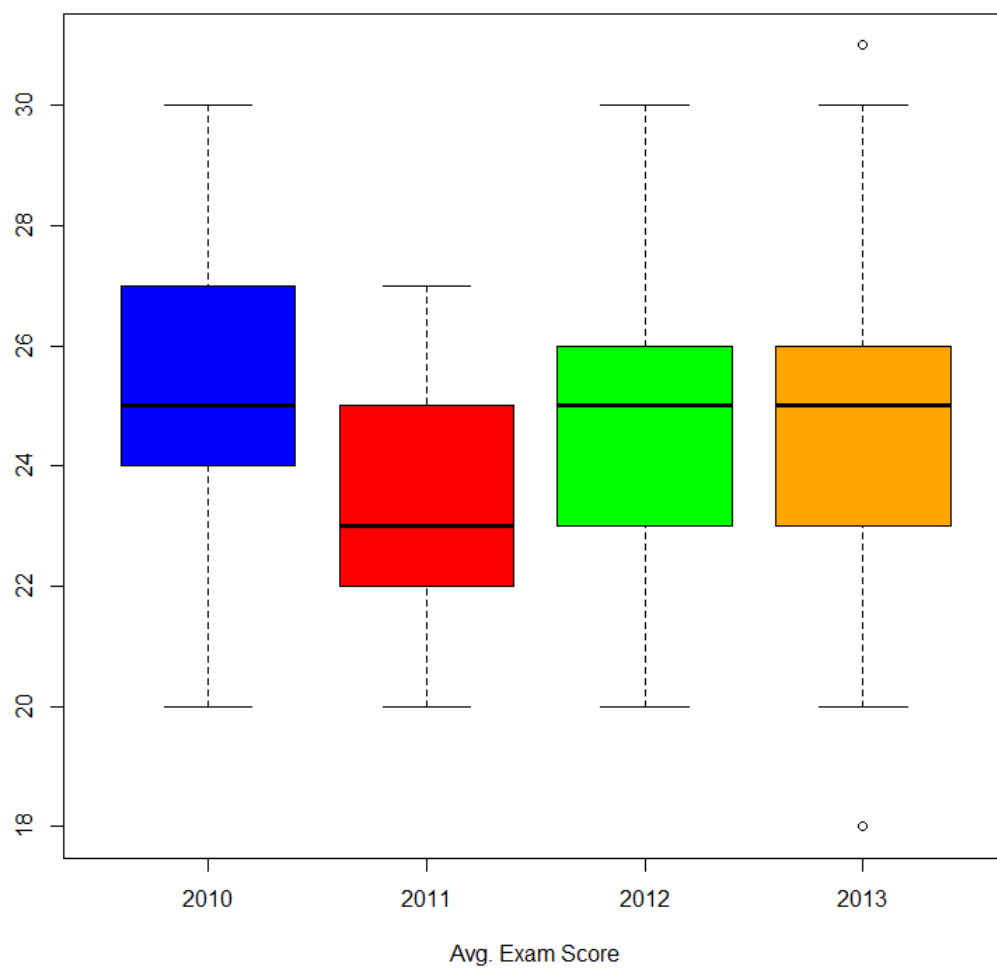
```

1 boxplot(students[,4]~students[,1],data = students,xlab="Total Marked
   CFU",col=colors)
2 boxplot(students[,5]~students[,1],data = students,xlab="Avg. Exam
   Score",col=colors)

```

Riguardo al diagramma in Figura 11, si noti come gli immatricolati nelle annate 2010 e 2011 hanno prestazioni molto simili sul fronte dei cre-

Figura 12: diagramma a scatole e baffi relativo all'attributo "voto medio"



diti conseguiti. La coorte 2012 risulta la migliore, avendo addirittura per mediana il massimo ammontare di crediti ottenibili², mentre gli studenti immatricolati nel 2013 sono stati i meno performanti. Nel caso invece del voto medio, rappresentato in Figura 12, si evidenzia solo un leggero peggioramento negli immatricolati nel 2011 rispetto alle altre coorti di studenti. Sono degni di attenzione anche i due *outliers* nella coorte 2013.

4.2.2 Prestazioni degli Studenti su Gruppi di Esami

Una scelta significativa nell'ambito di questa fase è stata la suddivisione degli esami in vari gruppi. Si è cercato di raggruppare intuitivamente degli esami i cui risultati potrebbero essere correlati in qualche modo. L'ovvio rischio che si è scelto di correre è quello di non notare correlazioni che esistono, ma che non sono intuitive.

La suddivisione più sensata è sembrata essere la seguente:

- esami del primo anno
- esami su argomenti principalmente informatici
- esami su argomenti principalmente matematici

Esami del primo anno

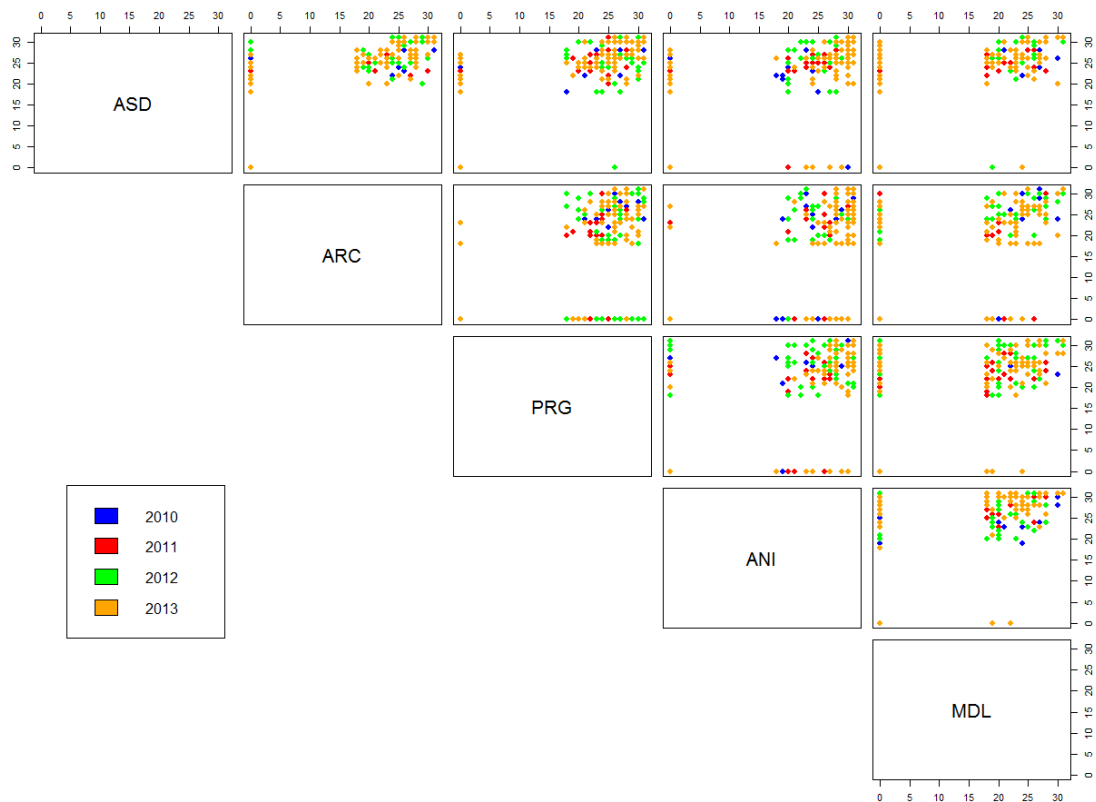
Il seguente script **R** ha generato i grafici di dispersione, le matrici di correlazione e di deviazione standard utilizzate per l'analisi visiva del gruppo di esami del primo anno. Per gli altri gruppi di esami, gli script sono stati molto simili.

```

1 # general first year exams performances
2 students_subset2 <- students[, -c(1 : 5, 7, 9, 11, 13, 15 : 45)]
3 pairs(students_subset2, col = coorte_colors, lower.panel =
  NULL, cex.labelsiris=2, pch=19, cex = 1.2)
4 par(xpd = TRUE)
5 legend(x = 0.05, y = 0.4, cex = 1, legend =
  as.character(levels(coorte_labels)), fill =
  unique(coorte_colors))
6 par(xpd = NA)
```

² non si ignorino però le istanze considerate *outliers* dall'algoritmo!

Figura 13: grafici di dispersione riguardanti gli attributi relativi agli esami del primo anno



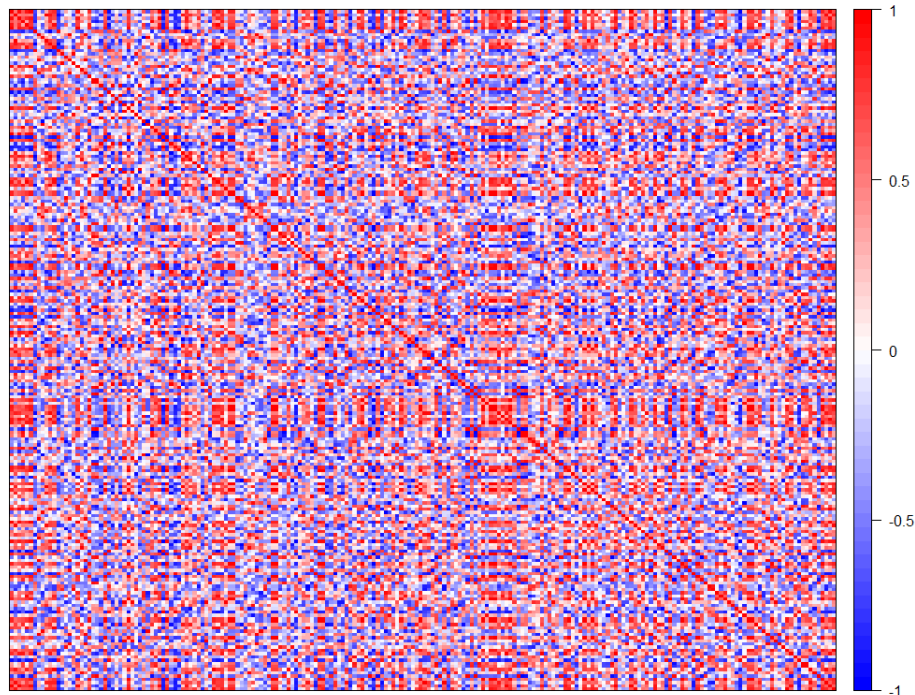
```

7
8 students_scaled <- scale(students_subset2)
9 pimage(students_scaled, ylab="Students", main="Standard Deviations
  from Mean Mark")
10
11 matrix <- as.matrix(students_scaled)
12 cm <- cor(t(matrix), method="pearson")
13 pimage(cm, main="Correlation Matrix considering 1st Year exams",
  xlab="Students", ylab="Students", zlim = c(-1,1), col =
  bluered(50))

```

Dall'analisi visiva del grafico di dispersione in Figura 13 non si è stati in grado di concludere molto di concreto. Eppure si ritiene possibile che qualche informazione interessante si possa nascondere dentro l'insieme

Figura 14: matrice di correlazione fra le istanze degli studenti, le cui righe e colonne rappresentano i vettori composti dai voti sugli esami del primo anno



di dati degli esami del primo anno; occorre scendere nel dettaglio. A questo proposito, sono state realizzate le matrici delle Figure 14 e 15.

Per quanto riguarda invece quanto mostrato in Figura 14, come misura di correlazione è stata scelta la *correlazione di Pearson*, ricercando quindi relazioni monotone — vengono considerati simili studenti che hanno un andamento simile su un gruppo sufficientemente grande di esami, senza contare eventuali *offset* fra le valutazioni. Si nota un accenno di trama in due punti, sintomo di similarità fra due fasce di studenti. Questo può far pensare che potrebbe essere possibile individuarle con algoritmi di *clustering*.

Osservando invece la Figura 15, si può evidenziare che i colori più tiepidi della deviazione standard per gli esami di *Matematica Discreta e Logica* e *Architetture degli Elaboratori* stanno ad indicare una minore tendenza a discostarsi dal voto medio. Perché studenti di vari anni tendono

Figura 15: matrice dello scarto quadratico medio dei risultati degli esami da sostenere il primo anno

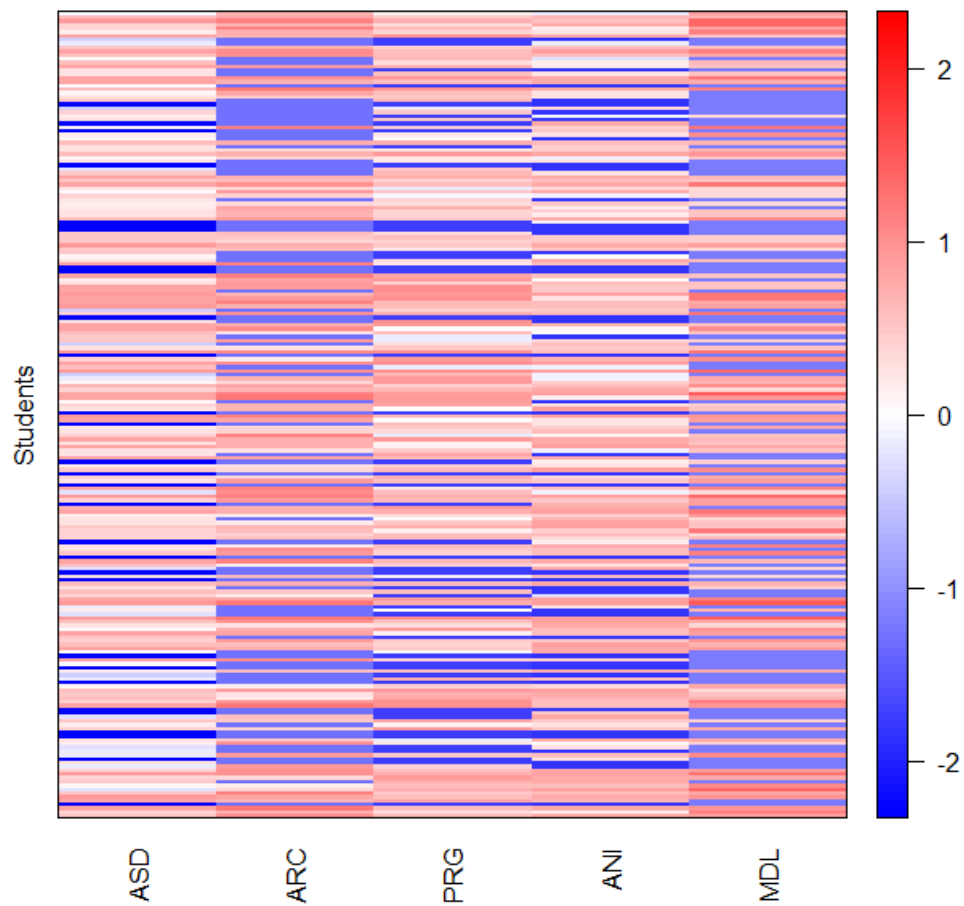


Figura 16: sommario dell'attributo "data" relativo agli esami del primo anno

data_ASD	data_ARC	data_PRG	data_ANI	data_MDL
0 : 30	0 : 77	0 : 47	0 : 46	0 : 85
2014-07-17: 27	2014-02-28: 9	2014-09-03: 20	2015-07-14: 16	2015-02-05: 6
2013-07-11: 15	2014-07-24: 9	2013-07-18: 17	2014-07-23: 14	2013-07-03: 5
2012-07-11: 13	2014-07-10: 8	2013-06-20: 11	2015-07-07: 13	2013-07-30: 5
2013-09-19: 13	2014-09-15: 8	2014-02-04: 11	2016-06-21: 8	2013-09-11: 5
2014-06-11: 12	2015-02-13: 8	2015-01-08: 9	2013-07-01: 7	2014-01-14: 5
(Other) :102	(Other) :93	(Other) :97	(Other) :108	(Other) :101

a convergere verso lo stesso voto in quei due esami? Per rispondere a questa domanda, si veda in Figura 16 un sommario dell'attributo *data*.

Gli esami che non sono stati superati dal maggior numero di istanze dell'intero data set — e che quindi hanno creato maggiore difficoltà agli studenti — sono Matematica Discreta e Logica e Architetture degli Elaboratori. Si è inoltre notato nella matrice della derivazione standard mostrata in Figura 15 che quei due esami presentano una differenza rispetto agli altri. Vale la pena di indagare oltre.

Si veda il grafico di dispersione specifico per Matematica Discreta e Logica in Figura 17: si riesce a notare una blanda tendenza dei Crediti Formativi Universitari ottenuti in totale dallo studente ad aumentare proporzionalmente al voto ottenuto. C'è inoltre una chiara indicazione che molti studenti — con quelli della coorte 2012 a estremizzare questa caratteristica — hanno ottenuto comunque molti CFU senza superare questo esame. Inoltre, incrociando il voto conseguito e la data in cui l'esame è stato superato (si veda la Figura 18), si riescono a notare due aspetti:

- chi ha dato l'esame al primo appello del suo anno, ha conseguito un voto alto (si vedano ad esempio gli studenti dell'annata 2010)
- la maggioranza degli studenti ha superato Matematica Discreta e Logica qualche anno dopo il suo anno di immatricolazione (comportamento estremizzato dagli studenti di coorte 2012)

Nel caso di Architetture degli Elaboratori, della quale possiamo vedere un grafico di dispersione dettagliato in Figura 19, le tendenze evidenziate prima sono mitigate. Rimane comunque importante la quantità di studenti che hanno conseguito molti crediti senza superare l'esame: si potrebbe

Figura 17: grafico di dispersione fra gli attributi "voto ottenuto a Matematica Discreta e Logica" e "CFU ottenuti nel periodo in esame"

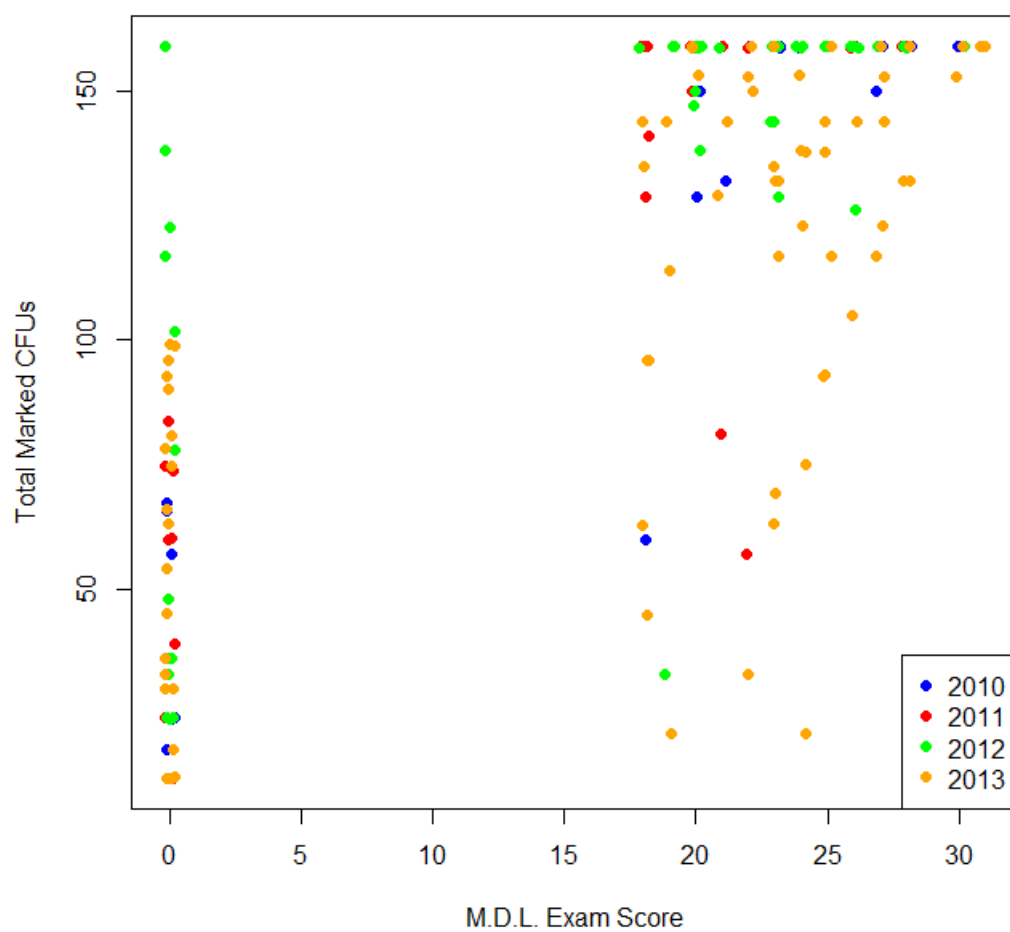


Figura 18: grafico di dispersione fra gli attributi "voto" e "data di superamento" relativi all'esame di Matematica Discreta e Logica

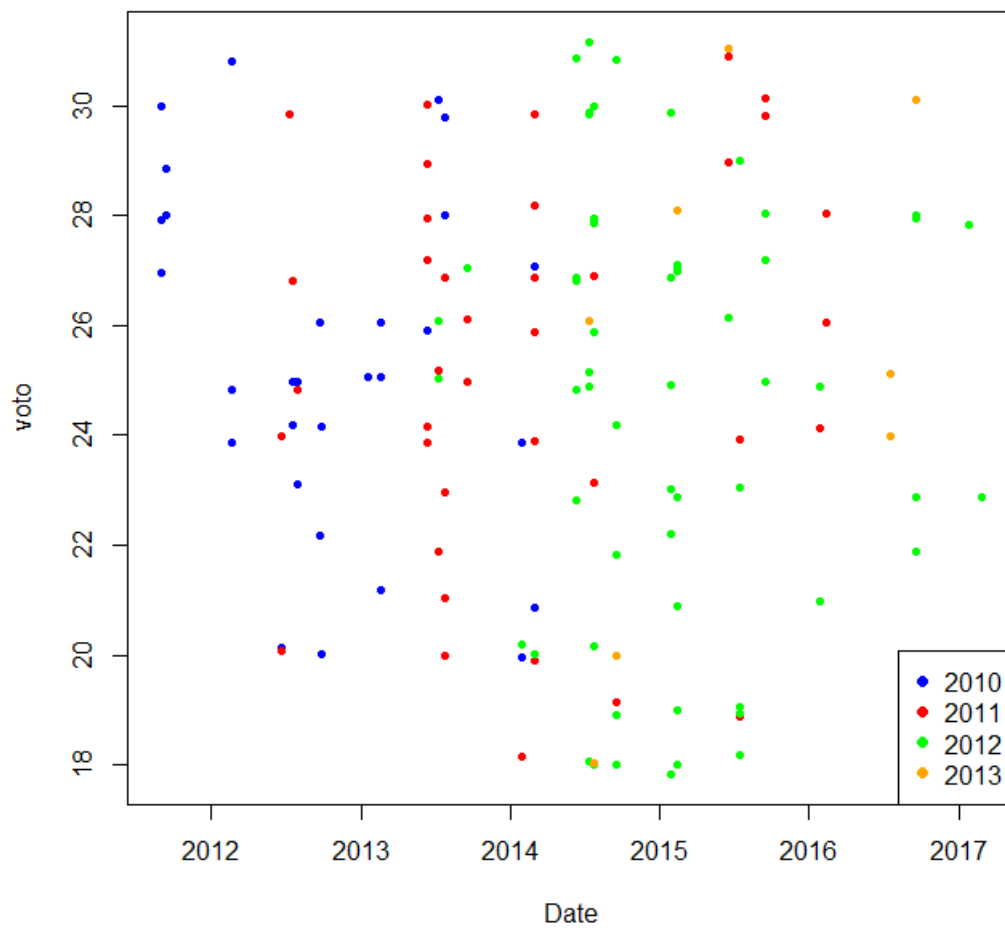


Figura 19: grafico di dispersione fra gli attributi "voto ottenuto a Architetture degli Elaboratori" e "CFU ottenuti nel periodo in esame"

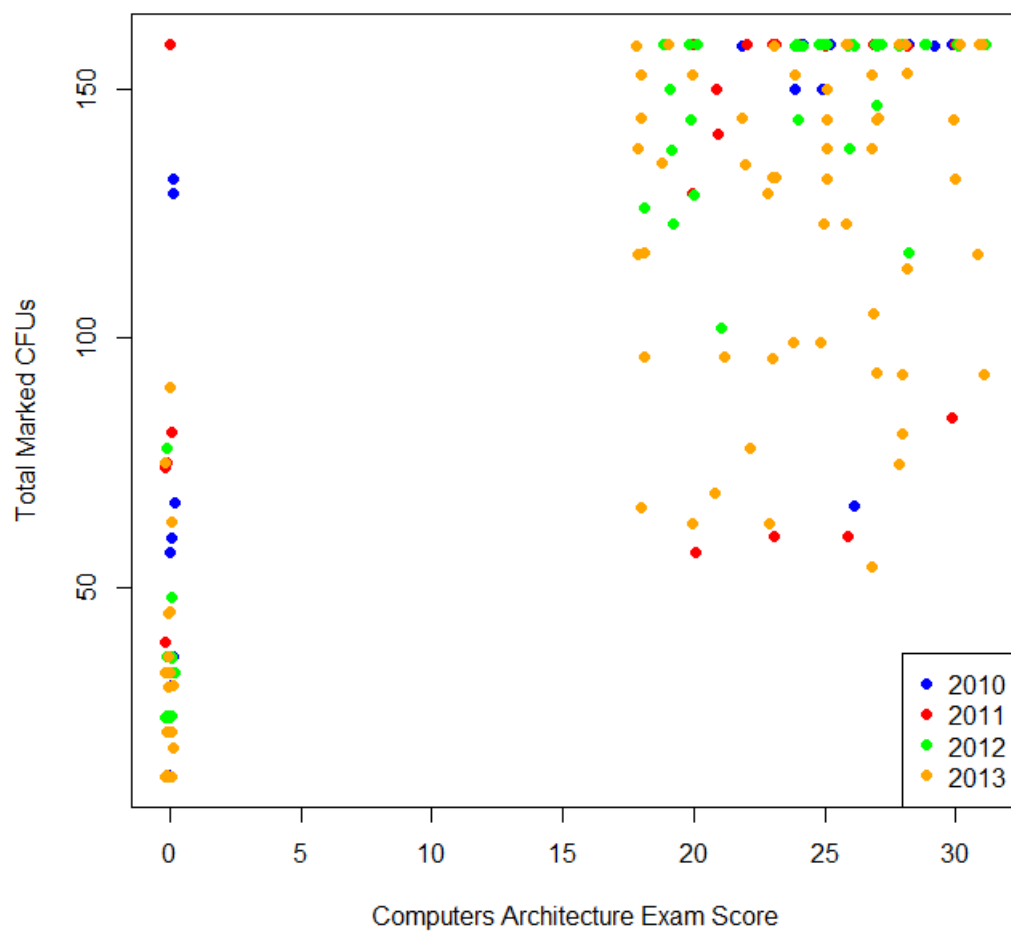
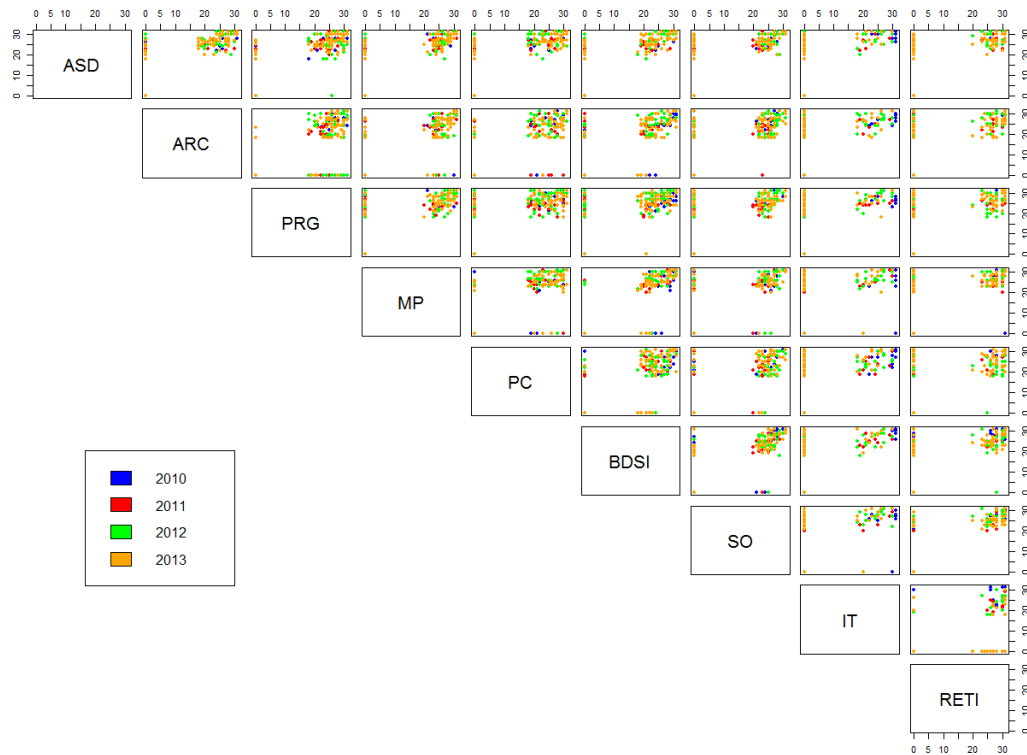


Figura 20: grafici di dispersione relativi agli esami a contenuto prevalentemente informatico



speculare che l'ignorare certi esami sia una pratica comune nel pool di studenti descritti dal data set.

Esami su argomenti informatici

Riguardo agli esami a contenuto prevalentemente informatico, sono state effettuate analoghe analisi visive. Poco è stato notato dal grafico di dispersione in Figura 20 e dalla matrice di correlazione in Figura 21. Invece, dalla matrice della deviazione standard in Figura 22 è stato possibile trarre un'interessante deduzione: con l'avvicinarsi agli esami del terzo anno, si nota che il voto conseguito da ogni studente tende a discostarsi sempre meno dalla media.

In ogni caso, non appare interessante proseguire sull'analisi con tecniche di *data mining* su questa suddivisione.

Figura 21: matrice di correlazione fra gli studenti, le cui righe e colonne rappresentano i vettori composti dai voti sugli esami a contenuto informatico

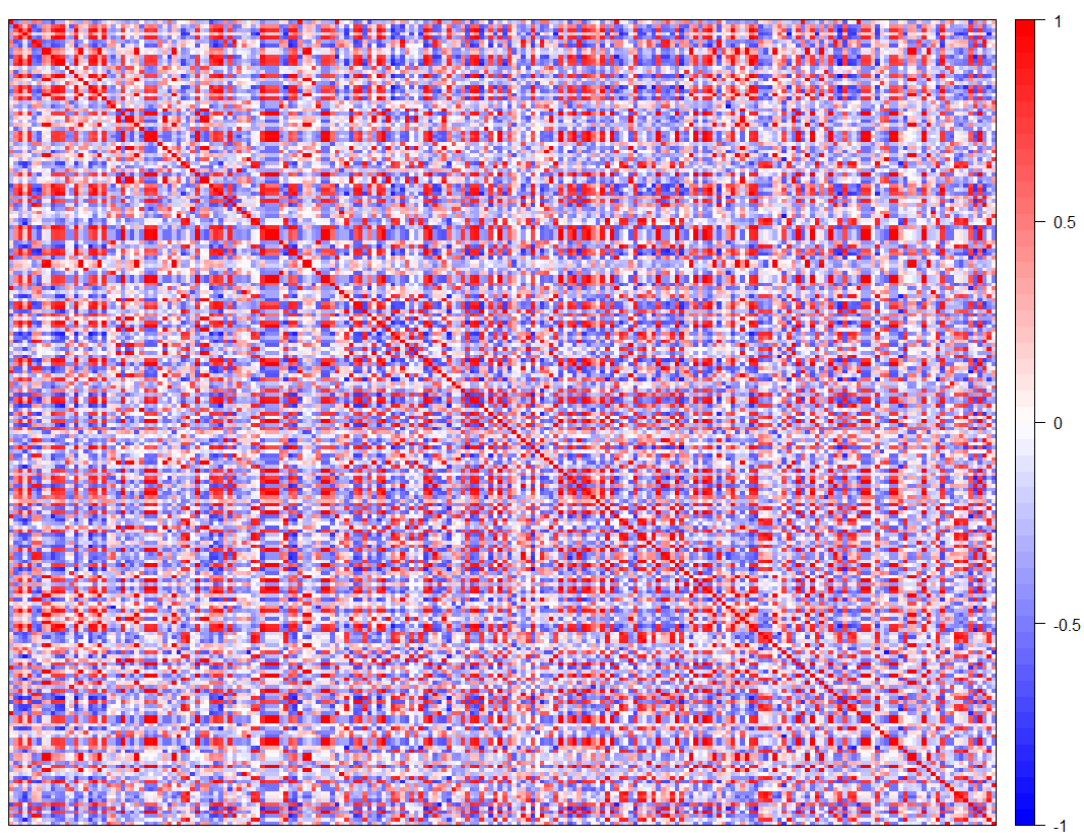


Figura 22: matrice dello scarto quadratico medio dei risultati degli esami ad argomento informatico

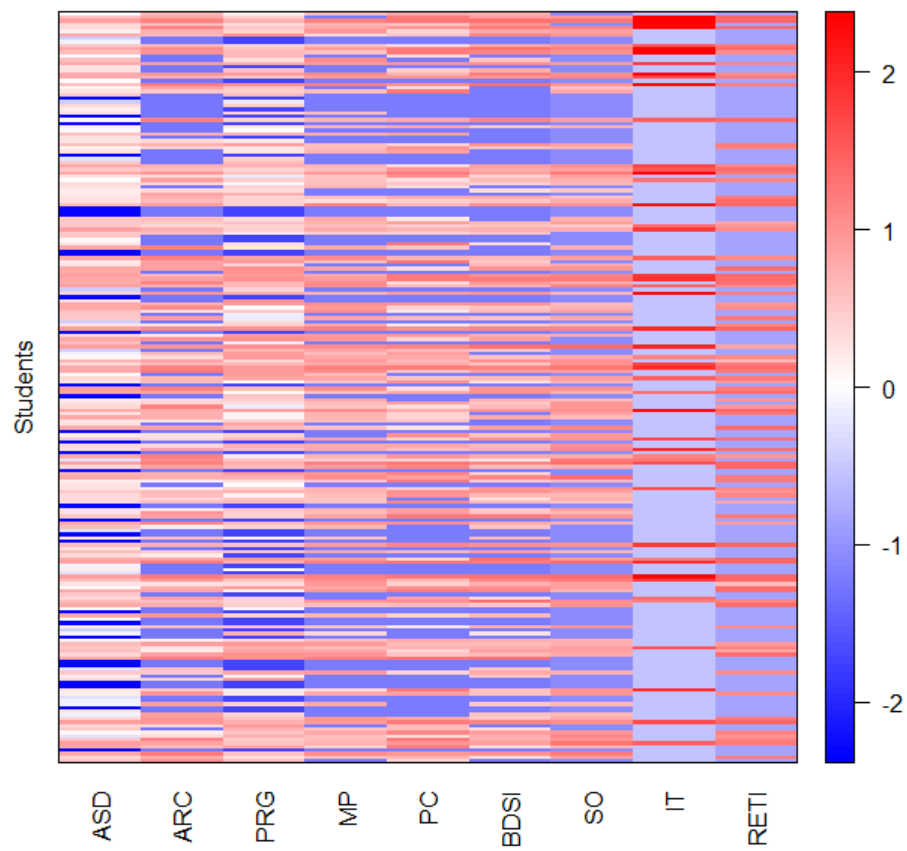
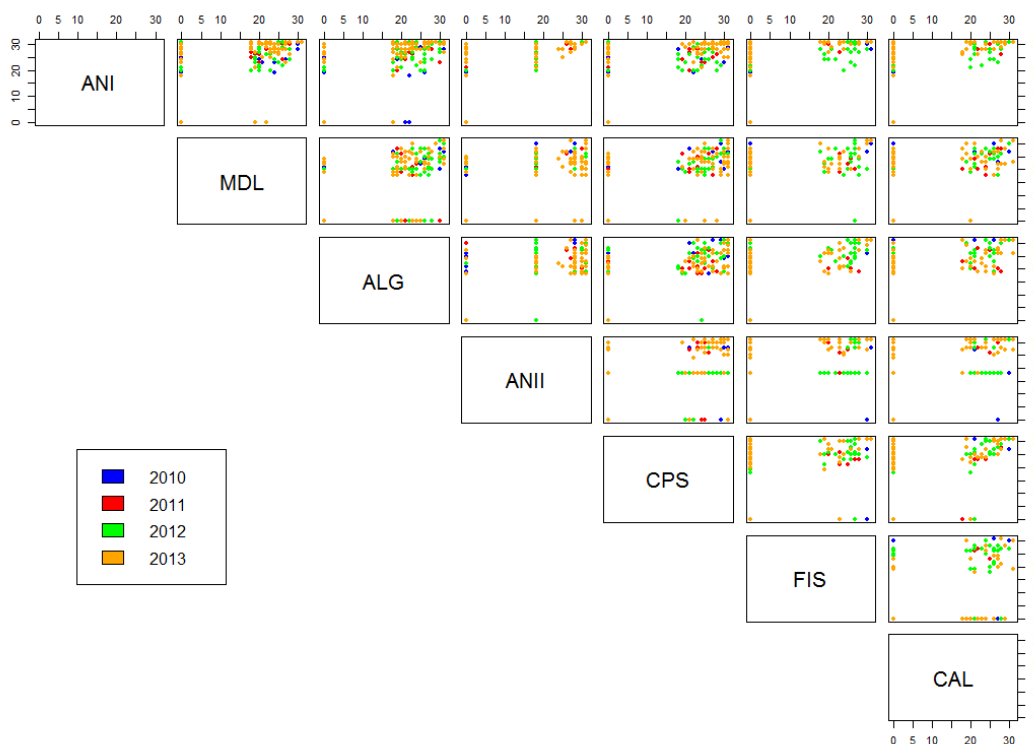


Figura 23: grafici di dispersione relativi agli esami ad argomento prevalentemente matematico



Esami su argomenti matematici

Riguardo a questa partizione degli esami, di cui si può vedere un grafico di dispersione generale in Figura 23, un aspetto che balza immediatamente alla vista è l'enorme quantità di studenti che ha conseguito 18 ad Analisi 2. Date le modalità di esame previste per il corso³, questa non è affatto un'anomalia, pertanto non appare interessante; inoltre, potrebbe addirittura risultare distorta rispetto ad una ipotetica *cluster analysis*, perciò occorrerebbe considerarla preventivamente.

Sulla matrice di correlazione in Figura 24, si notano due aspetti interessanti: una fascia di studenti con correlazione nulla e una «casella» di forti correlazioni all'incirca nel centro della matrice. Dei due, il primo è

³ Se le prove intermedie scritte risultano sufficienti, si può accettare un voto di 18 senza sostenere la prova orale.

Figura 24: matrice di correlazione fra gli studenti, le cui righe e colonne rappresentano i vettori composti dai voti sugli esami a contenuto matematico

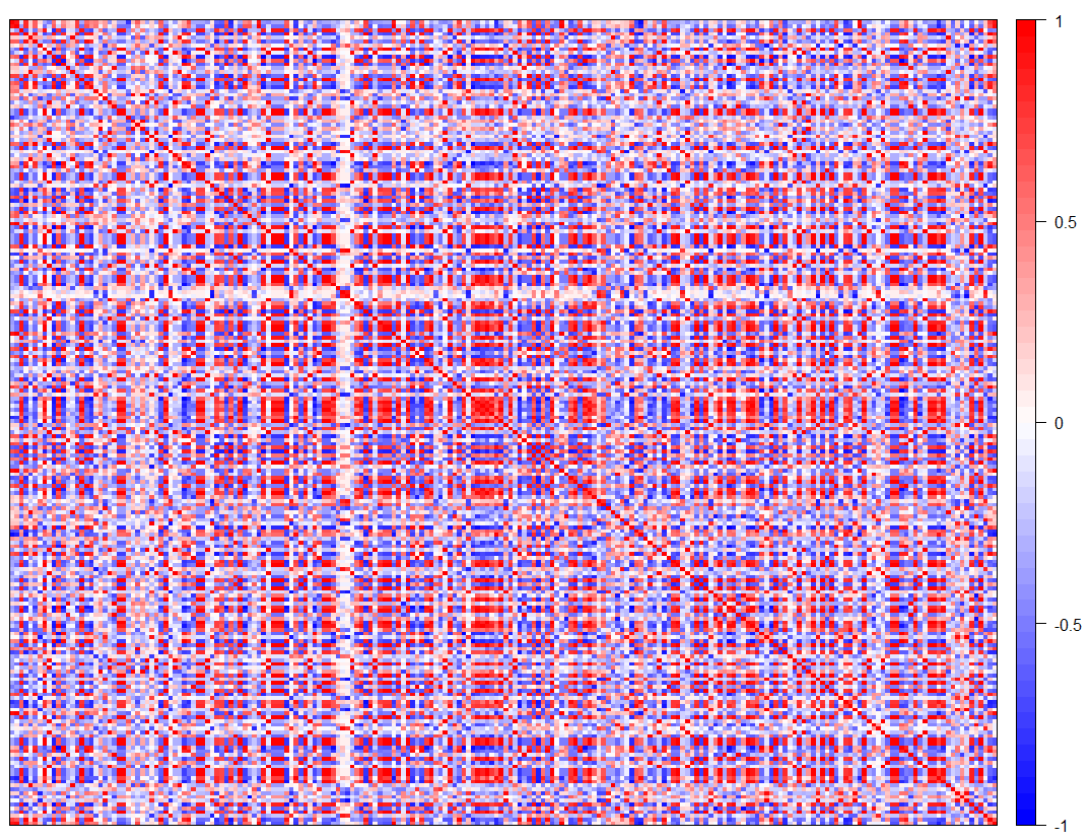
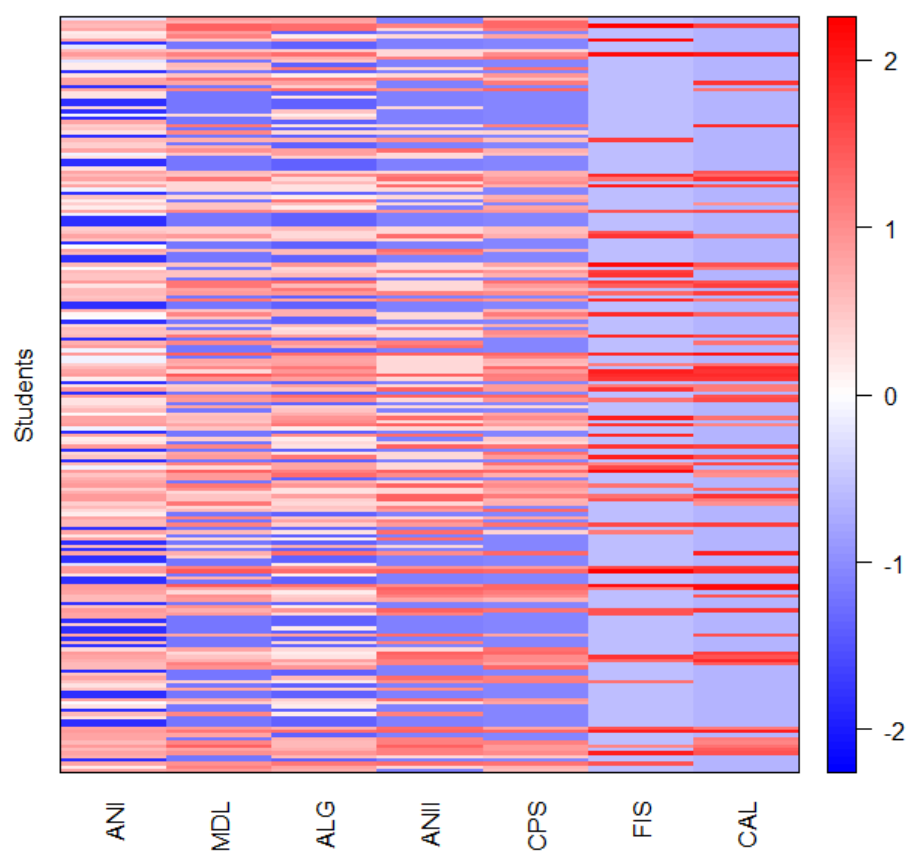


Figura 25: matrice dello scarto quadratico medio dei risultati degli esami a contenuto matematico



sicuramente il più inusuale. Quale aspetto reale potrebbe generare una osservazione come questa?

Riguardo alla matrice della deviazione standard in Figura 25, si possono notare le stesse caratteristiche evidenziate per quanto riguarda gli esami a tema principalmente informatico. In questo caso, il ruolo del cattivo non lo interpreta più Informatica Teorica ma è condiviso da Fisica Generale e Calcolo Numerico⁴.

Analogamente alla suddivisione precedente, anche per questa partizione di esami non appare interessante proseguire l'analisi con tecniche di *data mining*.

4.2.3 Conclusioni

Da quello che abbiamo visto, una *cluster analysis* mirata alle prestazioni generali degli studenti potrebbe dare buoni risultati. Inoltre, come si è potuto intuire dal caso di Matematica Discreta e Logica, questo data set racchiude importanti informazioni nelle coppie di valori (data, voto): si potrebbe pensare di estrarre in qualche senso dei *pattern sequenziali* relativi all'ordine in cui vengono superati i vari esami dell'intero Corso di Laurea.

Inoltre, è apparso chiaro anche un altro aspetto: osservare il comportamento dei dati istanza per istanza è stato senza dubbio utile, ma in un data set come questo, in cui gli studenti appartengono a una precisa coorte di immatricolazione, può essere importante catturare gli aspetti fondamentali di ciascuna *classe di record* — appunto, la coorte di immatricolazione. Si potrebbe quindi pensare di effettuare una massiccia aggregazione di tutti i dati per osservare le differenze fra le classi.

4.3 DATA SET: VALUTAZIONI DEGLI INSEGNAMENTI

Sebbene questa porzione di dati a disposizione sia quella dalla mole più elevata, si è ritenuto che essa possa esprimere la sua maggiore utilità

⁴ Calcolo Numerico tratta argomenti che trascendono i confini fra l'informatica e la matematica. Si è scelto d'inserirlo in questa partizione, ma è stata una decisione puramente arbitraria.

se impiegata insieme all'altro dataset che abbiamo deciso di considerare principale. Questo comporterà ovviamente la necessità di effettuare una **join** di qualche tipo fra i due insiemi di dati.

Inoltre, si ribadisce che si tratta di dati sì aggregati, ma coprenti comunque un ampio spettro di aspetti di ogni corso. Pertanto, effettuare un'analisi approfondita sullo stile di quella fatta sul data set degli studenti darebbe risultati dispersivi e di difficile interpretazione.

Ci si riserva quindi di adottare un approccio non convenzionale: ritardare l'analisi visiva di questo data set, effettuandola **dopo** una massiccia aggregazione eseguita nella fase di *preprocessing*. A questo proposito, rimangono valide le considerazioni espresse per il dataset degli studenti riguardo all'osservare il comportamento di ciascuna *classe di record* — in questo caso, l'Anno Accademico di riferimento.

4.4 SCELTA DELLE TECNICHE DA IMPIEGARE

Come risultato ultimo di questa fase, si è stati in grado di decidere il tipo di tecniche di *preprocessing* e *data mining* che sarà opportuno utilizzare per proseguire lo studio di questi dati.

Si prevede di utilizzare queste tecniche:

- *tecniche di visualizzazione* con focus sulle *classi di record* dei due data set;
- *cluster analysis* sulle prestazioni generali degli studenti;
- individuazione dei *pattern sequenziali frequenti* nell'ordine di superamento degli esami;
- *cluster analysis* sulla relazione fra prestazioni degli studenti e valutazioni dei corsi;
- ricerca di *regole associative* fra i risultati conseguiti in un dato esame e la valutazione del suo corso.

Al fine di utilizzarle al meglio, occorre preparare nella fase di *preprocessing* i seguenti data set:

- prestazioni generali degli studenti aggregate per coorte di immatricolazione — *visualizzazione*;
- valutazione degli insegnamenti aggregate per Anno Accademico — *visualizzazione*;
- join dei due insiemi di dati con attributi continui — *clustering*;
- join dei due insiemi di dati con attributi discreti — *analisi associativa*;
- sequenze ordinate di esami superati — *pattern sequenziali*.

Altre *cluster analysis* sono poi state effettuate direttamente sul dataset della carriera degli studenti con minima preparazione, essendo esso già predisposto all'utilizzo con questo tipo di algoritmi.

PREPROCESSING

Quella che sarà descritta in questo capitolo è sicuramente la fase più impegnativa e delicata dell'intero lavoro. Vista quindi l'importanza che l'attività di *preprocessing* ha rivestito, è stato scelto di descriverla con un elevato livello di dettaglio, evidenziando passaggio per passaggio le operazioni necessarie per dare all'insieme di dati grezzi una forma adeguata al tipo di analisi che ci si è prefissati di fare.

Nell'illustrare i vari procedimenti, per favorire una spiegazione lineare e il più possibile comprensibile, si impiegherà ancora la metafora della lavorazione meccanica, intesa in questo caso come una sgrossatura volta ad ottenere un semilavorato — il data set *preprocessato*, pronto per essere ulteriormente lavorato con le tecniche di *data mining*. Ricordando quanto affermato nell'introduzione della sezioni precedenti, i dati iniziali rappresentano il pezzo grezzo da lavorare, mentre le tecnologie scelte gli utensili da impiegare nella sgrossatura.

5.1 PREPARAZIONE DELL'AMBIENTE DI LAVORO

5.1.1 Ottenere gli strumenti necessari

Innanzitutto è necessario predisporre gli utensili necessari al lavoro da svolgere — fuor di metafora, si tratta di installare i programmi necessari al *preprocessing*. Come abbiamo detto nella sezione dedicata alla *technology stack*, abbiamo bisogno del *d.b.m.s.* MongoDB e del suo *driver*¹.

¹ Il termine *driver* non è perfettamente proprio per descrivere quella che in realtà è una semplice implementazione in Python delle *A.P.I.* di MongoDB, ma colloquialmente rende bene l'idea della funzione di *pymongo*.

La piattaforma impiegata è un personal computer con sistema operativo Arch Linux, perciò occorrerà installare MongoDB su di essa. Questo può essere fatto in modo estremamente agile, scaricando i pacchetti `mongodb` e `mongodb-tools` dalle repository ufficiali con il seguente comando:

Listing 5.1: installazione di MongoDB

```
1 sudo pacman -S mongodb mongodb-tools --noconfirm
```

Per ottenere `pymongo`, invece, occorre utilizzare il *package manager* di Python, `pip`, invocandolo semplicemente come segue:

Listing 5.2: installazione di pymongo

```
1 pip install pymongo
```

A questo punto disponiamo degli utensili necessari per la nostra lavorazione.

5.1.2 Inizializzazione di un server MongoDB

Predisposti gli utensili, occorre adesso avviare la macchina e montare il pezzo — in una *vera* lavorazione meccanica, ovviamente *prima* si monta l’utensile e si posiziona il pezzo, *poi* si avvia la macchina; in questo caso, occorre avviare prima un processo di MongoDB affinché si possano importare i dati grezzi in uno *schema* ed organizzarli in *collections*, per poi lavorarli tramite `pymongo`.

MongoDB fornisce un database estremamente veloce, e di default utilizza come supporto fisico di memorizzazione una cartella sul disco di installazione. La macchina utilizzata dispone di un disco a stato solido come unità di memoria non volatile, perciò le velocità di lettura e scrittura nel database di MongoDB risulterebbero ottime anche nella configurazione standard. Tuttavia, sia per migliorare ulteriormente le performances delle operazioni che per preservare la vita del disco², è stato scelto di

² Le celle di memoria degli SSD, o dischi a stato solido, possono sopportare un numero limitato di scritture prima di rovinarsi.

creare un *ramdisk*³ da far utilizzare a MongoDB.

Per pura comodità, le operazioni necessarie per realizzare quanto appena descritto sono state delegate ad uno script:

Listing 5.3: script di lancio di un server MongoDB

```
1      #!/bin/zsh
2      sudo killall mongod
3      yes | rm -rf /mnt/ramdisk/db
4      mkdir db /mnt/ramdisk/db
5      mongod --dbpath=/mnt/ramdisk/db
```

Nella *shell* con cui è stato lanciato, si può vedere lo *standard output* del processo *mongod*. Significa che MongoDB è attivo ed invocabile tramite gli strumenti a nostra disposizione.

5.1.3 Importazione dei Dati Grezzi

A questo punto, sia la macchina che gli utensili sono pronti: occorre posizionare il pezzo grezzo da lavorare, ovvero importare i dati in MongoDB.

Come visto nel capitolo precedente, i dati a disposizione sono contenuti in otto file *csv*: si manterrà questa struttura — almeno inizialmente — importando quindi ogni file in una sua *collection*. Come si può vedere di seguito, questa operazione è stata descritta nel *makefile* con l'etichetta *import*:

```
27  reset_db:
28      mongo $(DB) --eval "db.dropDatabase()"
29
30  import: reset_db
31      $(PRDIR) sh import.sh
```

³ Si tratta di un *filesystem* implementato in un'area di RAM; è una tecnica per velocizzare estremamente le operazioni di lettura e scrittura, ma dato che il *filesystem* è implementato su memoria volatile, i dati scritti in esso vengono persi dopo lo spegnimento della macchina.

Le variabili \$(DB) e \$(PRDIR) sono specifiche dell'ambiente predisposto⁴. Essa si può lanciare con il seguente semplice comando di shell:

Listing 5.4: importazione dei dati in MongoDB

```
1 make import
```

La vera e propria invocazione del comando necessario all'import dei file nel database — `mongoimport`, del pacchetto `mongodb-tools` — è stata delegata in un file esterno al `makefile`, per preservare la compattezza e la brevità di quest'ultimo. Si veda comunque di seguito come è strutturata la chiamata a `mongoimport` per uno degli otto file:

Listing 5.5: dettaglio dell'importazione dei dati in MongoDB

```
1 mongoimport -d exams -c rawStudentsPr1013 --type csv --file
  ../raw/prod_stud_10-11-12-13.csv --headerline
```

Il comando, come ogni chiamata da shell, ha degli argomenti che ne specificano il comportamento. In questo caso, sono:

- `-d`: il database nel quale importare i dati;
- `-c`: la collection nella quale inserire i dati;
- `-type`: il tipo del file da leggere;
- `-file`: il riferimento al file;
- `-headerline`: indica che gli attributi delle istanze sono specificati nella prima riga del file.

A questo punto, nel database `exams` di MongoDB ci sono otto *collections*, contenenti i *documenti* che rappresentano le istanze dei dati a disposizione.

5.2 AGGREGAZIONE PER ANNO ACCADEMICO

Avendo predisposto tutto, si può procedere con la prima lavorazione da fare.

⁴ Per un esempio pratico, si consulti l'intero `makefile` riportato nell'Appendice A.6.

Si comincia intanto con l'ottenere dei data set *minimali*, uno per le valutazioni dei corsi, l'altro per la produttività degli studenti, condensando in essi le informazioni principali contenute in quello delle prestazioni degli studenti in pochi parametri relativi ad un certo anno accademico.

5.2.1 *Produttività degli Studenti*

L'obiettivo è sintetizzare le informazioni sulla produttività degli studenti nei seguenti attributi:

- coorte di immatricolazione;
- numero di studenti totali;
- percentuale di studenti laureati;
- valutazione media ottenuta al test di ingresso;
- voto medio ottenuto agli esami;
- ritardo medio con cui è stato dato un esame.

Considerando il tipo dei dati a disposizione, occorrerà innanzitutto aggregare le istanze dei vari studenti in modo opportuno. Questo viene fatto utilizzando un modulo Python programmato *ad hoc*, del quale si riporta il codice nell'Appendice A.2.

L'oggetto `StudAggregator` definito in tale modulo viene impiegato per realizzare una prima aggregazione per corsi dei risultati dei singoli studenti. Si può usare questo risultato intermedio per ricavare le informazioni prefissate come necessarie, usando un altro apposito script Python riportato nell'Appendice A.5.3. Si è così prodotta una *collection* che contiene esattamente il data set che ci si è prefissati di ottenere, calcolando un ritardo approssimativo medio⁵, la percentuale di studenti che hanno finito il corso di laurea nell'arco temporale a disposizione e le varie medie degli altri attributi.

⁵ L'attributo esprime la media, espressa in *semestri* della differenza fra la data di superamento di ogni esame e il suo primo appello disponibile.

Coorte	N.	Laureati [%]	Test Ingr. [0-25]	Voto [18-31]	Ritardo [sem.]
			<i>media</i>	<i>media</i>	<i>media</i>
2010	30	6.67	15.4	25.5	0.81
2011	39	10.26	13.26	24.81	1.07
2012	58	25.86	14.05	24.79	1.01
2013	80	11.25	14.39	24.98	0.77

Il lancio di tutte queste operazioni è specificato nella ricetta `stud_gen` del `makefile`.

5.2.2 Valutazione degli Insegnamenti

Analogamente a quanto fatto nella sezione immediatamente precedente, per questa famiglia di dati si vuole ottenere un'aggregazione che riassume i seguenti attributi:

- anno accademico
- numero di valutazioni registrate
- valutazione complessiva media dei corsi
- deviazione standard delle valutazioni
- percentuale delle valutazioni sufficienti

In questo caso è tutto più semplice, in quanto i dati relativi alla valutazione dei corsi sono già in forma aggregata: si tratta quindi solo di comprimerli ulteriormente, facendoli rientrare nello schema che ci si è prefissati.

A tale proposito, si utilizzerà ancora il modulo `aggregs.py`, impiegandone stavolta un oggetto diverso: `ParAggregator`. Esso mantiene nella *chiave primaria* i corsi, pertanto occorre aggregare ancora i dati, rendendo l'Anno Accademico l'unico parametro in grado di discriminare una tupla dall'altra. Tale operazione viene eseguita con un apposito script, riportato nell'Appendice A.5.2.

Il risultato di quanto appena descritto, il cui lancio è stato specificato nella ricetta `teval_gen` del `makefile`, produce un data set di questo tipo:

A. A.	N.	Valutazione [1-10]	Std. Dev. Val.	Val. Sufficienti [%]
		<i>media</i>	<i>media</i>	
2010-2011	17	7.54	1.74	82.14
2011-2012	26	7.93	1.61	90.68
2012-2013	30	7.98	1.7	90.55
...

5.3 JOIN DEI DUE INSIEMI DI DATI

Il *join* delle due famiglie di dati è l'operazione più delicata fra quelle di tutto il *preprocessing*. Occorre definire bene gli attributi sui quali definire la relazione, e prestare in generale attenzione ai vari errori che, se commessi, comprometterebbero totalmente il significato del risultato.

5.3.1 *Join con valutazioni estese e attributi continui*

Per prima cosa, si è realizzata una *join* fra i due data set aggregando i dati degli studenti **per esame**, e i dati delle valutazioni dei corsi **per paragrafo**. Osservando l'albero delle dipendenze specificato nel `makefile` e mostrato in Figura 26 — che, si ripete, si può consultare per intero nell'Appendice A.6 — è possibile notare che sono state prima realizzate le aggregazioni dei singoli data set e, solo successivamente, performata l'operazione di *join* fra di essi.

L'aggregazione dei dati degli studenti avviene in modo del tutto analogo a quanto fatto nella sezione precedente con l'oggetto `StudAggregator` del modulo `Python aggregs.py`.

Per quanto riguarda l'aggregazione dei dati delle valutazioni dei corsi, sono stati effettuati vari passaggi per arrivare a condensare le informazioni nei pochi attributi generali necessari. Tali passaggi sono visibili consultando nell'Appendice A.5.2 il codice con cui sono stati realizzati.

Predisposti opportunamente i due insiemi di dati in due *collection*, la fase di merge è stata portata avanti grazie all'ausilio di un altro modulo Python scritto *ad hoc*: si tratta dell'oggetto *Merger*, contenuto nel file *merge.py*, la cui implementazione è visibile nell'Appendice A.4. Tale oggetto viene utilizzato nello script *dataset_merge.py*. Il *join* avviene, chiaramente, sugli attributi **Anno Accademico** e **Corso d'Esame**, trasferendo gli attributi specifici di una collezione direttamente nell'altra.

A questo punto, la collezione risultante da queste operazioni ha le caratteristiche cercate. Tuttavia, presenta delle "sbavature" che è conveniente rimuovere prima di impiegarla efficacemente nel *data mining*. Per eliminarle, ci si avvale di un ulteriore modulo Python realizzato per l'occasione, *cleanings.py*, riportato per intero nell'Appendice A.1.

Concluso quanto appena descritto, si è creato un primo data set utilizzabile. Esso presenta i seguenti attributi, tutti di tipo *continuo* — tranne ovviamente le *chiavi primarie* e le istanze considerate per la produttività degli studenti:

- Anno Accademico
- Hash Docente/i
- Insegnamento
- **Produttività Studenti:**
 - N [*istanze*]
 - Ritardo ≥ 1 sem [*percent*]
 - Ritardo [*semestre, media*]
 - Voto ≥ 24 [*perc*]
 - Voto [*media*]
 - Voto [*std dev*]
- **Valutazione degli Aspetti Specifici dell'Insegnamento:**
 - N [*istanze*]
 - Std Dev [*media pesata*]
 - Val ≥ 6 [*percent*]

Figura 26: porzione del makefile che specifica le ricette per eseguire il *join* fra le valutazioni dei corsi e i dati degli studenti

```
33 #
34 # teaching evaluation recipes
35 #
36 teval: teval_merge
37
38 teval_clean: import
39 | $(PRDIR) $(TIME) $(PY) teval_clean.py
40 |
41 teval_aggr: teval_clean
42 | $(PRDIR) $(TIME) $(PY) teval_aggr.py
43 |
44 teval_merge: teval_aggr
45 | $(PRDIR) $(TIME) $(PY) teval_merge.py
46 |
47 teval_gen: teval_aggr
48 | $(PRDIR) $(TIME) $(PY) dataset_eval_gen.py
49 |
50 #
51 # students productivity recipes
52 #
53 stud: stud_aggr
54
55 stud_aggr: import
56 | $(PRDIR) $(TIME) $(PY) stud_aggr.py
57 |
58 stud_gen: stud_aggr
59 | $(PRDIR) $(TIME) $(PY) dataset_stud_gen.py
60 |
61 stud_seq: import
62 | $(PRDIR) $(TIME) $(PY) dataset_stud_seq.py
63 |
64 #
65 # finalize
66 #
67 merged: stud teval
68 | $(PRDIR) $(TIME) $(PY) dataset_merge.py
```

Val [*media pesata*]

- **Valutazione dell'adeguatezza delle Aule e delle Attrezzature:**
attributi identici alla valutazione precedente
- **Valutazione sul Docente:**
attributi identici alla valutazione precedente
- **Valutazione sulla Disponibilità di Informazioni Aggiuntive:**
attributi identici alla valutazione precedente
- **Valutazione dell'Organizzazione dell'Insegnamento:**
attributi identici alla valutazione precedente
- **Valutazione dell'Organizzazione della Didattica:**
attributi identici alla valutazione precedente
- **Soddisfazione degli Studenti riguardo al Corso di Studi:**
attributi identici alla valutazione precedente

Come si può notare agilmente, il data set presenta un gran numero di attributi, molti fin troppo specifici per poterne estrarre un qualche tipo di informazione generale. Pertanto, è stato deciso aggregare ulteriormente fra loro gli attributi relativi alle valutazioni dei corsi, per sfoltire significativamente la quantità di campi presenti.

5.3.2 *Join con valutazioni aggregate e attributi continui*

Lavorando direttamente con uno script Python sulla collezione appena creata, si è provveduto a ridurre il numero di attributi del data set aggregando fra loro le informazioni riguardanti i singoli paragrafi della valutazione dei corsi (si veda per un maggior dettaglio l'Appendice A.5.1).

A seguito di queste operazioni, si è ottenuta una *collection* i cui documenti hanno sei attributi riguardanti la produttività degli studenti e quattro riguardo invece alle valutazioni dei corsi, sempre con *chiave primaria* la tupla < Anno Accademico, Insegnamento >.

5.3.3 *Attributi discreti*

Come è stato spiegato alla fine della fase di *data understanding*, c'è bisogno di avere una versione *discretizzata* dei data set dei quali è stata appena descritta la produzione. Questo è stato realizzato impiegando un altro modulo Python, *discretize.py*, la cui implementazione può essere consultata nell'Appendice A.3.

I *range* nei quali far rientrare i valori continui degli attributi sono stati scelti cercando un giusto compromesso fra rappresentatività e sinteticità. Non è stato ovviamente possibile stabilirli a priori: quelli che si possono vedere nella sopra citata appendice sono il risultato di un processo iterativo del tipo *trial and error*, e sono risultati essere il miglior compromesso cercato.

5.4 SEQUENZE ORDINATE DI ESAMI SUPERATI

Per utilizzare l'algoritmo *GSP* messo a disposizione dalla suite di software Weka, c'è bisogno che i dati di input siano in un formato ben preciso. Si tratta del formato *arff*, il cui acronimo sta a significare *Attribute Relationship File Format*, cioè, traducendo quasi letteralmente, formato di file con attributi e relazioni. Le informazioni contenute in file di questo tipo sono memorizzate come *plain text*, pertanto possono essere accedute da praticamente qualunque strumento.

Per dare un'idea della qualità di questo requisito, un data set composto da due transazioni così composte:

tr.1 : {ITEM1, ITEM2}

tr.2 : {ITEM3}

dovrà essere rappresentato in questo modo:

1, ITEM1
1, ITEM2
2, ITEM3

ovvero scomponendo le transazioni in singoli item, indicando per ognuno di essi la transazione di appartenenza secondo il formato: TRANSAZIONE,

ITEM. L'ordine degli item nelle transazioni è rappresentato dalla posizione delle righe nel file, ovviamente dall'alto verso il basso.

Dato che non esistono strumenti di esportazione da MongoDB in grado di creare file di questo tipo, è stato programmato uno script *ad hoc* che, ottenuti i dati degli studenti tramite delle interrogazioni sulla relativa collezione MongoDB, li trasforma in transazioni e li scrive direttamente su un file di testo avente la struttura richiesta.

L'implementazione dello script sopra riferito è contenuta nell'Appendice A.5.4. Come si può notare dal codice, non ci si è limitati a produrre un singolo file contenente l'interezza delle informazioni a disposizione, ma si è provveduto a dividerlo in molte sezioni a seconda di vari criteri. Questo perché *GSP* è un algoritmo che può essere estremamente pesante da eseguire su data set molto grandi: il problema della generazione degli *itemset* frequenti è un problema N_p – Completo⁶, pertanto considerato intrattabile al crescere della dimensione dei dati di ingresso.

Data la natura dell'analisi che ci si è prefissati di fare su questo tipo di dati, è risultato utile impiegare un primo, generale criterio di filtraggio: **si sono considerate solo le sequenze di esami di studenti che hanno completato almeno un terzo del Corso di Laurea**, cioè coloro che hanno conseguito almeno 60 Crediti Formativi Universitari. Questo perché le transazioni contenenti pochi *item* vanno a contribuire all'analisi solo con sequenze corte, diluendo la frequenza di eventuali sequenze più lunghe che possono risultare interessanti.

Stabilito questo principio, sono stati creati quindi i seguenti file, realizzati secondo ulteriori criteri di divisione: la **coorte di immatricolazione** e la **scuola superiore di provenienza**.

- Divisione per coorte di immatricolazione:

seq_stud_2010.arff

seq_stud_2011.arff

seq_stud_2012.arff

⁶ Significa che è un problema risolvibile in tempo polinomiale con algoritmi *non deterministici*, cosa che implica la risolvibilità con algoritmi deterministici in tempo esponenziale.

seq_stud_2013.arff

- Divisione successiva per scuola superiore di provenienza:

seq_stud_2010_ist_tecn.arff

seq_stud_2010_liceo.arff

seq_stud_2011_ist_tecn.arff

seq_stud_2011_liceo.arff

seq_stud_2012_ist_tecn.arff

seq_stud_2012_liceo.arff

seq_stud_2013_ist_tecn.arff

seq_stud_2013_liceo.arff

Inoltre, per puro scrupolo, è stato comunque creato un file contenente la totalità dei dati a disposizione — fermo restando il criterio di filtraggio preliminare.

5.5 ESTRAZIONE DEI DATA SET PREPROCESSATI

Ultimata finalmente la preparazione dei data set necessari alle successive analisi, occorre renderli disponibili in un formato che ne consenta una facile fruibilità. Al di là di necessità specifiche — in particolare, il caso esposto nella sezione precedente — è stato scelto di esportare i dati da MongoDB in formato csv, direttamente importabile in Weka e accessibile in pratica da ogni tipo di programma dato che, come per i file arff, si tratta di informazioni codificate come *plain text*.

Questo fine è stato raggiunto abbastanza agilmente esportando le varie collezioni tramite il comando `mongoexport`, come è possibile vedere in Figura 27, mostrante una piccola porzione delle ricette del `makefile` relative a questo compito.

Una parte significativa di questa operazione è rappresentata dalla richiesta di `mongoexport` di una lista degli attributi da esportare come parametro di input. Per evitare di doverla compilare manualmente, è stata utilizzata una *query* sulla base di dati MongoDB per estrarre automaticamente la lista degli attributi dei documenti di tutte le collezioni da esportare.

Figura 27: porzione del makefile che specifica le ricette per eseguire il *join* fra le valutazioni dei corsi e i dati degli studenti

```

82 EXP= mongoexport
83 FIELDS_FULL= --type=csv --fieldFile=prepr/_exp_fields.txt
84 FIELDS_MIN= --type=csv --fieldFile=prepr/_exp_fields_min.txt
85 FIELDS_STUD= --type=csv --fieldFile=prepr/_exp_fields_stud_gen.txt
86 FIELDS_EVAL= --type=csv --fieldFile=prepr/_exp_fields_eval_gen.txt
87
88 # out of recipes tree, run it manually when needed
89 list_fields:
90     $(PRDIR) sh list_fields.sh
91
92 prep_exp:
93     yes | rm -rf datasets && mkdir datasets
94
95 exp_stud_gen: stud_gen prep_exp
96     $(EXP) --db $(DB) --collection stud_gen $(FIELDS_STUD) > datasets/gen_stud.csv
97
98 exp_eval_gen: teval_gen prep_exp
99     $(EXP) --db $(DB) --collection eval_gen $(FIELDS_EVAL) > datasets/gen_eval.csv
100
101 exp_full: cleaned prep_exp
102     $(EXP) --db $(DB) --collection minable $(FIELDS_FULL) > datasets/full.csv
103
104 exp_min: minified prep_exp
105     $(EXP) --db $(DB) --collection minable_min $(FIELDS_MIN) > datasets/min.csv
106
107 exp_d: discretized prep_exp
108     $(EXP) --db $(DB) --collection minable_discretized $(FIELDS_FULL) > datasets/full_d.csv
109     $(EXP) --db $(DB) --collection minable_min_discretized $(FIELDS_MIN) > datasets/min_d.csv

```

Listing 5.6: script della shell di MongoDB per ottenere la lista degli attributi dei documenti in una collezione

```

1 use exams
2
3 mr = db.runCommand({
4     "mapreduce" : "minable",
5     "map" : function() {
6         for (var key in this) { emit(key, null); }
7     },
8     "reduce" : function(key, stuff) { return null; },
9     "out": "minable" + "_keys"
10 })
11
12 db.minable_keys.distinct("_id")

```

Fatto questo, la fase di *preprocessing* può dirsi conclusa. Si andrà quindi a lavorare sui data set prodotti, impiegando finalmente qualche tecnica di *data mining*.

DATA UNDERSTANDING PER CLASSI

In questo capitolo si andranno a impiegare i due data set minimali, ottenuti mediante una massiccia aggregazione dei dati a disposizione effettuata nella precedente fase di *preprocessing*, come base per l'utilizzo di alcune tecniche di visualizzazione.

Il fine è quello di evidenziare elementi che facciano intuire un *trend* fra le varie **classi di record** in cui si è aggregato tutti i dati a disposizione: si sta parlando, ovviamente, delle *coorti di immatricolazione* per quanto riguarda il data set degli studenti, e degli *Anni Accademici* riguardo alle valutazioni dei corsi.

6.1 VALUTAZIONE DEI CORSI

Per la valutazione dei corsi, si lavorerà sul dataset ottenuto come descritto nella Sezione 5.2.2. Data la sua dimensione molto ristretta, è possibile riportarlo qui sotto per intero:

6.1.1 *Panoramica Generale sulle Valutazioni dei Corsi*

Per inquadrare subito la situazione e mettere a fuoco con un singolo colpo d'occhio le caratteristiche di ogni Anno Accademico, è stato realizzato l'istogramma mostrato in Figura 28.

Come si può facilmente verificare con una rapida osservazione del grafico, l'unico attributo che varia sensibilmente è il numero medio di valutazioni dei corsi registrate; si può notare inoltre che esso presenta una tendenza a crescere con l'avanzare dell'Anno Accademico. Per quanto

A. A.	N.	Valutazione [1-10] <i>media</i>	Std. Dev. Val. <i>media</i>	Val. Sufficienti [%]
2010-2011	17	7.54	1.74	82.14
2011-2012	26	7.93	1.61	90.68
2012-2013	30	7.98	1.7	90.55
2013-2014	47	7.7	1.77	87.17
2014-2015	51	7.94	1.77	89.36
2015-2016	49	8.01	1.76	89.94
2016-2017	53	8.01	1.82	89.34

riguarda gli altri attributi, rimangono sostanzialmente simili, fatta eccezione per la percentuale di valutazioni sufficienti registrata negli Anni Accademici 2010-2011, leggermente inferiori rispetto agli altri.

6.1.2 Dettaglio sulla Percentuale di Valutazioni Sufficienti

Volendo vedere con un maggior livello di dettaglio l'andamento di quest'ultimo aspetto, è stato realizzato un grafico mostrante le percentuali di valutazioni dei corsi sufficienti come serie storica attraverso tutti gli Anni Accademici per i quali si hanno a disposizione delle informazioni. Tale grafico, mostrato in Figura 29, mostra con maggiore chiarezza quanto si è osservato sull'istogramma generale di Figura 28.

6.2 PRODUTTIVITÀ DEGLI STUDENTI

Riguardo ai dati relativi alla produttività degli studenti, il dataset su cui si è lavorato è quello descritto nella Sezione 5.2.1. Esso è già stato riportato nella sua interezza in tale sede ma, data la sua piccola dimensione, è comunque ripetuto qui di seguito per comodità di consultazione.

Figura 28: istogramma mostrante una panoramica generale su tutti gli attributi del dataset descritto in Sezione 5.2.2

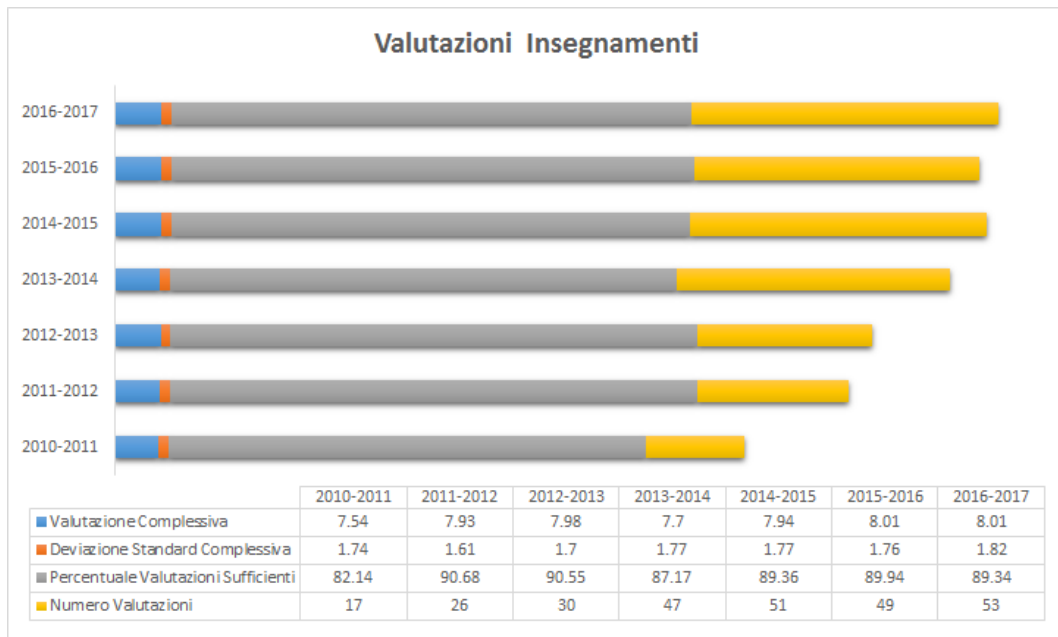
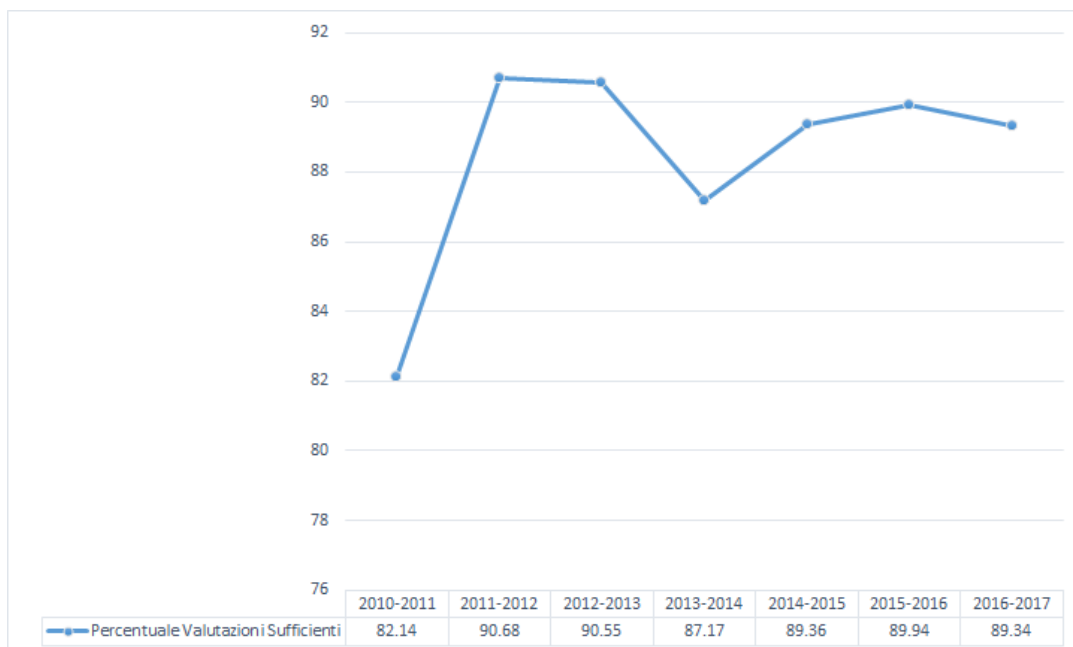


Figura 29: serie storica dell'attributo "percentuale di valutazioni sufficienti" attraverso gli Anni Accademici coperti dal data set delle valutazioni dei corsi



Coorte	N.	Laureati [%]	Test Ingr. [0-25] <i>media</i>	Voto [18-31] <i>media</i>	Ritardo [sem.] <i>media</i>
2010	30	6.67	15.4	25.5	0.81
2011	39	10.26	13.26	24.81	1.07
2012	58	25.86	14.05	24.79	1.01
2013	80	11.25	14.39	24.98	0.77

6.2.1 *Panoramica generale sulla Produttività degli Studenti*

Per quanto riguarda una preliminare analisi generale dell'intero data set, restano valide le considerazioni espresse in Sezione 6.1.1. Pertanto, è stato generato un istogramma del tutto analogo a quanto fatto in tale sezione, mostrato in Figura 30.

In seguito a un'analisi visiva del grafico, si può notare come gli attributi che più variano sono il numero di studenti immatricolati e la percentuale di studenti laureati entro la fine del periodo in esame. L'andamento degli studenti immatricolati è considerevolmente simile a quello riscontrato sul numero di valutazioni dei corsi registrate, quindi è plausibile supporre una certa correlazione diretta fra i due attributi — piuttosto banalmente, se ci sono più studenti iscritti, perverranno anche più valutazioni dei corsi.

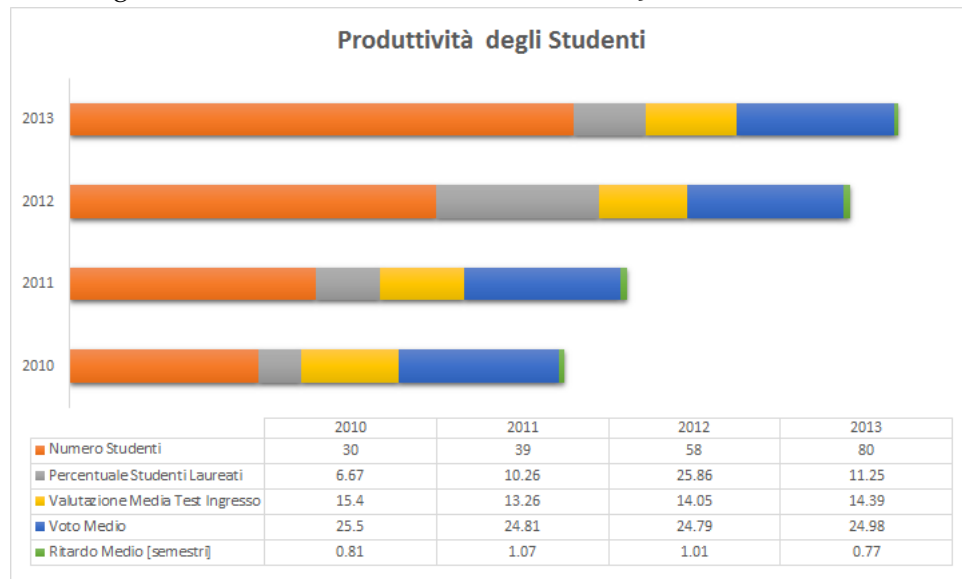
6.2.2 *Relazione fra Test di Ingresso, Voto Medio e Ritardo*

Un aspetto curioso sul quale si è voluto fare luce è l'esistenza o meno di una qualche correlazione fra il risultato conseguito nel test di ingresso e le valutazioni ottenute nei successivi esami di profitto. Avendo a disposizione in questa sede entrambi gli attributi in forma aggregata, è stato realizzato il grafico di Figura 31.

Si noti che gli attributi considerati non esprimono valori nella stessa scala: il test di ingresso prevede un punteggio massimo di 25, mentre per gli esami di profitto il punteggio massimo previsto è di 31 punti¹. C'è stato perciò bisogno di effettuare una *normalizzazione* di tali attributi, per

¹ I voti negli esami vanno ovviamente da 18 a 30, con il 31 a rappresentare in realtà il 30 con lode.

Figura 30: istogramma mostrante una panoramica generale su tutti gli attributi generali del dataset descritto in Sezione 5.2.1



poterli confrontare direttamente.

Come si può vedere, pare esserci una correlazione estremamente blanda, ma non risulta così significativa da essere presa in considerazione per ulteriori approfondimenti. Inoltre, si può notare un *offset* abbastanza pronunciato fra i due attributi; concedendoci una speculazione, potrebbe essere dato da un minor impegno degli studenti nel test di ingresso dato dalla sua soglia di sufficienza molto bassa — 12 punti su 25 — e dal fatto che il risultato in tale test non impatta in alcun modo la futura carriera accademica.

La curiosa flessione della valutazione al test di ingresso fra la coorte di studenti immatricolati nel 2011 trova un riscontro nell'andamento del ritardo medio con cui sono stati superati gli esami di profitto rispetto al loro primo appello, come si può vedere dalla Figura 45. Comunque, si tratta di una correlazione abbastanza debole, pertanto è stato scelto di limitarsi a notarla, senza proseguire nell'analisi.

Figura 31: serie temporale mostrante i valori degli attributi normalizzati "voto medio" e "valutazione media al test di ingresso"

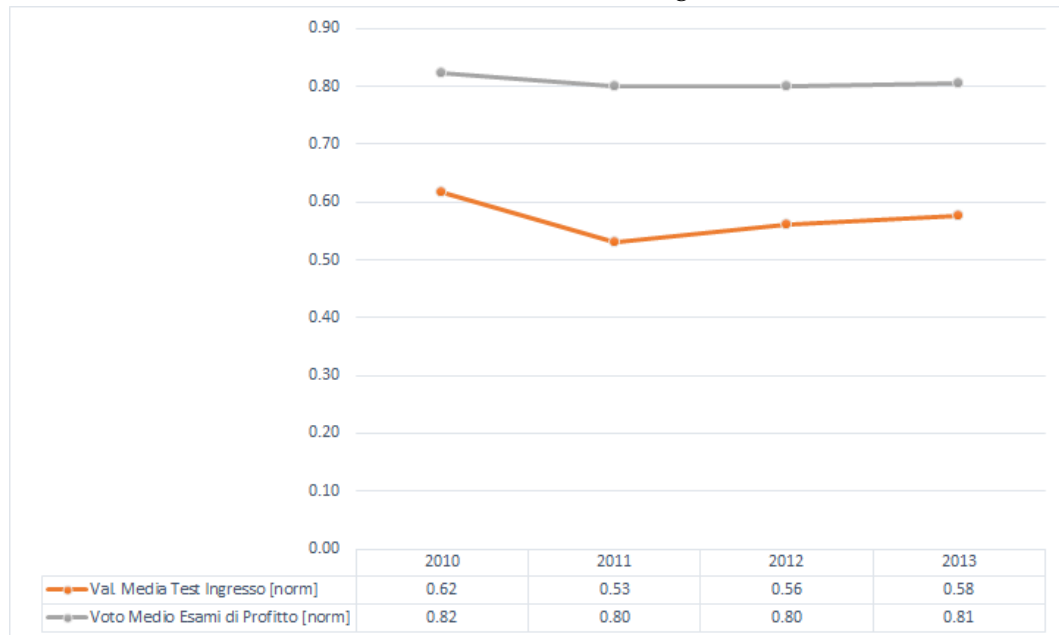


Figura 32: serie temporale mostrante l'andamento dell'attributo "ritardo medio" relativo al superamento degli esami di profitto

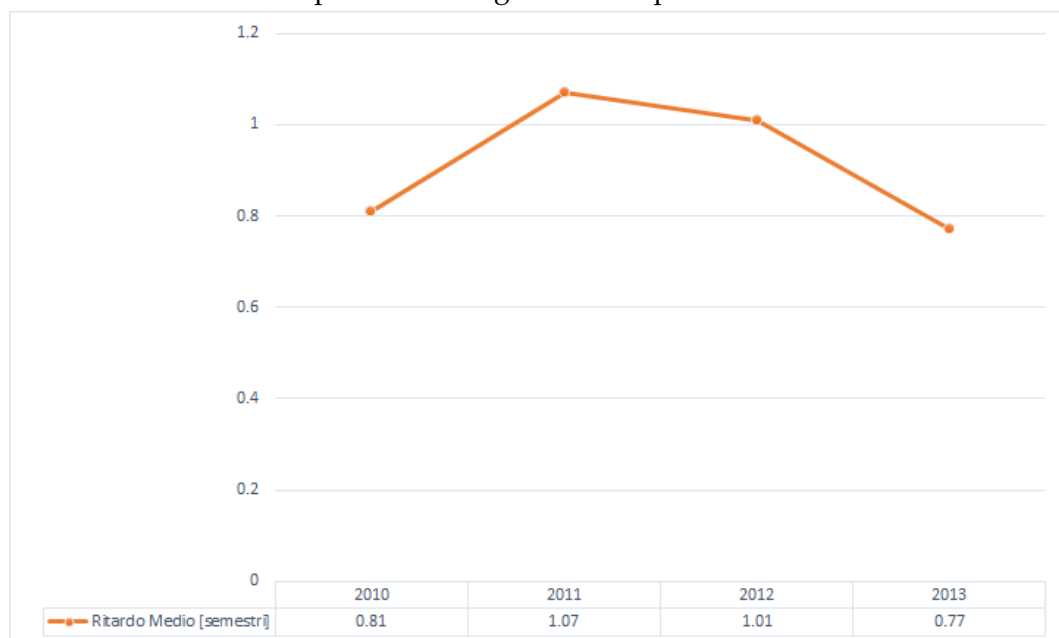
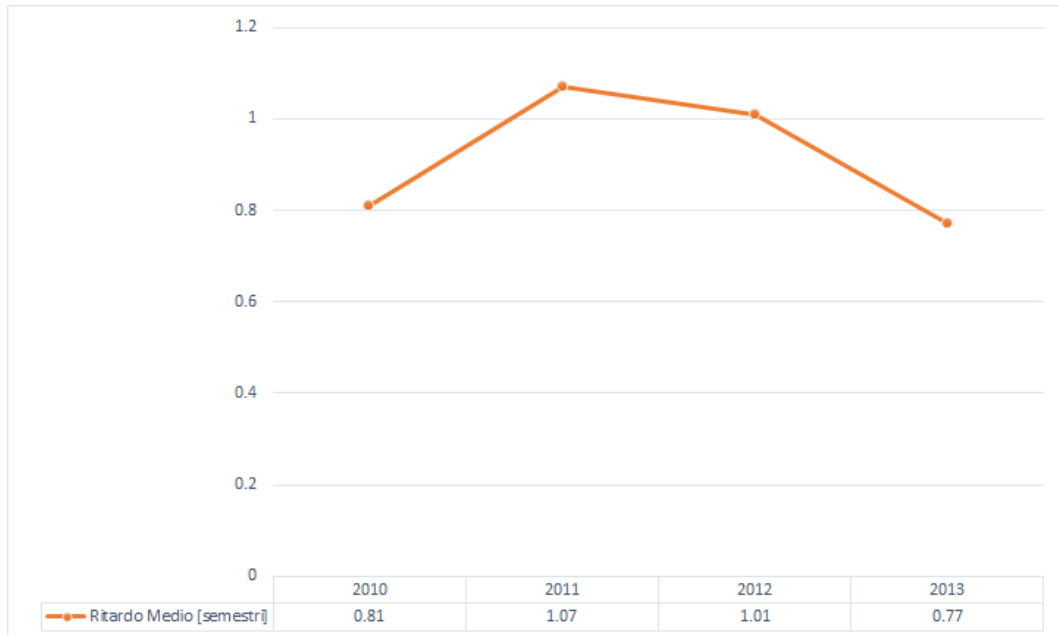


Figura 33: serie temporale mostrante l'andamento dell'attributo "percentuale di studenti laureati entro tre Anni Accademici dall'immatricolazione"



6.2.3 Percentuale di Studenti Laureati

Un indice interessante dell'performance di una certa coorte di immatricolazione è la percentuale di studenti che ha terminato con successo il corso di studi entro il periodo di tre anni considerato dai dati a disposizione. Si è quindi deciso di porre attenzione su questo attributo, mostrandolo come serie temporale in Figura 33.

Si nota una sensibile oscillazione di questo valore, che tocca il suo picco massimo nella coorte di studenti immatricolati nel 2012, ma non si è riscontrato — almeno in seguito a una preliminare analisi visiva — alcuna correlazione con altri aspetti evidenziati da altri attributi.

CLUSTER ANALYSIS

In questo capitolo si descriveranno le attività svolte per realizzare dei *clustering* sulle versioni ad attributi continui dei data set prodotti nella fase di *preprocessing* (descritta nel Capitolo 5).

7.1 INTRODUZIONE ALLA CLUSTER ANALYSIS

Volendo spendere poche parole per delineare un quadro generale e impiegando una massiccia astrazione dai dettagli, possiamo descrivere un *clustering* come segue, citando direttamente da [14]:

"Il **clustering** o **analisi dei gruppi** (dal termine inglese *cluster analysis* introdotto da Robert Tryon nel 1939) è un insieme di tecniche di analisi multivariata dei dati volte alla selezione e raggruppamento di elementi omogenei in un insieme di dati."

Volendo sintetizzare ulteriormente questi concetti, si può dire che un *clustering* su di un certo data set è un raggruppamento di istanze **simili fra loro**. Citando sempre da [14]:

"Le tecniche di clustering si basano su misure relative alla somiglianza tra gli elementi. In molti approcci questa similarità, o meglio, dissimilarità, è concepita in termini di distanza in uno spazio multidimensionale. La bontà delle analisi ottenute dagli algoritmi di clustering dipende molto dalla scelta della metrica, e quindi da come è calcolata la distanza. Gli algoritmi di clustering raggruppano gli elementi sulla base della loro distanza reciproca, e quindi l'appartenenza o meno ad un insieme dipende da quanto l'elemento preso in esame è distante dall'insieme stesso."

Questa breve ma efficace descrizione sommaria fornisce già una sufficiente infarinatura riguardo al tipo di azioni da compiere per ottenere un *clustering*: si tratta infatti di cercare all'interno dei data set in esame dei gruppi di istanze simili fra di loro per qualche criterio — che vedremo essere delle metriche di distanza in uno spazio multidimensionale definito dagli attributi dei dati.

Quanto scritto può essere sufficiente per delineare il quadro generale nel quale si è operato per compiere i passi descritti successivamente. Ovviamente, una trattazione esaustiva sull'argomento necessiterebbe ben altro spazio di quello che si può dedicare in questa tesi di laurea, perciò si rimanda alla consultazione di [13] per ulteriori dettagli, qualora quanto sopra riportato non dovesse essere sufficiente per la comprensione di quanto seguirà.

7.2 ALGORITMI DI CLUSTERING

La realizzazione di un *clustering* su dei *big data* è ovviamente una di quelle attività che hanno bisogno di essere delegate a un algoritmo per essere eseguite. Si descriveranno in questa sezione alcuni dei più efficaci algoritmi di *clustering* e le implementazioni di Weka.

7.2.1 Algoritmo K-Means

Uno dei più noti algoritmi di *clustering* che consente di imporre a priori il numero di *cluster* cercati è senza dubbio **K-Means**. Come si può leggere in [13]:

"The K-means clustering technique is simple [...]. We first choose K initial centroids, where K is a user-specified parameter, namely, the number of clusters desired. Each point is then assigned to the closest centroid, and each collection of points assigned to a centroid is a cluster. The centroid of each cluster is then updated based on the points assigned to the cluster. We repeat the assignment and update steps until no point changes clusters, or equivalently, until the centroids remain the same."

Il che significa, traducendo e parafrasando:

"La tecnica K-Means è semplice: scegliamo inizialmente K centroidi iniziali, dove K è il numero di cluster desiderati. Ogni punto del data set è assegnato al centroide più vicino, e ogni collezione di punti assegnati a un centroide è un cluster. Il centroide di ogni cluster viene poi ricalcolato, basandosi sui punti che gli sono stati assegnati. Si ripetono questi passi fino a che nessun punto cambia cluster, o i centroidi non cambiano."

Quindi, si tratta di una procedura iterativa che, a partire da un numero fissato di punti iniziali, chiamati centroidi, migliora ad ogni passo gli assegnamenti fino a che non si raggiunge una situazione di stabilità.

Le librerie di Weka mettono a disposizione un'implementazione di K-Means, che permette di configurare un gran numero di parametri (come si può vedere in Figura 34).

7.2.2 *Algoritmo DBSCAN*

Un algoritmo sostanzialmente diverso dal precedente è DBSCAN, che non si basa sul minimizzare le distanze dal centroide del *cluster*, ma bensì sul raggruppare le istanze del data set in base alla loro **densità**.

Un'informale ma potente descrizione di questo si può leggere in [13]:

"Density-based clustering locates regions of high density that are separated from one another by regions of low density. DBSCAN is a simple and effective density-based clustering algorithm [...]"

Che tradotto significa:

"Le tecniche di clustering basate sulla densità individuano regioni dense separate le une dalle altre da regioni meno dense. DBSCAN è un algoritmo di clustering basato sulla densità semplice ed efficace."

Quindi, si può dire che DBSCAN sia un algoritmo che, a differenza di K-Means, non cerca di individuare a quale dei cluster decisi inizialmente appartenga una data istanza, ma individua invece i cluster "naturali" del data set basandosi sulla densità di istanze nelle varie regioni dello spazio multidimensionali definito dagli attributi.

Figura 34: finestra che mostra i parametri impostabili dell'algoritmo K-Means

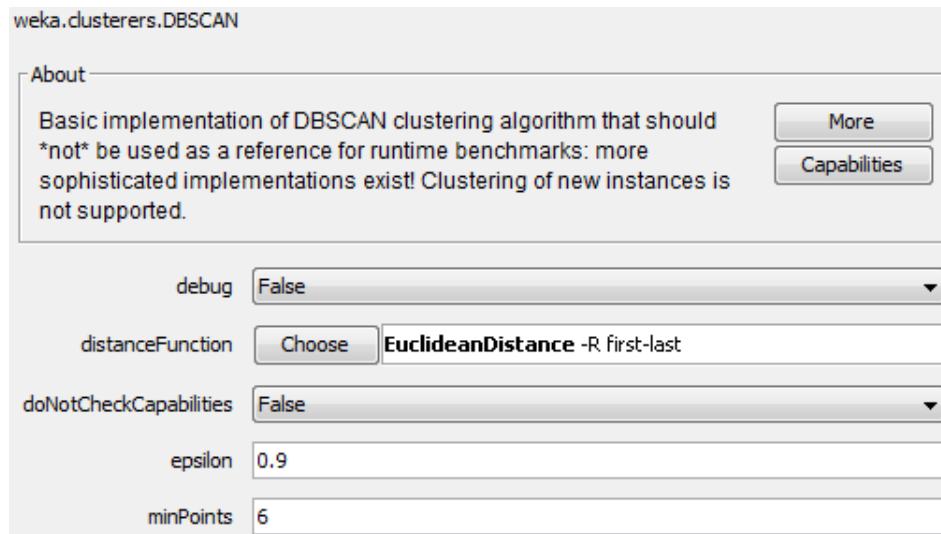
`weka.clusterers.SimpleKMeans`

About

Cluster data using the k means algorithm. [More](#)
[Capabilities](#)

canopyMaxNumCanopiesToHoldInMemory	100
canopyMinimumCanopyDensity	2.0
canopyPeriodicPruningRate	10000
canopyT1	-1.25
canopyT2	-1.0
debug	False
displayStdDevs	False
distanceFunction	Choose EuclideanDistance -R first-last
doNotCheckCapabilities	False
dontReplaceMissingValues	False
fastDistanceCalc	False
initializationMethod	Random
maxIterations	500
numClusters	2
numExecutionSlots	1
preserveInstancesOrder	False
reduceNumberOfDistanceCalcsViaCanopies	False
seed	10

Figura 35: finestra che mostra i parametri impostabili dell'implementazione di DBSCAN su Weka



L'implementazione di DBSCAN fornita da Weka, come si può vedere in Figura 35, è molto basilare, ma totalmente funzionale.

7.3 CLUSTERING SULLE VALUTAZIONI DEI CORSI

Una prima serie di tentativi di *cluster analysis* è stata fatta sul data set condensato delle valutazioni dei corsi, utilizzando l'algoritmo K-Means.

Si è ben coscienti che un'analisi di questo tipo su un data set che contiene solo sette elementi *generalmente* non avrebbe molto senso, ma in questo caso si può considerare come un approfondimento della fase di *data understanding* su questo data set.

7.3.1 Lancio di K-Means

Adottando un metodo empirico di *trial-and-error*, sono state effettuati innumerevoli lanci di K-Means variando di volta in volta i parametri di input.

Quella che segue è la configurazione che genera il miglior risultato ottenuto:

```
weka.clusterers.SimpleKMeans -init 0 -max-candidates 100
-periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -V
-M -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 5000
-num-slots 1 -S 997
```

Nonostante l'elevato numero di parametri a disposizione, in realtà soltanto alcuni hanno influenzato il risultato finale:

- come metrica è stata scelta la **distanza Euclidea**¹;
- sono stati imposti due centroidi iniziali scelti casualmente, ottenendo pertanto un partizionamento con due cluster

Ai fini del clustering, sono stati considerati soltanto tre attributi: la valutazione complessiva del corso, la deviazione standard delle valutazioni e la percentuali di valutazioni sufficienti.

Listing 7.1: Output della console di Weka relativo al lancio di K-Means sul data set delle valutazioni dei corsi

```
1 === Run information ===
2
3 Scheme:   weka.clusterers.SimpleKMeans -init 0 -max-candidates 100
           -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -V
           -M -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 5000
           -num-slots 1 -S 997
4 Relation: gen_eval-weka.filters.unsupervised.attribute.Remove-R5
5 Instances: 7
6 Attributes: 4
7           Valutazione Complessiva [media pesata]
8           Deviazione Standard Complessiva [media pesata]
9           Percentuale Valutazioni Sufficienti [media pesata]
10 Ignored:
11           Anno Accademico
12 Test mode: evaluate on training data
13
14
15 === Clustering model (full training set) ===
16
17 kMeans
```

¹ Si tratta di una metrica che misura la distanza fra due punti di uno spazio multidimensionale come la lunghezza del segmento che li unisce.

```

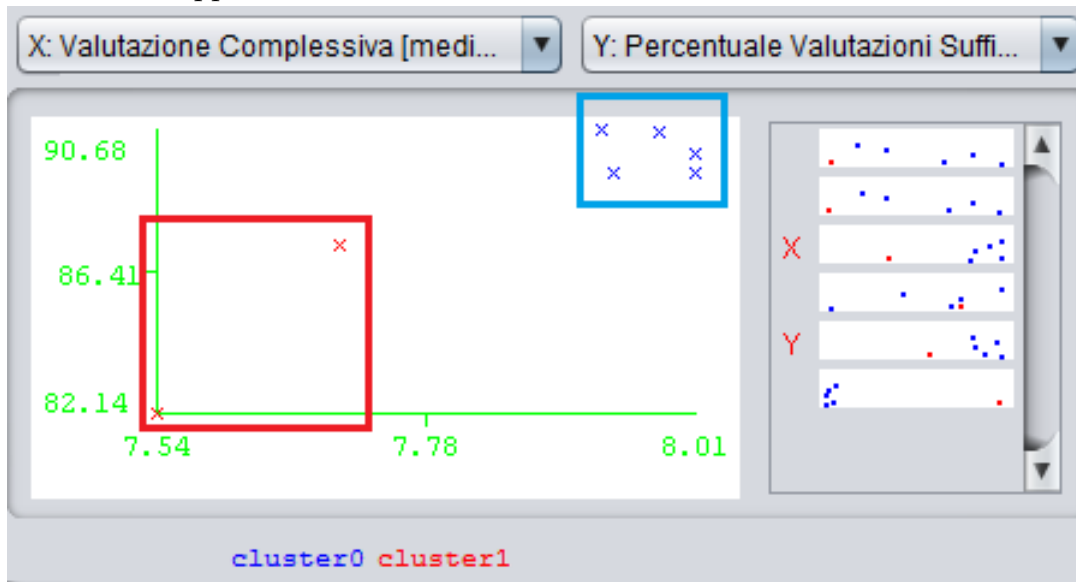
18 =====
19
20 Number of iterations: 2
21 Within cluster sum of squared errors: 0.876427223765708
22
23 Initial starting points (random):
24
25 Cluster 0: 7.93,1.61,90.68
26 Cluster 1: 7.54,1.74,82.14
27
28 Final cluster centroids:
29
30 Attribute                                Cluster#
31                                         Full Data  0          1
32                                         (7.0)      (5.0)      (2.0)
33 =====
34 Val. Compl. [media pesata]              7.8729      7.974      7.62
35                                         +/-0.1814  +/-0.0378  +/-0.1131
36
37 Std. Dev. Compl. [media pesata]         1.7386      1.732      1.755
38                                         +/-0.0672  +/-0.0804  +/-0.0212
39
40 Perc. Val. Suff. [media pesata]        88.4543      89.974      84.655
41                                         +/-3.0188  +/-0.6345  +/-3.5567
42
43 Time taken to build model (full training data) : 0 seconds
44
45 === Model and evaluation on training set ===
46
47 Clustered Instances
48 0      5 ( 71%)
49 1      2 ( 29%)

```

Come si può vedere dalla Figura 36, il data set è stato diviso in due cluster che appaiono visivamente ben separati.

- **Cluster 1** (rosso): A. A. 2013-2014 e 2010-2011
- **Cluster 0** (blu): gli altri cinque A. A.

Figura 36: sezione dello spazio di esistenza del data set delle valutazioni dei corsi lungo il piano definito da "valutazione complessiva" e "percentuale di valutazioni sufficienti", con evidenziato per ogni istanza il cluster di appartenenza



7.4 CLUSTERING SULLA JOIN DEI DUE DATA SET

Quanto segue è la parte fondamentale di questa *cluster analysis*, ovvero lo studio della versione ad attributi continui dell'unione dei due data set iniziali.

7.4.1 Lancio di K-Means

In modo del tutto analogo a quanto fatto per la precedente attività di clustering, sono state fatte molte prove al fine di raggiungere una configurazione di lancio di K-Means soddisfacente:

Listing 7.2: Output della console di Weka relativo al lancio di K-Means sul data set risultante dal join delle due collezioni di dati a disposizione

```
1 === Run information ===
2
3 Scheme:   weka.clusterers.SimpleKMeans -init 0 -max-candidates 100
           -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -V
```



```

43 Time taken to build model (full training data) : 0 seconds
44
45 === Model and evaluation on training set ===
46
47 Clustered Instances
48
49 0      28 ( 52%)
50 1      26 ( 48%)

```

Occorre notare varie cose, al fine di comprendere il significato di quanto è stato ottenuto:

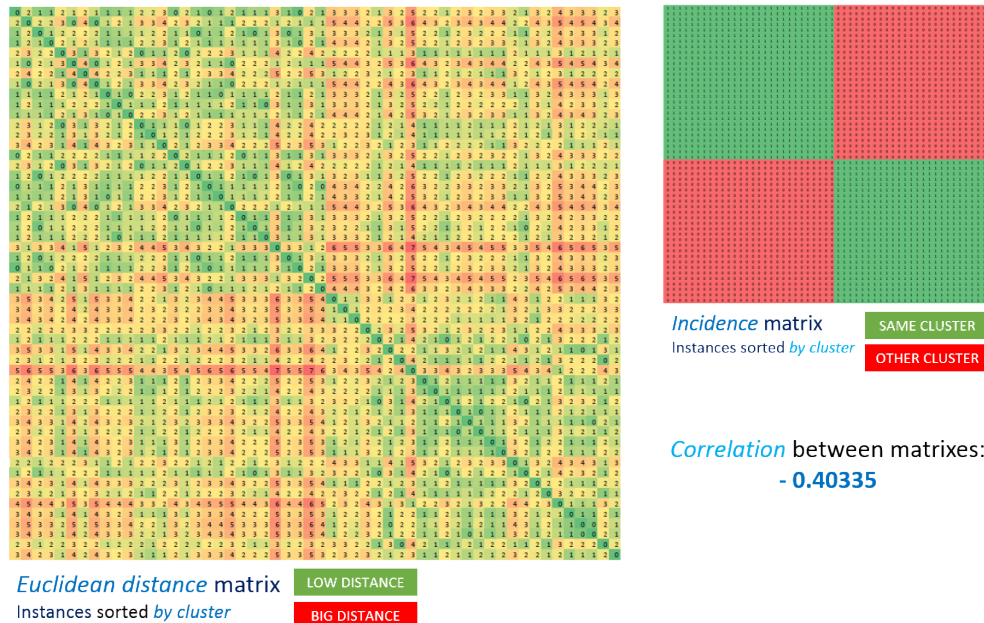
- la **somma dell'errore quadratico**² all'interno dei cluster è molto maggiore di quella ottenuta nell'occasione precedente, ma non si può assolutamente fare un confronto diretto fra le due quantità, sia per la diversa natura dell'indagine che per il diverso numero di istanze tenute in considerazione;
- al fine di semplificare il significato geometrico della metrica di distanza scelta (quella Euclidea), sono stati presi in considerazione ai fini del clustering soltanto gli attributi più caratterizzanti del dataset:
 - Voto medio degli studenti
 - Ritardo medio degli studenti
 - Valutazione media del corso
- sono stati cercati, anche in questo caso, due soli cluster; semanticamente, si può intendere di voler dividere i corsi "migliori", quelli che hanno ricevuto delle buone valutazioni e in cui gli studenti hanno prodotto delle buone performance, da quelli "peggiori".

7.4.2 Analisi dei risultati di K-Means

Da un punto di vista tecnico, la bontà del risultato può essere misurata con la correlazione fra la *matrice di incidenza* e la *matrice di prossimità*: un

² Si tratta di una metrica che misura quanto i punti che compongono il cluster siano distanti dal centroide — in parole estremamente semplici: più l'errore è alto, più il cluster è composto da punti sparsi.

Figura 37: matrice di prossimità, matrice di incidenza e correlazione fra di esse per misurare la conta del clustering effettuato con K-Means



valore negativo indica una tendenza delle distanze fra i punti dello stesso cluster ad essere minore di quelle fra punti di cluster diversi. Come si può vedere in Figura 37, nel caso di questo specifico clustering la correlazione fra le due matrici è circa -0.4 , il che indica i cluster ottenuti sono abbastanza separati.

Il risultato ottenuto è parzialmente in linea con l'intento che ci si era prefissati. Il cluster identificato da Weka come **Cluster 0**, e colorato di **blu** nei grafici che verranno mostrati, può essere considerato quello dei corsi migliori. Conseguentemente, il *cluster 1* può essere letto come quello dei corsi peggiori rispetto agli altri e rispetto alle metriche considerate.

Si vada innanzitutto a vedere come sono state raggruppate le istanze del data set nei due cluster, rispetto ai tre attributi caratterizzati (Figure 38, 39 e 40). Come si può agevolmente notare, la divisione che ci si aspettava di vedere è presente.

Volendo ragionare sulla composizione dei due cluster, si è andati a vedere innanzitutto se l'Anno Accademico di riferimento ha un qualche

Figura 38: sezione dello spazio di esistenza del data set delle valutazioni dei corsi lungo il piano definito da "voto medio" e "ritardo medio", con evidenziato per ogni istanza il cluster di appartenenza

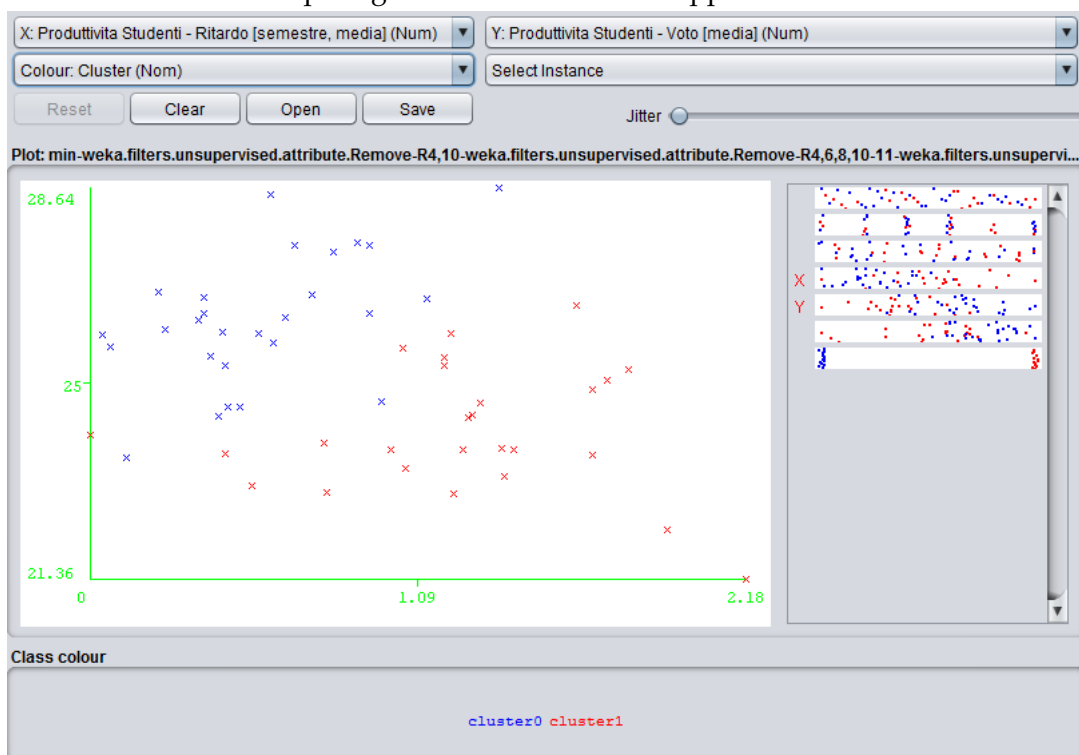


Figura 40: sezione dello spazio di esistenza del data set delle valutazioni dei corsi lungo il piano definito da "voto medio" e "valutazione della didattica", con evidenziato per ogni istanza il cluster di appartenenza

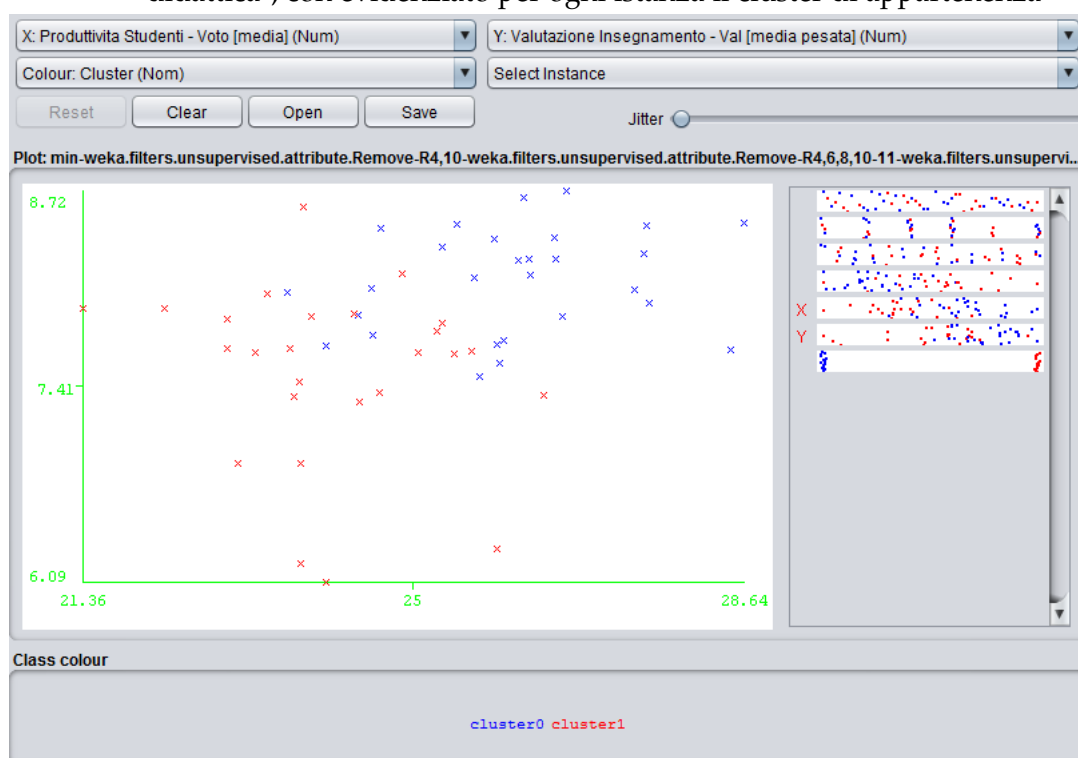
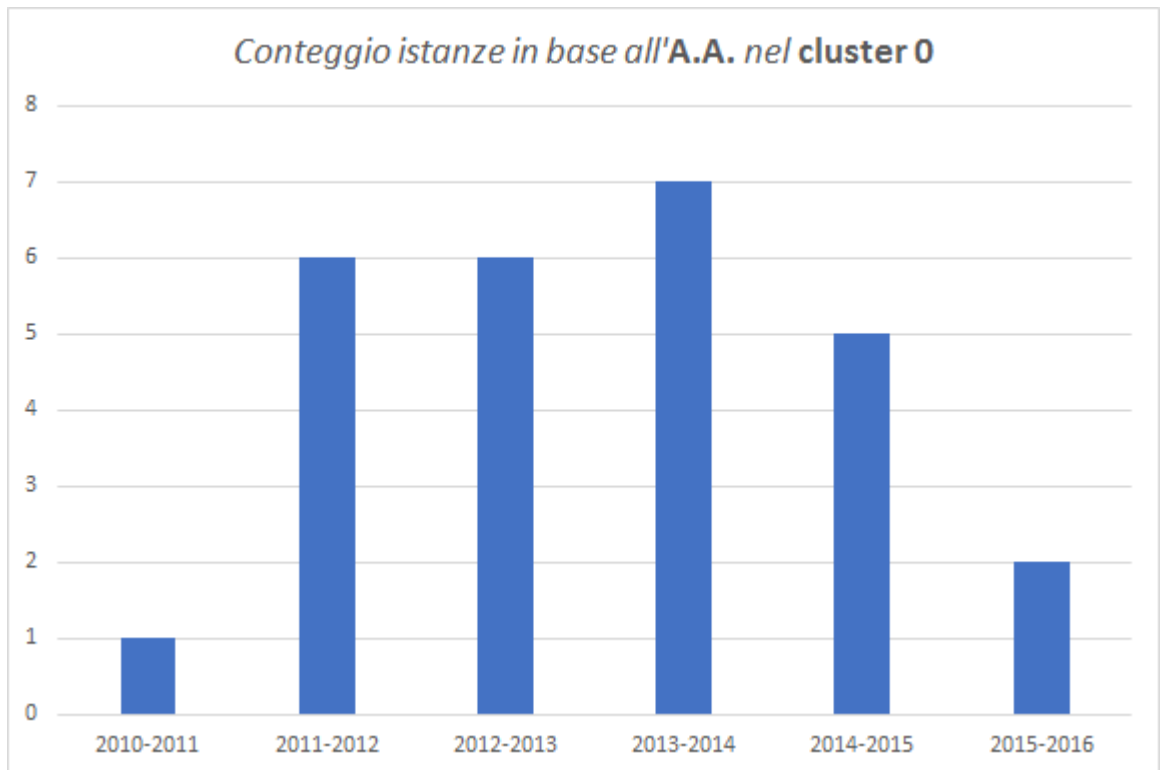


Figura 41: composizione del Cluster 0 riguardo gli Anni Accademici considerati dal data set

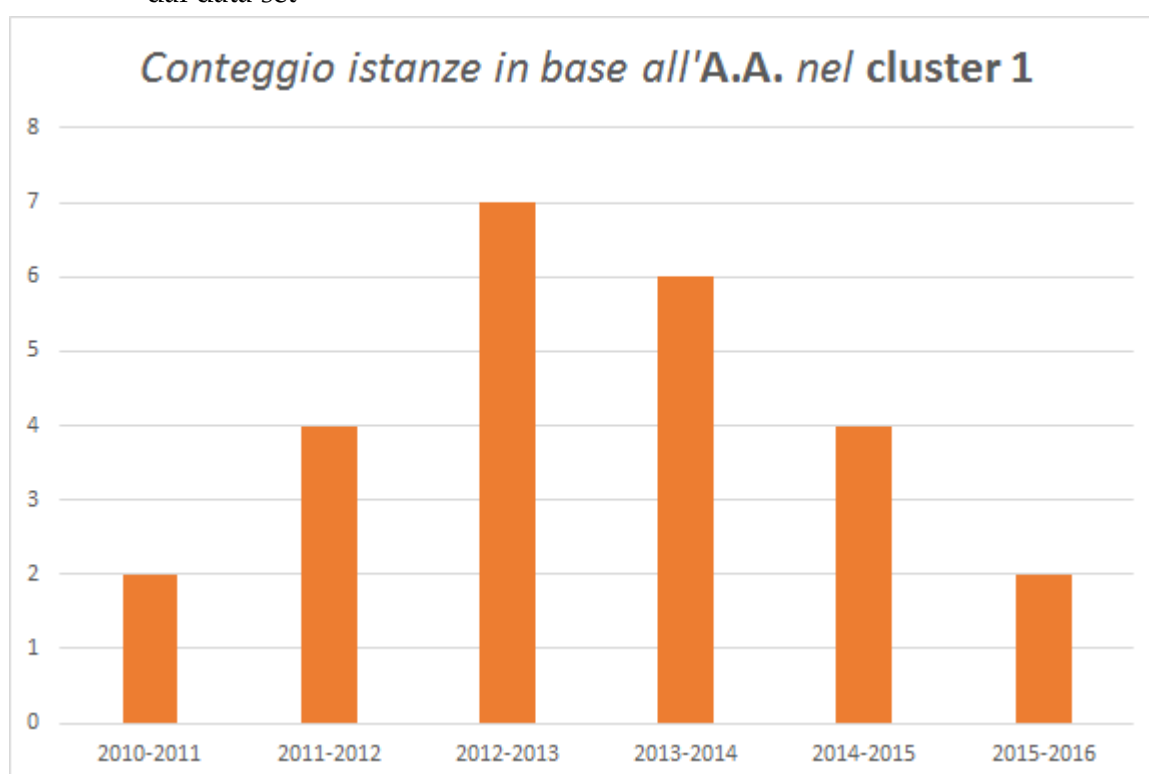


ruolo nella produzione di questo risultato. Come si può osservare nelle Figure 41 e 42, la popolazione di entrambi i cluster è abbastanza simile riguardo a questo aspetto.

Per quanto riguarda invece l'individuare in quale cluster sia finito ogni corso, si è innanzitutto osservata la composizione totale dei due cluster. In caso si fosse trattato di una *cluster analysis* su veri e propri *big data*, questo non sarebbe ovviamente stato possibile, ma dato che il data set in esame ha a malapena sessanta istanze, è stato possibile stilare la seguente lista:

-
- 1 Cluster 0 (blu):
 - 2 2010-2011 'ALGORITMI E STRUTTURE DATI'
 - 3 2010-2011 'PROGRAMMAZIONE'
 - 4 2011-2012 'BASIS DI DATI E SISTEMI INFORMATIVI'

Figura 42: composizione del Cluster 1 riguardo gli Anni Accademici considerati dal data set




```

5 2011-2012 'SISTEMI OPERATIVI'
6 2011-2012 'ALGORITMI E STRUTTURE DATI'
7 2011-2012 'METODOLOGIE DI PROGRAMMAZIONE'
8 2011-2012 'ALGEBRA LINEARE'
9 2011-2012 'ANALISI I: CALCOLO DIFFERENZIALE ED INTEGRALE'
10 2012-2013 'ANALISI I: CALCOLO DIFFERENZIALE ED INTEGRALE'
11 2012-2013 'INFORMATICA TEORICA'
12 2012-2013 'CALCOLO NUMERICO'
13 2012-2013 'PROGRAMMAZIONE'
14 2012-2013 'RETI DI CALCOLATORI'
15 2012-2013 'ALGORITMI E STRUTTURE DATI'
16 2013-2014 'CALCOLO NUMERICO'
17 2013-2014 'ALGORITMI E STRUTTURE DATI'
18 2013-2014 'BASI DI DATI E SISTEMI INFORMATIVI'
19 2013-2014 'PROGRAMMAZIONE'
20 2013-2014 'ANALISI I: CALCOLO DIFFERENZIALE ED INTEGRALE'
21 2013-2014 'METODOLOGIE DI PROGRAMMAZIONE'
22 2013-2014 'RETI DI CALCOLATORI'
23 2014-2015 'INTERPRETI E COMPILATORI'
24 2014-2015 'METODOLOGIE DI PROGRAMMAZIONE'
25 2014-2015 'BASI DI DATI E SISTEMI INFORMATIVI'
26 2014-2015 'RETI DI CALCOLATORI'
27 2014-2015 'CALCOLO NUMERICO'
28 2015-2016 'INTERPRETI E COMPILATORI'
29 2015-2016 'RETI DI CALCOLATORI'
30
31 Cluster 1 (rosso):
32 2010-2011 'ARCHITETTURE DEGLI ELABORATORI'
33 2010-2011 'ANALISI I: CALCOLO DIFFERENZIALE ED INTEGRALE'
34 2010-2011 'MATEMATICA DISCRETA E LOGICA'
35 2011-2012 'PROGRAMMAZIONE CONCORRENTE'
36 2011-2012 'MATEMATICA DISCRETA E LOGICA'
37 2011-2012 'PROGRAMMAZIONE'
38 2011-2012 'ARCHITETTURE DEGLI ELABORATORI'
39 2012-2013 'METODOLOGIE DI PROGRAMMAZIONE'
40 2012-2013 'ALGEBRA LINEARE'
41 2012-2013 'BASI DI DATI E SISTEMI INFORMATIVI'
42 2012-2013 'MATEMATICA DISCRETA E LOGICA'
43 2012-2013 'PROGRAMMAZIONE CONCORRENTE'
44 2012-2013 'ARCHITETTURE DEGLI ELABORATORI'
45 2012-2013 'SISTEMI OPERATIVI'

```

Figura 43: tabella che mostra la composizione dei cluster in riferimento ai corsi di esame

CLUSTER 0	CLUSTER 1
ALGEBRA LINEARE	ALGEBRA LINEARE
ALGORITMI E STRUTTURE DATI	
ANALISI I: CALCOLO DIFFERENZIALE ED INTEGRALE	ANALISI I: CALCOLO DIFFERENZIALE ED INTEGRALE
	ARCHITETTURE DEGLI ELABORATORI
BASI DI DATI E SISTEMI INFORMATIVI	BASI DI DATI E SISTEMI INFORMATIVI
CALCOLO NUMERICO	CALCOLO NUMERICO
INFORMATICA TEORICA	INFORMATICA TEORICA
INTERPRETI E COMPILATORI	
	MATEMATICA DISCRETA E LOGICA
METODOLOGIE DI PROGRAMMAZIONE	METODOLOGIE DI PROGRAMMAZIONE
PROGRAMMAZIONE	PROGRAMMAZIONE
	PROGRAMMAZIONE CONCORRENTE
RETI DI CALCOLATORI	
SISTEMI OPERATIVI	SISTEMI OPERATIVI

- 46 2013-2014 'SISTEMI OPERATIVI'
- 47 2013-2014 'ARCHITETTURE DEGLI ELABORATORI'
- 48 2013-2014 'MATEMATICA DISCRETA E LOGICA'
- 49 2013-2014 'ALGEBRA LINEARE'
- 50 2013-2014 'INFORMATICA TEORICA'
- 51 2013-2014 'PROGRAMMAZIONE CONCORRENTE'
- 52 2014-2015 'INFORMATICA TEORICA'
- 53 2014-2015 'PROGRAMMAZIONE CONCORRENTE'
- 54 2014-2015 'SISTEMI OPERATIVI'
- 55 2014-2015 'ALGEBRA LINEARE'
- 56 2015-2016 'INFORMATICA TEORICA'
- 57 2015-2016 'CALCOLO NUMERICO'

Tuttavia, tale lista non consente di vedere immediatamente alcuna caratteristica fondamentale del clustering in esame. La maggior parte dei corsi, rappresentati da varie istanze secondo l'Anno Accademico di riferimento, è presente sia nel Cluster 0 che nel Cluster 1. Pertanto, è stata realizzata la tabella mostrata in Figura 43 per evidenziare quali corsi di esame sono del tutto assenti da quali cluster.

Da tale Figura, si può notare che il Cluster 0, quello dei corsi "buoni",

comprende tutte le istanze relative ai corsi di:

- Algoritmi e Strutture dati
- Interpreti e Compilatori
- Reti di Calcolatori

Inoltre, si può vedere che nel Cluster 1, quello dei corsi "meno buoni", sono presenti tutte le istanze dei seguenti corsi:

- Architetture degli Elaboratori
- Matematica Discreta e Logica
- Programmazione Concorrente

Questa caratteristica richiama fortemente quanto visto nel Capitolo 4, ovvero durante la fase di *data understanding*, rafforzando quindi l'ipotesi della bontà dell'analisi che si sta effettuando.

Volendo approfondire ulteriormente l'analisi della composizione dei cluster, nelle Figure 44, 45 e 46 si possono vedere degli istogrammi che mostrano la somma dei tre attributi considerati per realizzare il clustering. Ovviamente, i valori assoluti di tali somme non significano niente, ma è interessante vedere come gli attributi "positivi" — la valutazione della didattica e il voto medio — hanno una somma più elevata nel Cluster 0, mentre quello "negativo" — il ritardo nel dare gli esami — è molto più accentuato nel Cluster 1.

7.4.3 Tentativo di utilizzo di DBSCAN

Come il titolo di questa sezione suggerisce, non è stato possibile utilizzare con profitto l'algoritmo DBSCAN.

Imitando quanto fatto con K-Means, ovvero limitando il numero di attributi da considerare per il clustering, ci si aspetterebbe che un algoritmo basato sulla densità possa esibire delle buone performance. Per facilitare ulteriormente l'esecuzione di DBSCAN, è stato addirittura scartato l'attributo relativo al ritardo degli studenti. Così facendo, si è ottenuto uno spazio bidimensionale in cui l'algoritmo può facilmente operare - e che

Figura 44: istogramma che mostra la somma dell'attributo "valutazione della didattica" nei due cluster ottenuti, divisa per Anno Accademico

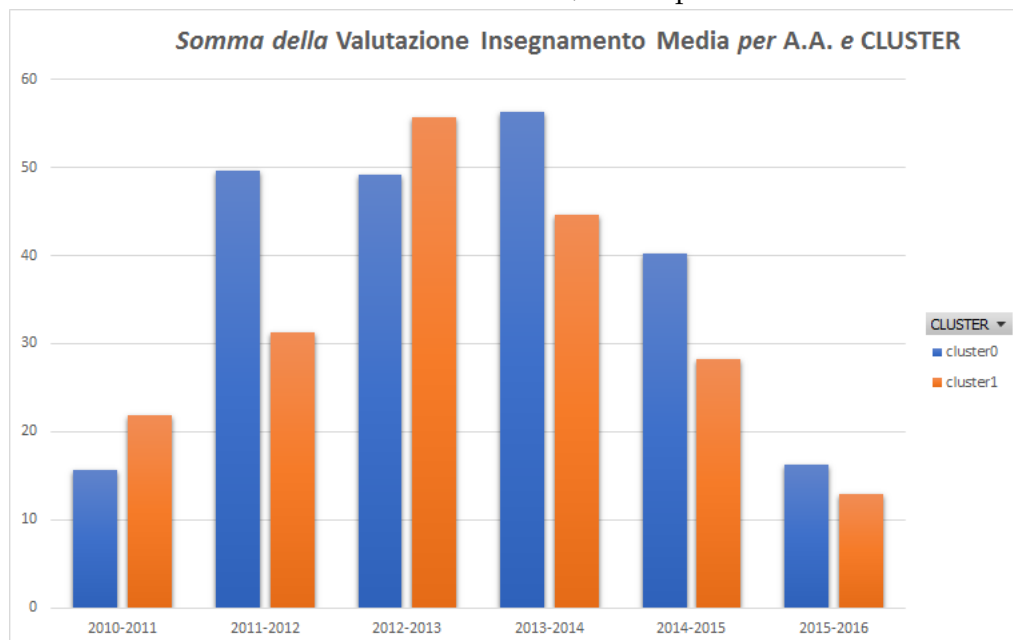


Figura 45: istogramma che mostra la somma dell'attributo "ritardo medio nel dare gli esami" nei due cluster ottenuti, divisa per Anno Accademico

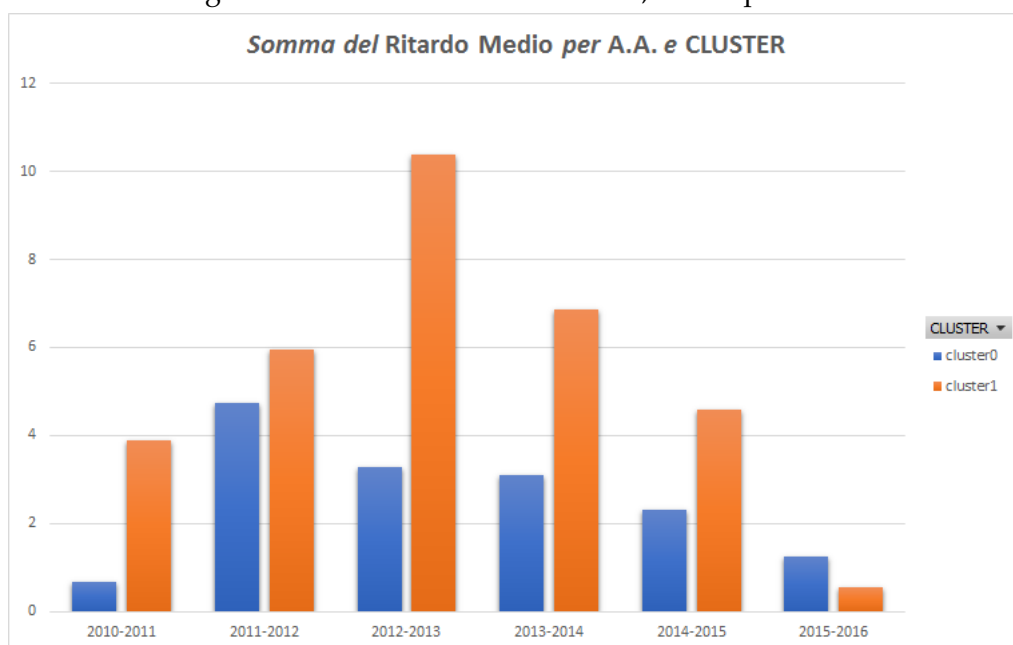
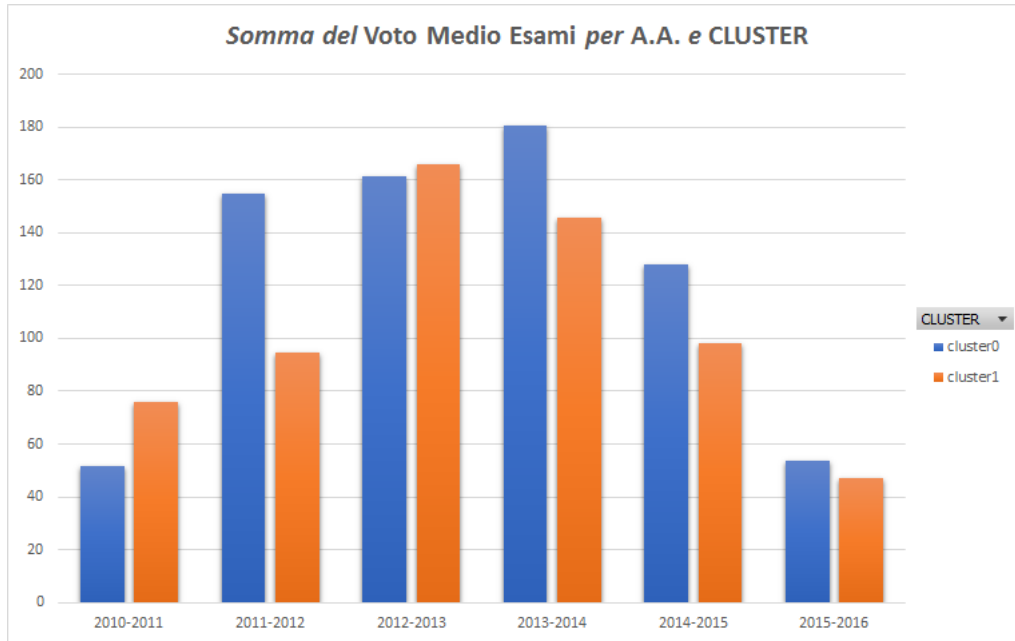


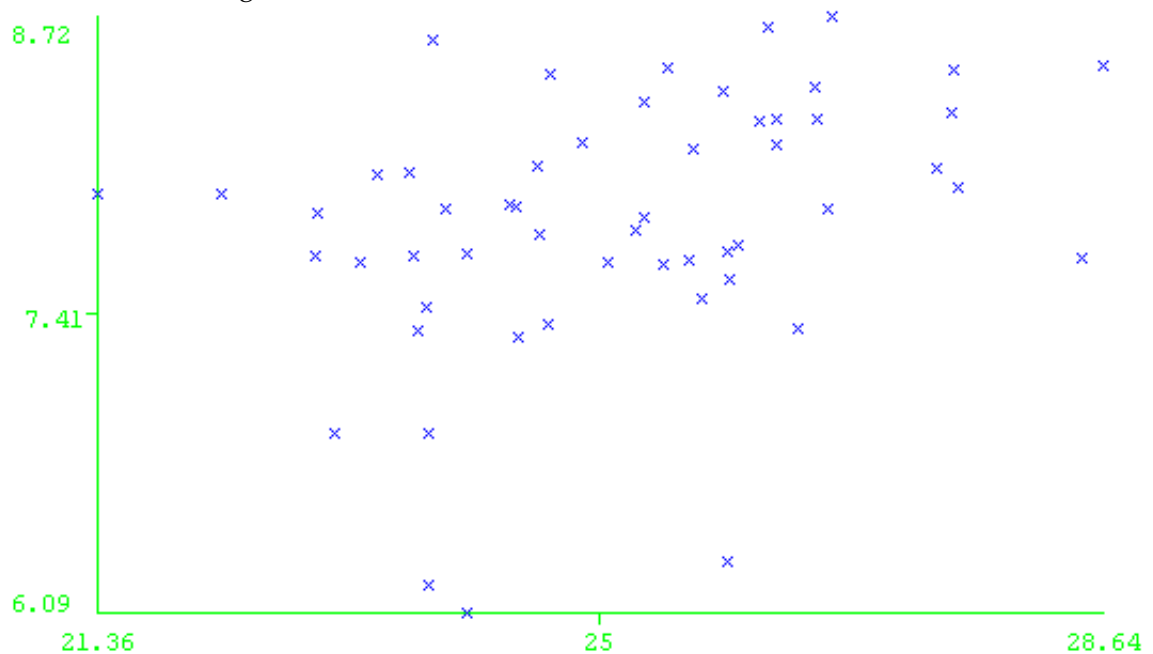
Figura 46: istogramma che mostra la somma dell'attributo "voto medio conseguito negli esami di profitto" nei due cluster ottenuti, divisa per Anno Accademico



si presta bene a un'interpretazione visiva.

Invece, non è stato possibile ottenere nessun risultato diverso da un cluster unico, per qualunque valore di epsilon e per qualsiasi numero di punti minimi richiesti. Questo perché evidentemente i punti nel nostro piano di analisi sono addensati tutti in un'unica regione, fatto che possiamo agevolmente verificare osservandone un grafico in Figura 47.

Figura 47: piano composto dalle dimensioni "Voto Medio negli Esami" e "Valutazione Media dell'Insegnamento", sul quale è stato tentato il clustering con l'algoritmo DBSCAN



ANALISI ASSOCIATIVA

Di seguito verrà descritto il lavoro svolto nell'ambito della ricerca di regole associative.

Come è già stato accennato in precedenza, questo tipo di analisi ha richiesto l'impiego delle versioni discretizzate dei data set provenienti dalla fase di *preprocessing*. Il motivo di questo requisito apparirà chiaro andando a descrivere brevemente il funzionamento delle tecniche di analisi associativa.

8.1 INTRODUZIONE ALL'ANALISI ASSOCIATIVA

Questa introduzione vuole essere solo un breve riassunto dei concetti fondamentali seguiti per realizzare all'atto pratico l'analisi. Una trattazione più approfondita — che trascenderebbe ampiamente lo scopo di questa tesi di laurea — può essere trovata in [13], ed è proprio quella che è stata consultata per realizzare quanto segue.

8.1.1 Regole Associative

Una regola associativa è un'implicazione del tipo $A \rightarrow B$, con A e B insiemi di item (detti, appunto *itemset*). Il significato di ciò dovrebbe essere palese: data la presenza di A in una istanza del data set, è *fortemente probabile* la presenza di B . La valutazione di questa probabilità avviene considerando alcune metriche particolari quali, ad esempio, la *confidenza* o il *lift*.

Portando un esempio sul data set oggetto di questa analisi, una tanto probabile quanto banale regola associativa che ci si aspetta di trovare

potrà essere del tipo “*Valutazione del corso: ALTA*” → “*Voto conseguito: ALTO*”.

Quello a cui si mira, però, è riuscire a scoprire qualche altra regola che trascenda il limite dell’ovvio, aprendo così le porte a interpretazioni non immediate dell’insieme di dati su cui si sta lavorando. Per questo motivo, oltre all’aiuto di criteri algoritmici di potatura, occorrerà comunque prevedere un *intervento umano* nel *post processing* delle regole generate.

8.1.2 L’algoritmo Apriori di Weka

Il processo di generazione delle regole associative utilizzato in questa analisi si basa sul **principio Apriori**. Dato che una regola non è altro che un’implicazione generata da un partizionamento binario di un itemset frequente, il pilastro fondamentale da cui partire per generare le regole è il *mining* degli itemset frequenti.

In estrema sintesi, l’idea generale è quella di generare una lista di *candidati* semplici — cioè di itemset che potrebbero essere frequenti — verificarne il *supporto* e usare quelli idonei per generare altri candidati più complessi. Il principio Apriori, infatti, stabilisce che se un itemset con pochi elementi è infrequente, lo saranno anche tutti gli itemset che lo comprendono. Questo fatto, detto *anti-motonia* del supporto, ci consente di evitare di generare e calcolare la frequenza di ogni possibile itemset, rendendo trattabile quello che sarebbe invece un problema NP – Completo.

Nel dettaglio dell’analisi da effettuare, l’algoritmo Apriori messo a disposizione dal software Weka accetta in input diversi parametri, che consentono di specificarne il comportamento dell’algoritmo, adattandolo agli scopi che ci si è prefissi. Si veda adesso una configurazione di esempio:

```
Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1
```

I vari *flags* da passare come argomenti vanno a regolare questi parametri della computazione:

- -N: numero di regole associative da trovare;

Figura 48: pannello di scelta delle impostazioni per l'algoritmo Apriori di Weka

Choose **Apriori** -N 10 -T 3 -C 1.3 -D 0.05 -U 1.3 -M 0.2 -S -1.0 -A -c -1

weka.gui.GenericObjectEditor

weka.associations.Apriori

About

Class implementing an Apriori-type algorithm. More Capabilities

car

classIndex

delta

doNotCheckCapabilities

lowerBoundMinSupport

metricType

minMetric

numRules

outputItemSets

removeAllMissingCols

significanceLevel

treatZeroAsMissing

upperBoundMinSupport

verbose

Open... Save... OK Cancel

- -T: tipo di metrica utilizzata per la valutazione degli itemset
 - 0 - *confidence*
 - 1 - *lift*
 - 2 - *leverage*
 - 3 - *conviction*;
- -C: valore minimo della metrica indicata per considerare frequente un itemset;
- -D: valore che viene usato per diminuire il supporto ad ogni iterazione dell'algoritmo;
- -U: limite superiore del supporto minimo richiesto;
- -M: limite inferiore del supporto minimo richiesto.

Ovviamente, come si può vedere in Figura 48, sono possibili molte altre personalizzazioni e impostazioni, fatto che rende l'implementazione di Weka dell'algoritmo Apriori estremamente flessibile ed adattabile a molte esigenze.

8.2 APRIORI SU DATASET AGGREGATO

Decidere a priori i parametri ottimali per l'algoritmo Apriori¹ è da considerarsi impossibile. Pertanto, anche in questo caso l'iter più adatto per ottenere dei risultati che possono essere giudicati interessanti è il banale ma efficace *trial-and-error*.

La conseguenza immediata di questo è che, similamente a quanto fatto per la descrizione di altri tipi di analisi sui dati, in questa sezione verranno riportati solo gli output giudicati in qualche senso significativi.

In ogni caso, alcune scelte sono state comuni a tutte le analisi:

- come *metrica* è stato scelto il **Lift**, un cui valore positivo indica una correlazione effettiva — cosa che non è affatto garantita da un alto valore di *confidenza*;

¹ gioco di parole **non** intenzionale.

- non è stata usata la valutazione per classe, né il pruning per livello di significato;
- sono stati sempre stampati gli itemset frequenti, in quanto utili per capire l'andamento della generazione delle regole;
- come *delta* è stato scelto un valore il più piccolo possibile, ma compatibile con dei tempi macchina "umani".

8.2.1 Focus sul corso

L'analisi che viene presentata nell'immediato seguito è stata ottenuta cercando correlazioni fra gli attributi relativi all'andamento generale dei corsi d'esame. In particolare, oltre che verificare, come ipotizzato, la presenza di regole del tipo "*Valutazione del corso: ALTA*" \rightarrow "*Voto conseguito: ALTO*", andare a vedere se esistono legami fra questi due aspetti e il ritardo con cui uno studente supera un esame.

L'algoritmo è stato lanciato sul data set `min_d.csv`, ulteriormente semplificato grazie alla considerazione di questi soli attributi:

- identificativo dell'insegnamento;
- ritardo medio del superamento dell'esame (rispetto al primo appello disponibile, contato in semestri);
- voto medio ottenuto all'esame dagli studenti;
- valutazione media del corso d'insegnamento.

Il miglior risultato che si è ottenuto in quest'ottica è il seguente:

```

1 === Run information ===
2
3 Scheme:   weka.associations.Apriori -I -N 10 -T 1 -C 1.15 -D 0.001
          -U 1.0 -M 0.1 -S -1.0 -c 1
4 Relation: [...]
5 Instances: 54
6 Attributes: 4
7             Insegnamento
8             Produttivita Studenti - Ritardo [semestre, media]
```

```

9          Produttivita Studenti - Voto [media]
10         Valutazione Insegnamento - Val [media pesata]
11
12 === Associator model (full training set) ===
13
14 Apriori
15 =====
16
17 Minimum support: 0.19 (10 instances)
18 Minimum metric <lift>: 1.15
19 Number of cycles performed: 806
20
21 Generated sets of large itemsets:
22
23 Size of set of large itemsets L(1): 7
24
25 Large Itemsets L(1):
26 Produttivita Studenti - Ritardo [semestre, media]=0-0.5 17
27 Produttivita Studenti - Ritardo [semestre, media]=0.5-1 14
28 Produttivita Studenti - Ritardo [semestre, media]=1-1.5 16
29 Produttivita Studenti - Voto [media]=25-28 27
30 Produttivita Studenti - Voto [media]=22-25 24
31 Valutazione Insegnamento - Val [media pesata]=8-10 21
32 Valutazione Insegnamento - Val [media pesata]=6-8 33
33
34 Size of set of large itemsets L(2): 7
35
36 Large Itemsets L(2):
37 Produttivita Studenti - Ritardo [semestre, media]=0-0.5
38 Produttivita Studenti - Voto [media]=25-28 10
39
40 Produttivita Studenti - Ritardo [semestre, media]=0-0.5
41 Valutazione Insegnamento - Val [media pesata]=8-10 10
42
43 Produttivita Studenti - Ritardo [semestre, media]=1-1.5
44 Produttivita Studenti - Voto [media]=22-25 10
45
46 Produttivita Studenti - Ritardo [semestre, media]=1-1.5
47 Valutazione Insegnamento - Val [media pesata]=6-8 12
48
49 Produttivita Studenti - Voto [media]=25-28

```

```

50 Valutazione Insegnamento - Val [media pesata]=8-10 14
51
52 Produttivita Studenti - Voto [media]=25-28
53 Valutazione Insegnamento - Val [media pesata]=6-8 13
54
55 Produttivita Studenti - Voto [media]=22-25
56 Valutazione Insegnamento - Val [media pesata]=6-8 18
57
58 Best rules found:
59
60 1. Produttivita Studenti - Ritardo [semestre, media]=0-0.5 17 ==>
    Valutazione Insegnamento - Val [media pesata]=8-10 10
    conf:(0.59) < lift:(1.51)> lev:(0.06) [3] conv:(1.3)
61 2. Valutazione Insegnamento - Val [media pesata]=8-10 21 ==>
    Produttivita Studenti - Ritardo [semestre, media]=0-0.5 10
    conf:(0.48) < lift:(1.51)> lev:(0.06) [3] conv:(1.2)
62 3. Produttivita Studenti - Ritardo [semestre, media]=1-1.5 16 ==>
    Produttivita Studenti - Voto [media]=22-25 10 conf:(0.63) <
    lift:(1.41)> lev:(0.05) [2] conv:(1.27)
63 4. Produttivita Studenti - Voto [media]=22-25 24 ==> Produttivita
    Studenti - Ritardo [semestre, media]=1-1.5 10 conf:(0.42) <
    lift:(1.41)> lev:(0.05) [2] conv:(1.13)
64 5. Produttivita Studenti - Voto [media]=25-28 27 ==> Valutazione
    Insegnamento - Val [media pesata]=8-10 14 conf:(0.52) <
    lift:(1.33)> lev:(0.06) [3] conv:(1.18)
65 6. Valutazione Insegnamento - Val [media pesata]=8-10 21 ==>
    Produttivita Studenti - Voto [media]=25-28 14 conf:(0.67) <
    lift:(1.33)> lev:(0.06) [3] conv:(1.31)
66 7. Produttivita Studenti - Voto [media]=22-25 24 ==> Valutazione
    Insegnamento - Val [media pesata]=6-8 18 conf:(0.75) <
    lift:(1.23)> lev:(0.06) [3] conv:(1.33)
67 8. Valutazione Insegnamento - Val [media pesata]=6-8 33 ==>
    Produttivita Studenti - Voto [media]=22-25 18 conf:(0.55) <
    lift:(1.23)> lev:(0.06) [3] conv:(1.15)
68 9. Produttivita Studenti - Ritardo [semestre, media]=1-1.5 16 ==>
    Valutazione Insegnamento - Val [media pesata]=6-8 12
    conf:(0.75) < lift:(1.23)> lev:(0.04) [2] conv:(1.24)
69 10. Valutazione Insegnamento - Val [media pesata]=6-8 33 ==>
    Produttivita Studenti - Ritardo [semestre, media]=1-1.5 12
    conf:(0.36) < lift:(1.23)> lev:(0.04) [2] conv:(1.06)

```

Semplificando la *naming convention* a favore della leggibilità, le regole trovate sono queste:

- **R1:** Ritardo: BASSO \rightarrow Valutazione insegnamento: OTTIMA
- **R2:** Valutazione insegnamento: OTTIMA \rightarrow Ritardo: BASSO
- **R3:** Ritardo: ALTO \rightarrow Voto: MEDIO – BASSO
- **R4:** Voto: MEDIO – BASSO \rightarrow Ritardo: ALTO
- **R5:** Voto: MEDIO – ALTO \rightarrow Valutazione insegnamento: OTTIMA
- **R6:** Valutazione insegnamento: OTTIMA \rightarrow Voto: MEDIO – ALTO
- **R7:** Voto: MEDIO – BASSO \rightarrow Valutazione insegnamenti: SUFF.
- **R8:** Valutazione insegnamenti: SUFF. \rightarrow Voto: MEDIO – BASSO
- **R9:** Ritardo: ALTO \rightarrow Valutazione insegnamenti: SUFF.
- **R10:** Valutazione insegnamenti: SUFF. \rightarrow Ritardo: ALTO

Andando ad analizzare le regole trovate, eliminando ridondanze ed effettuando una sorta di *post processing* manualmente, le informazioni chiave che si sono ottenute risultano essere le seguenti:

- Ritardo: BASSO \leftrightarrow Valutazione insegnamento: OTTIMA
- Ritardo: ALTO \leftrightarrow Voto: MEDIO – BASSO
- Ritardo: ALTO \leftrightarrow Valutazione insegnamenti: SUFF.
- Voto: MEDIO – ALTO \leftrightarrow Valutazione insegnamento: OTTIMA
- Voto: MEDIO – BASSO \leftrightarrow Valutazione insegnamenti: SUFF.

Quelle trovate sono tutte implicazioni doppie, il che significa che fra gli attributi esiste una forte correlazione. Messa da parte l'attesa e apparente proporzionalità diretta fra la valutazione del corso e il voto ottenuto all'esame, si può notare l'esistenza di un'analogia correlazione fra il ritardo con cui gli studenti hanno superato l'esame e la valutazione del corso.

8.2.2 Focus sul docente

Facendo uso di alcuni, specifici attributi del dataset `full_d.csv`, si è tentato di scoprire correlazioni fra la produttività degli studenti e l'aspetto della valutazione del corso relativo al docente. Nel particolare, sono stati considerati questi attributi:

- Identificativo del docente
- Ritardo medio del superamento dell'esame
- Voto medio ottenuto all'esame dagli studenti
- Valutazione media del paragrafo relativo al docente

Il miglior risultato che si è ottenuto, a seguito di iterazioni volte a proseguire nella direzione sopra descritta, è il seguente:

```

1 === Run information ===
2
3 Scheme:   weka.associations.Apriori -I -N 10 -T 1 -C 1.15 -D 0.001
         -U 1.0 -M 0.1 -S -1.0 -c 1
4 Relation: [...]
5 Instances: 57
6 Attributes: 4
7           Hash Docente/i
8           Produttivita Studenti - Ritardo [semestre, media]
9           Produttivita Studenti - Voto [media]
10          Valutazione Insegnamento - Docente - Val [media pesata]
11 === Associator model (full training set) ===
12
13
14 Apriori
15 =====
16
17 Minimum support: 0.18 (10 instances)
18 Minimum metric <lift>: 1.15
19 Number of cycles performed: 816
20
21 Generated sets of large itemsets:
22

```

```

23 Size of set of large itemsets L(1): 7
24
25 Large Itemsets L(1):
26 Produttivita Studenti - Ritardo [semestre, media]=0-0.5 18
27 Produttivita Studenti - Ritardo [semestre, media]=0.5-1 14
28 Produttivita Studenti - Ritardo [semestre, media]=1-1.5 16
29 Produttivita Studenti - Voto [media]=25-28 27
30 Produttivita Studenti - Voto [media]=22-25 26
31 Valutazione Insegnamento - Docente - Val [media pesata]=8-10 30
32 Valutazione Insegnamento - Docente - Val [media pesata]=6-8 23
33
34 Size of set of large itemsets L(2): 6
35
36 Large Itemsets L(2):
37 Produttivita Studenti - Ritardo [semestre, media]=0-0.5
38 Produttivita Studenti - Voto [media]=25-28 10
39
40 Produttivita Studenti - Ritardo [semestre, media]=0-0.5
41 Valutazione Insegnamento - Docente - Val [media pesata]=8-10 12
42
43 Produttivita Studenti - Ritardo [semestre, media]=1-1.5
44 Produttivita Studenti - Voto [media]=22-25 10
45
46 Produttivita Studenti - Voto [media]=25-28
47 Valutazione Insegnamento - Docente - Val [media pesata]=8-10 18
48
49 Produttivita Studenti - Voto [media]=22-25
50 Valutazione Insegnamento - Docente - Val [media pesata]=8-10 10
51
52 Produttivita Studenti - Voto [media]=22-25
53 Valutazione Insegnamento - Docente - Val [media pesata]=6-8 14
54
55
56 Best rules found:
57
58 1. Produttivita Studenti - Ritardo [semestre, media]=1-1.5 16 ==>
    Produttivita Studenti - Voto [media]=22-25 10 conf:(0.63) <
    lift:(1.37)> lev:(0.05) [2] conv:(1.24)
59 2. Produttivita Studenti - Voto [media]=22-25 26 ==> Produttivita
    Studenti - Ritardo [semestre, media]=1-1.5 10 conf:(0.38) <
    lift:(1.37)> lev:(0.05) [2] conv:(1.1)

```


- 60 3. Produttivita Studenti - Voto [media]=22-25 26 ==> Valutazione
Insegnamento - Docente - Val [media pesata]=6-8 14 conf:(0.54)
< lift:(1.33)> lev:(0.06) [3] conv:(1.19)
- 61 4. Valutazione Insegnamento - Docente - Val [media pesata]=6-8 23
==> Produttivita Studenti - Voto [media]=22-25 14 conf:(0.61) <
lift:(1.33)> lev:(0.06) [3] conv:(1.25)
- 62 5. Produttivita Studenti - Voto [media]=25-28 27 ==> Valutazione
Insegnamento - Docente - Val [media pesata]=8-10 18 conf:(0.67)
< lift:(1.27)> lev:(0.07) [3] conv:(1.28)
- 63 6. Valutazione Insegnamento - Docente - Val [media pesata]=8-10 30
==> Produttivita Studenti - Voto [media]=25-28 18 conf:(0.6) <
lift:(1.27)> lev:(0.07) [3] conv:(1.21)
- 64 7. Produttivita Studenti - Ritardo [semestre, media]=0-0.5 18 ==>
Valutazione Insegnamento - Docente - Val [media pesata]=8-10 12
conf:(0.67) < lift:(1.27)> lev:(0.04) [2] conv:(1.22)
- 65 8. Valutazione Insegnamento - Docente - Val [media pesata]=8-10 30
==> Produttivita Studenti - Ritardo [semestre, media]=0-0.5 12
conf:(0.4) < lift:(1.27)> lev:(0.04) [2] conv:(1.08)
- 66 9. Produttivita Studenti - Ritardo [semestre, media]=0-0.5 18 ==>
Produttivita Studenti - Voto [media]=25-28 10 conf:(0.56) <
lift:(1.17)> lev:(0.03) [1] conv:(1.05)
- 67 10. Produttivita Studenti - Voto [media]=25-28 27 ==> Produttivita
Studenti - Ritardo [semestre, media]=0-0.5 10 conf:(0.37) <
lift:(1.17)> lev:(0.03) [1] conv:(1.03)
-

Scartando immediatamente le regole già ottenute nell'analisi precedente — quelle relative alle correlazioni fra il ritardo nel superamento e il voto all'esame — si consideri la seguente sintesi delle regole trovate:

- Voto medio-basso \longleftrightarrow Valutazione docente sufficiente
- Voto medio-alto \longleftrightarrow Valutazione docente ottima
- Ritardo basso \longleftrightarrow Valutazione docente ottima

La correlazione fra la valutazione del docente e i risultati ottenuti appare molto forte, ma purtroppo è anche del tutto analoga a quella fra i risultati e la valutazione generale del corso.

8.2.3 Focus sulla Deviazione Standard

Nel data set `min_d.csv` sono state mantenute informazioni complementari rispetto alle medie pesate dei valori che hanno composto l'aggregazione, quali ad esempio lo scarto quadratico medio di un certo attributo. Gli attributi contenenti questi dati sono stati ignorati nelle due precedenti analisi associative, pertanto è sembrato interessante vedere se esiste una qualche regola fra i seguenti parametri:

- Hash del docente
- Identificativo dell'insegnamento
- Percentuale di studenti il cui ritardo nel dare un certo esame supera il semestre
- Percentuale di studenti il cui voto conseguito in un dato esame supera il 24
- Deviazione standard dei voti ottenuti dagli studenti in quell'esame
- Media delle deviazioni standard delle valutazioni dell'insegnamento nei vari paragrafi
- Media delle percentuali di valutazioni dell'insegnamento sufficienti

```

1 === Run information ===
2
3 Scheme:   weka.associations.Apriori -I -N 5 -T 1 -C 1.05 -D 0.001
         -U 0.75 -M 0.4 -S -1.0 -V -c -1
4 Relation: [...]
5 Instances: 54
6 Attributes: 7
7         Hash Docente/i
8         Insegnamento
9         Produttivita Studenti - Ritardo >=1sem [percent]
10        Produttivita Studenti - Voto >= 24 [perc]
11        Produttivita Studenti - Voto [std dev]
12        Valutazione Insegnamento - Std Dev [media pesata]
13        Valutazione Insegnamento - Val >= 6 [media, percent]
14 === Associator model (full training set) ===
15

```

```

16
17 Apriori
18 =====
19
20 Minimum support: 0.4 (22 instances)
21 Minimum metric <lift>: 1.05
22 Number of cycles performed: 600
23
24 Generated sets of large itemsets:
25
26 Size of set of large itemsets L(1): 3
27
28 Large Itemsets L(1):
29 Produttivita Studenti - Voto [std dev]=3-5 35
30 Valutazione Insegnamento - Std Dev [media pesata]=1.5-3 41
31 Valutazione Insegnamento - Val >= 6 [media, percent]=80-100 47
32
33 Size of set of large itemsets L(2): 3
34
35 Large Itemsets L(2):
36 Produttivita Studenti - Voto [std dev]=3-5 Valutazione Insegnamento
   - Std Dev [media pesata]=1.5-3 28
37 Produttivita Studenti - Voto [std dev]=3-5 Valutazione Insegnamento
   - Val >= 6 [media, percent]=80-100 30
38 Valutazione Insegnamento - Std Dev [media pesata]=1.5-3 Valutazione
   Insegnamento - Val >= 6 [media, percent]=80-100 34
39
40 Size of set of large itemsets L(3): 1
41
42 Large Itemsets L(3):
43 Produttivita Studenti - Voto [std dev]=3-5 Valutazione Insegnamento
   - Std Dev [media pesata]=1.5-3 Valutazione Insegnamento - Val >=
     6 [media, percent]=80-100 23
44
45 Best rules found:
46
47 1. Produttivita Studenti - Voto [std dev]=3-5 35 ==> Valutazione
   Insegnamento - Std Dev [media pesata]=1.5-3 28 conf:(0.8) <
   lift:(1.05)> lev:(0.03) [1] conv:(1.05)
48 2. Valutazione Insegnamento - Std Dev [media pesata]=1.5-3 41 ==>
   Produttivita Studenti - Voto [std dev]=3-5 28 conf:(0.68) <

```

```
lift:(1.05)> lev:(0.03) [1] conv:(1.03)
```

Non ci si aspettava di trovare niente di particolarmente significativo riguardo questi aspetti del data set. Invece, sono comparse due regole associative che possiamo riassumere in un'implicazione- doppia come segue:

Deviazione standard del voto: MOLTO GRANDE \longleftrightarrow Deviazione
standard della valutazione dell'insegnamento: GRANDE

Si tratta di un'associazione forte, in quanto le due regole che la compongono hanno entrambe una confidenza abbastanza elevata — R1: 0.8, R2: 0.68 — ed un valore del lift sì minimo, ma comunque positivo.

PATTERN SEQUENZIALI FREQUENTI

Questo capitolo descrive un'analisi effettuata sul solo data set degli studenti. L'idea generale è quella di estrarre dei *pattern sequenziali* dal data set ottenuto nella fase di *preprocessing* (descritta nella Sezione 5.4), per poi analizzarli ulteriormente con degli script Python realizzati *ad hoc*.

9.1 ALGORITMO GSP

Per l'estrazione dei pattern sequenziali **frequenti** è stato fatto uso dell'algoritmo *Generalized Sequential Pattern*. Una breve descrizione del funzionamento di tale algoritmo può essere trovata in [15]:

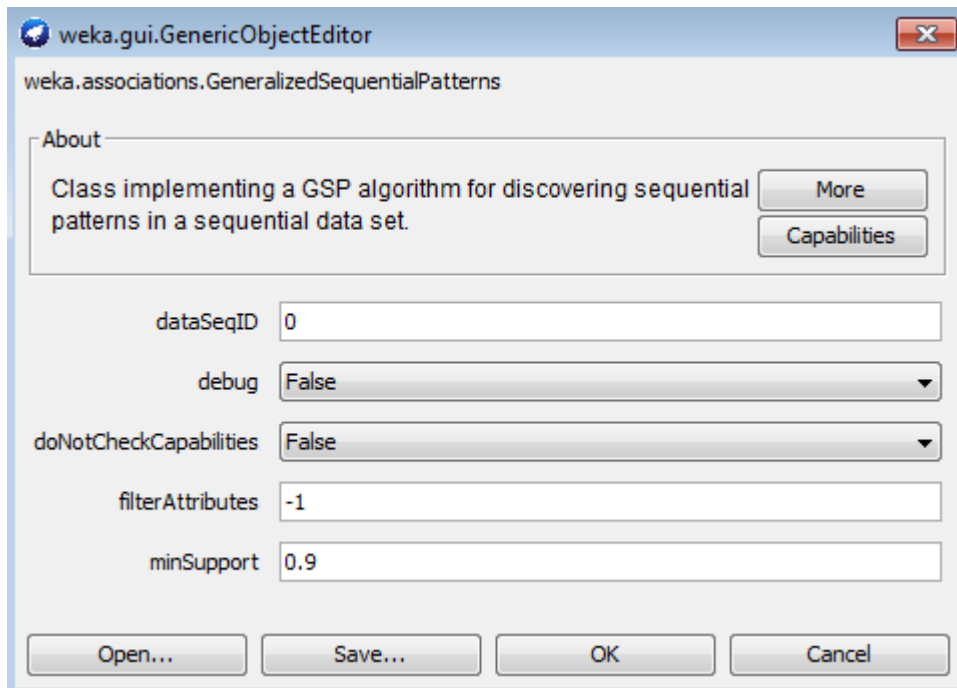
"There are two main steps in the algorithm:

- **Candidate Generation.** Given the set of frequent $(k-1)$ -frequent sequences $F(k-1)$, the candidates for the next pass are generated by joining $F(k-1)$ with itself. A pruning phase eliminates any sequence, at least one of whose subsequences is not frequent.
- **Support Counting.** Normally, a hash tree-based search is employed for efficient support counting. Finally non-maximal frequent sequences are removed.

The algorithms for solving sequence mining problems are mostly based on the *a priori* (level-wise) algorithm."

Parafrasando quanto sopra riportato, l'algoritmo GSP è basato sul **principio Apriori** — lo stesso già incontrato in 8.1.2, per quanto riguardava la ricerca di regole associative.

Figura 49: finestra che mostra i parametri impostabili dell'implementazione di GSP su Weka



L'implementazione utilizzata per l'analisi in oggetto, è quella fornita da Weka. Come si può vedere in Figura 49, si tratta di una versione piuttosto basilare in cui si può specificare un limitato numero di parametri; tuttavia, la (poca) flessibilità concessa è risultata sufficiente al raggiungimento dello scopo.

9.2 SCOPO DELL'ANALISI

Questa analisi si basa sull'idea di considerare come *pattern sequenziale* l'ordine in cui ogni studente ha superato i vari esami del Corso di Laurea, ignorando l'entità del *gap* presente fra il superamento di un esame e l'altro¹ e il voto conseguito.

Avendo ben chiara questa astrazione, lo scopo che ci si è prefissati è quello di confrontare le varie sequenze di esami con una *sequenza ideale*.

¹ Tale elemento, che caratterizza in modo significativo le prestazioni degli studenti, è stato ampiamente considerato nelle analisi descritte negli altri capitoli di questo documento.

Data la codifica degli esami riportata in Tabella 1, si è intesa come tale sequenza ideale la seguente:

1_ASD, 1_ADE, 1_AN1, 1_PRG, 1_MDL, 1_ENG
 2_1_ALG, 2_1_MDP, 2_1_CPS, 2_1_PRC
 2_2_AN2, 2_2_FIG, 2_2_BDS, 2_2_SOP
 3_1_REC, 3_1_INC, 1_1_CAZ, 3_2_CAN, 3_2_ITE

In pratica, è stato considerato come ordine ideale per superare i corsi quello naturale previsto dal Corso di Laurea, contando come non rilevante la posizione degli esami i cui corsi sono svolti nello stesso periodo — ad esempio, non si è considerato importante l'ordine in cui sono stati superati gli esami del primo anno, a patto che siano stati intrapresi con successo prima degli esami successivi.

Una volta ottenute le sequenze di esami che compaiono frequentemente nei dati degli studenti *preprocessati*, lo step successivo consiste nell'individuare in esse gli esami che sono più frequentemente **fuori posto** rispetto alla sequenza ideale precedentemente descritta. Questo ci consente di ampliare la conoscenza di quanto già compreso negli altri capitoli riguardo a quali sono i corsi che creano più difficoltà agli studenti — o che comunque, in questo specifico caso, vengono rimandati più spesso in favore di altri esami teoricamente successivi ad essi.

9.2.1 Esempio di Analisi di una Sequenza Frequente

Si porta un esempio, per assicurarsi una chiarezza totale nell'esposizione di quella che è l'idea fondamentale dell'intera analisi. Si assuma che l'algoritmo GSP restituisca come output il seguente pattern sequenziale frequente:

1_ASD, 1_PRG, 2_1_MDP, 1_MDL, 2_1_PRC

Il fatto che la sequenza è frequente, significa ovviamente che quello è "un pezzo" della carriera di molti² studenti. La caratteristica apparente

² Si utilizza la locuzione *molti* grazie alla sua flessibilità, visto che la quantità di istanze necessarie affinché una sequenza diventi frequente cambia a seconda del supporto inserito come criterio di potatura.

Esame	Codifica
Algoritmi e Strutture Dati	1_ASD
Architetture degli Elaboratori	1_ADE
Analisi 1	1_AN1
Programmazione	1_PRG
Matematica Discreta e Logica	1_MDL
Inglese	1_ENG
Algebra Lineare	2_1_ALG
Metodologie di Programmazione	2_1_MDP
Calcolo della Probabilità e Statistica	2_1_CPS
Programmazione Concorrente	2_1_PRC
Analisi 2	2_2_AN2
Fisica Generale	2_2_FIG
Basi di Dati e Sistemi Informativi	2_2_BDS
Sistemi Operativi	2_2_SOP
Reti di Calcolatori	3_1_REC
Interpreti e Compilatori	3_1_INC
Competenze Aziendali	3_1_CAZ
Calcolo Numerico	3_2_CAN
Informatica Teorica	3_2_ITE

Tabella 1: Tabella che mostra l'associazione fra il nome di ogni esame e la sua codifica nei pattern sequenziali.

di tale sequenza di esami è che Matematica Discreta e Logica, un esame del primo anno, è stato superato *dopo* Metodologie di Programmazione, un esame del primo semestre del secondo anno. Questa contraddizione è evidente, se confrontiamo questa sequenza con quella ideale: significa che molti studenti hanno deviato da quello che è il percorso pianificato.

Quello che ci si prefigge di fare è, quindi, analizzare in questo senso tutte le sequenze frequenti generate da GSP — e saranno così tante da rendere impossibile vagliarle tutte come fatto per quella di esempio. Pertanto, come si vedrà, si useranno degli script per automatizzare tale lavoro.

9.3 PROCEDIMENTO DI ANALISI

Come si è appena accennato, il vero scopo di ottenere dei pattern sequenziali frequenti non è tanto quello di considerarli come tali — e si vedrà presto che, in ogni caso, sarebbe impossibile farlo — ma quello di usarli come base per ulteriori operazioni di *data mining*.

Senza dilungarsi in eccessive descrizioni, si illustra adesso il procedimento seguito per ottenere i risultati che verranno mostrati nella sezione immediatamente successiva alla presente.

- **Preprocessing dei dati grezzi sugli studenti:** si veda quanto già descritto nella Sezione 5.4.
- **Lancio dell'algoritmo GSP:** si tratta del passo fondamentale di tutto il processo, in quanto consente di ottenere i pattern sequenziali frequenti su cui poi lavorare ulteriormente. Come si vedrà in seguito, in questa fase è stato riscontrato il primo, vero problema di *big data* dell'intero lavoro svolto per questa tesi.
- **Mining di pattern inusuali:** gli output prodotti dall'algoritmo GSP vengono analizzati e filtrati, in modo da considerare soltanto i pattern che presentano un'anomalia rispetto alla sequenza di esami ideale. Questo viene realizzato tramite lo script Python riportato in A.5.5.
- **Mining degli esami "fuori posto":** si analizza ulteriormente l'output del precedente step, andando a creare una sintesi di quali particolari

esami hanno causato l'anomalia delle sequenze trovate. Tale scopo è stato raggiunto grazie allo script Python riportato in A.5.6.

9.3.1 Lancio di GSP

Gli output restituiti da GSP sono di questa forma:

```

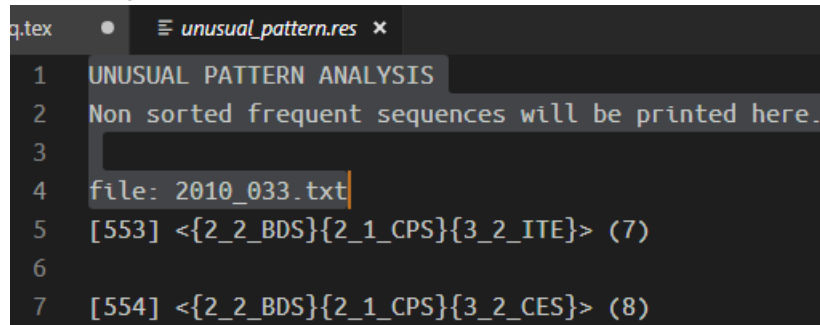
1      GeneralizedSequentialPatterns
2      =====
3
4      Number of cycles performed: 7
5      Total number of frequent sequences: 19478
6
7      Frequent Sequences Details (filtered):
8
9      - 1-sequences
10     [1] <{1_ASD}> (147)
11     [2] <{1_ADE}> (137)
12         - omiss. -
13
14     - 2-sequences
15
16     [1] <{1_ASD}{1_ADE}> (121)
17     [2] <{1_ASD}{1_AN1}> (94)
18         - omiss. -
19
20     - omiss. -

```

Per ogni data set preparato precedentemente, l'algoritmo GSP è stato lanciato tre volte, con tre valori diversi di supporto minimo: 0.33, 0.50 e 0.75; in un caso è stato addirittura tentato il lancio con supporto minimo pari a 0.10. Questo ha prodotto un'elevata mole di sequenze frequenti, che è stata appositamente memorizzata in file di testo, pronti per essere analizzati dai successivi script.

Come già accennato, la complessità dell'algoritmo GSP ha creato il primo, vero problema relativo all'impiego di *big data*: i tempi di analisi sulla versione intera del data set preprocessato — quella che contiene le sequenze di tutte le istanze di studenti a disposizione — esulano dalla

Figura 50: File contenente le sequenze più interessanti estratte dai risultati dell'algoritmo GSP



```
q.tex • unusual_pattern.res x
1 UNUSUAL PATTERN ANALYSIS
2 Non sorted frequent sequences will be printed here.
3
4 file: 2010_033.txt
5 [553] <{2_2_BDS}{2_1_CPS}{3_2_ITE}> (7)
6
7 [554] <{2_2_BDS}{2_1_CPS}{3_2_CES}> (8)
```

trattabilità. Perciò, per ogni data set preparato, è stato applicato il filtro relativo alla lunghezza della sequenza, scartando tutti gli studenti che non hanno completato almeno un terzo del Corso di Laurea, come già descritto in Sezione ottenuta in Sezione 5.4. Nonostante questo filtro, i tempi macchina sono stati comunque molto alti per quanto riguarda il data set completo. Per gli altri, invece, si è trattato di computazioni brevi.

9.3.2 Filtraggio delle Sequenze Inusuali

L'estrazione dei pattern sequenziali interessanti, cioè quelli che presentano almeno un'anomalia rispetto all'ordine ideale degli esami, è stata effettuata lanciando da shell lo script riportato in A.5.5 nella stessa directory dei file contenenti i risultati ottenuti con l'algoritmo GSP.

Lo script creato produce un file *plaintext* nominato `unusual_patterns.res`, del quale si mostra il brevissimo incipit in Figura 50. Tale file semplicemente riporta le righe degli output di GSP che contengono le sequenze con le caratteristiche sopra descritte.

9.3.3 Sommario degli Esami Non Congruenti

Il passo finale del procedimento si completa lanciando lo script contenuto nel file `out_of_place.py`, riportato in A.5.6.

Lo script scorre tutte le sequenze del file precedentemente generato,

individua quali esami la rendono incongruente con la sequenza ideale e li memorizza, contando quante volte un esame è comparso in un pattern sequenziale frequente che lo vede "fuori posto".

Il suo output consiste, in tutta semplicità, nello stampare la struttura dati che ha memorizzato il conto degli esami — non prima di aver trasformato il numero grezzo di istanze che contengono gli item, valore privo di significato assoluto, in una più fruibile percentuale rispetto al totale degli esami individuati.

```
{ '1_MDL': 22.16, '1_ADE': 1.55, '2_2_SOP': 0.44, '2_2_FIG': 42.72,
  '2_1_MDP': 13.69, '2_1_CPS': 1.66, '2_1_PRC': 16.39, '2_1_ALG': 1.39
}
```

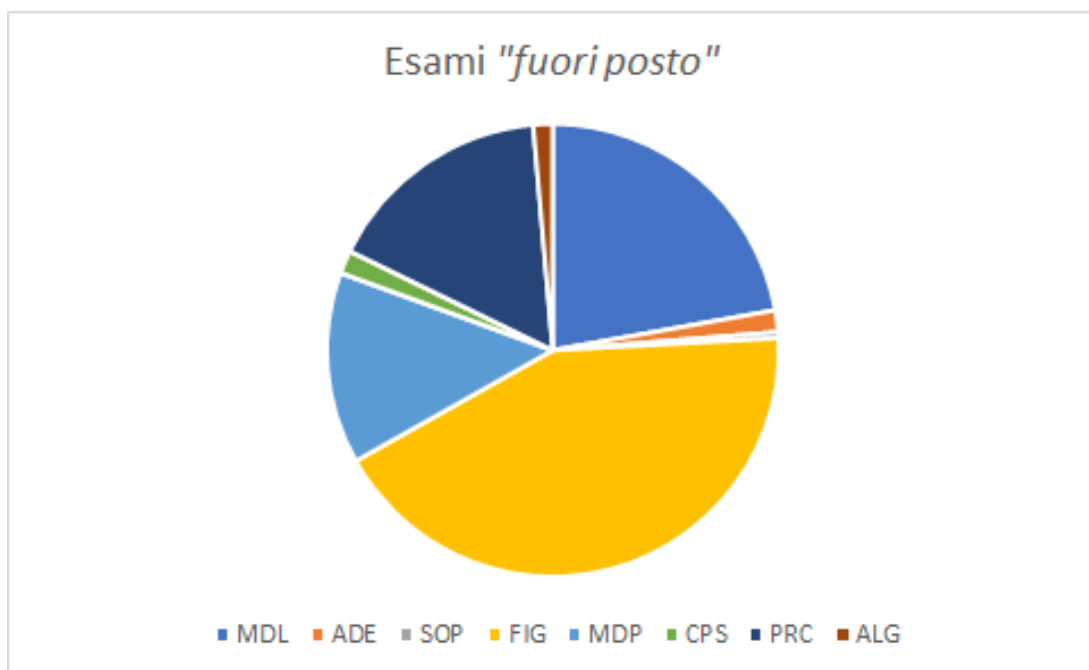
9.4 RISULTATI FINALI DELL'ANALISI

Come appena visto, si è infine arrivati a quello che, abbandonandoci alla suggestiva metafora principale del *data mining*, è il vero oro finalmente estratto dall'intera mole di dati della produttività degli studenti.

Con l'ausilio di un foglio di calcolo, le informazioni estratte sono state incanalate nel diagramma a torta mostrato in Figura 51.

Grazie al potere espressivo del grafico, si osserva senza difficoltà che l'esame che viene rimandato più spesso è di gran lunga Fisica Generale, fatto del quali si è trovato conferma nelle altre analisi illustrate in questo documento. Per la stessa ragione, non è da considerarsi sorprendente la presenza — minore, ma comunque massiccia — di Matematica Discreta e Logica, mentre appaiono più curiosi i dati su Programazione Concorrente e Metodologie di Programazione.

Figura 51: Grafico a torta che riassume la composizione degli esami più spesso rimandati in favore di altri ad essi successivi



CONCLUSIONI E FUTURI SVILUPPI

Dopo aver ampiamente esposto e illustrato tutto il lavoro fatto, non resta che trarre una sintesi conclusiva.

Quanto è stato portato alla luce dalle varie attività di *data mining* che sono state svolte rappresenta sicuramente soltanto una minima parte delle informazioni preziosamente racchiuse nella mole di dati scelta come base. Ci si può comunque definire soddisfatti di quanto ottenuto, in quanto esaurire la "miniera" in quest'unica occasione non è mai stato l'obiettivo prefisso: al contrario, ciò che ha reso importante questa serie di analisi è stata la possibilità di affinare l'intuito e la fantasia necessaria al *data scientist* per poter svolgere efficacemente il suo lavoro.

Non si presenteranno speculazioni in merito alle implicazioni di quanto estratto, in questa sede: ciò trascenderebbe lo scopo dell'intera tesi e la competenza dell'autore. Si è certi però che questo lavoro possa essere una base di partenza per ulteriori approfondimenti, che potranno fornire un solido supporto per scoprire le cause di alcuni degli aspetti relativi a quanto studiato.

Magari, proseguendo per questa strada, si riuscirà a portare un concreto miglioramento al Corso di Laurea in Informatica.

BIBLIOGRAFIA

- [1] <https://docs.mongodb.com/manual/> — *The MongoDB 3.6 Manual* — MongoDB, Inc (Cited on page 23.)
- [2] <https://it.wikipedia.org/wiki/MongoDB> — *MongoDB* — Wikipedia, l'enciclopedia libera (Cited on page 23.)
- [3] <https://api.mongodb.com/python/3.6.0/> — *PyMongo 3.6.0 Documentation* — MongoDB, Inc (Cited on page 24.)
- [4] <https://docs.python.org/3.6/> — *Python 3.6.6rc1 Documentation* — Python Software Foundation (Cited on page 24.)
- [5] <https://it.wikipedia.org/wiki/Python> — *Python* — Wikipedia, l'enciclopedia libera (Cited on page 23.)
- [6] <https://www.gnu.org/software/make/> — *GNU Make* — Free Software Foundation (Cited on page 25.)
- [7] <https://www.cs.waikato.ac.nz/ml/weka/documentation.html> — *WEKA Manual for Version 3-8-2* — Machine Learning Group at the University of Waikato (Cited on page 27.)
- [8] <https://it.wikipedia.org/wiki/Weka> — *Weka* — Wikipedia, l'enciclopedia libera (Cited on pages 26 and 27.)
- [9] <https://cran.r-project.org/manuals.html> — *The R Manuals* — The R Foundation (Cited on page 28.)
- [10] [https://it.wikipedia.org/wiki/R_\(software\)](https://it.wikipedia.org/wiki/R_(software)) — *R (software)* — Wikipedia, l'enciclopedia libera (Cited on page 27.)
- [11] <https://documentation.libreoffice.org/en/> — *Calc Guide* — Libre Office, The Document Foundation (Cited on page 28.)
- [12] *An Analysis of Courses Evaluation Through Clustering* — Renza Campagni, Donatella Merlini, Maria Cecilia Verri (Cited on page 11.)

- [13] *Introduction to Data Mining* — Pang-Ning Tan, Michael Steinbach, Vipin Kumar (Cited on pages 80, 81, and 101.)
- [14] <https://it.wikipedia.org/wiki/Clustering> — *Clustering* — Wikipedia, l'enciclopedia libera (Cited on page 79.)
- [15] https://en.wikipedia.org/wiki/GSP_algorithm — *GSP algorithm* — Wikipedia, l'enciclopedia libera (Cited on page 115.)
- [16] *The Pragmatic Programmer: From Journeyman to Master* — Andrew Hunt, David Thomas (Cited on page 21.)



MAKEFILE E CODICE PYTHON

A.1 PULIZIA DELLE COLLECTION

Modulo Python per la pulizia delle *collection*:

```
1  """Preliminary filtering of the data with usefulness criteria."""
2  import hashlib
3
4  QSET_OLD = {'D1': 'Carico di lavoro accettabile',
5             'D2': 'Organizzazione corso',
6             'D3': 'Orario consente studio individuale',
7             'D4': 'Carico di studio proporzionato a crediti',
8             'D5': 'Materiale didattico adeguato',
9             'D6': 'Attivita integrative utili',
10            'D7': 'Modalita esame chiare',
11            'D8': 'Orari rispettati',
12            'D9': 'Docente reperibile',
13            'D10': 'Docente stimola interesse',
14            'D11': 'Docente chiaro',
15            'D12': 'Docente disponibile ed esauriente',
16            'D13': 'Aule lezioni adeguate',
17            'D14': 'Strumenti e locali adeguati',
18            'D15': 'Conoscenze preliminari sufficienti',
19            'D16': 'Argomenti trattati nuovi o integrativi',
20            'D17': 'Argomenti interessanti',
21            'D18': 'Soddisfazione complessiva corso'
22          }
23
24  QSET_GEN = {'D4': 'Conoscenze preliminari sufficienti',
25             'D5': 'Argomenti trattati nuovi o integrativi',
26             'D6': 'Carico di studio proporzionato a crediti',
```

```

27         'D7': 'Materiale didattico adeguato',
28         'D8': 'Attivita integrative utili',
29         'D9': 'Modalita esame chiare',
30         'D10': 'Orari didattica rispettati',
31         'D11': 'Docente stimola interesse',
32         'D12': 'Docente chiaro',
33         'D13': 'Docente reperibile',
34         'D14': 'Docente disponibile ed esauriente',
35         'D17': 'Argomenti interessanti',
36         'D18': 'Soddisfazione complessiva corso',
37         'D19': 'Copertura programma a lezione',
38         'D20': 'Prove intermedie utili',
39         'D21': 'Prove intermedie danneggiano frequenza'
40     }
41
42
43 class Cleaner:
44     """Cleaning of the teachings evaluation collections's
45     documents."""
46     _cleaned = list()
47
48     def __init__(self, destination):
49         self._dest = destination
50         self._qset = None
51
52     def set_qset(self, qset):
53         """Simple setter for the questions set. """
54         self._qset = qset
55
56     def drop(self):
57         """Drop the original collections that has been cleaned."""
58         for coll in self._cleaned:
59             coll.drop()
60
61     def clean(self, source, year, delete):
62         """Do the work and insert cleaned docs into destination
63         collection."""
64         if self._qset is None:
65             raise Exception('Questions Set not set!')
66
67         for doc in source.find():

```

```

66         if delete:
67             source.delete_one(doc)
68
69         doc = _clarify_questions(doc, self._qset)
70         doc = _time_ref(doc, year)
71         doc = _polish(doc)
72
73         self._dest.insert_one(doc)
74
75     self._cleaned.append(source)
76
77
78 class FinalCleaner:
79     """Polish the final, minable collection before exporting."""
80     _cleaned = list()
81
82     def __init__(self, collection):
83         self._collection = collection
84
85     def clean(self, rename, remove, uppercase):
86         """Self explaining method."""
87         for doc in self._collection.find():
88
89             self._collection.delete_one(doc)
90             self._cleaned.append(doc)
91             newdoc = doc
92
93             for ren in rename:
94                 for key in newdoc:
95                     if ren['old'] in key:
96                         newdoc[key.replace(ren['old'], ren['new'])] =
97                             newdoc.pop(key)
98
99             for rem in remove:
100                 for key in list(newdoc.keys()):
101                     if rem in key:
102                         del newdoc[key]
103
104             for upp in uppercase:
105                 newdoc[upp] = newdoc[upp].upper()

```

```

106         # clean meaningless attributes
107         for key in list(newdoc.keys()):
108             if str(newdoc[key]).lower() == 'n.c.':
109                 del newdoc[key]
110
111         self._collection.insert_one(newdoc)
112
113     def get_cleaned(self):
114         """For debug purposes."""
115         return self._cleaned
116
117
118     def _polish(doc):
119         del doc[''] # little quirk by mongoimport
120         del doc['CID'] # useless as a key for this application
121         del doc['Corso'] # always 'INFORMATICA' since it is the object of
122             this study
123         del doc['Tipo corso'] # as above, always 'INFORMATICA'
124
125         # simply clearer
126         doc['P<6'] = doc.pop('P1')
127         doc['P>=6'] = doc.pop('P2')
128
129         # hide teacher name
130         teacher_hash =
131             hashlib.sha1(doc['Docente/i'].encode('utf-8')).hexdigest()[12:]
132         del doc['Docente/i']
133         doc['Hash Docente/i'] = teacher_hash
134
135         return doc
136
137     def _time_ref(doc, year):
138         doc['Anno Accademico'] = str(year) + '-' + str(year + 1)
139         return doc
140
141     def _clarify_questions(doc, qset):
142         doc['Oggetto Valutazione'] = qset[doc['Q']]
143
144         del doc['Q']

```

```
145     del doc['Quesito']
146
147     return doc
```

Script che utilizzano il modulo, riferiti nel makefile:

Listing A.1: dataset_clean.py

```
1 from pymongo import MongoClient
2
3 from mymodules import cleanings
4
5 coll = MongoClient().exams['minable']
6
7 for doc in coll.find():
8
9     cleaned = False
10    old_doc = doc
11
12    for key in list(doc.keys()):
13        if doc[key] == 'n.c.':
14            del doc[key]
15            cleaned = True
16
17    if cleaned:
18        coll.delete_one(old_doc)
19        coll.insert_one(doc)
20
21 cleanings.FinalCleaner(coll).clean([], [], ['Insegnamento'])
22
23 MongoClient().exams['rawStudentsPr1013'].drop()
24 MongoClient().exams['teachEval_aggr'].drop()
25 MongoClient().exams['sprod'].drop()
```

Listing A.2: teva_clean.py

```
1 """Clean freshly imported teaching evaluation collections."""
2
3 from pymongo import MongoClient
4
5 from mymodules import cleanings
```

```

6
7 scheme = MongoClient().exams
8 teval = scheme.create_collection("teachEval")
9
10 cln = cleanings.Cleaner(teval)
11 cln.set_qset(cleanings.QSET_OLD)
12 cln.clean(scheme.rawTeachingsEv1011, 2010, True)
13
14 cln.set_qset(cleanings.QSET_GEN)
15 cln.clean(scheme.rawTeachingsEv1112, 2011, True)
16 cln.clean(scheme.rawTeachingsEv1213, 2012, True)
17 cln.clean(scheme.rawTeachingsEv1314, 2013, True)
18 cln.clean(scheme.rawTeachingsEv1415, 2014, True)
19 cln.clean(scheme.rawTeachingsEv1516, 2015, True)
20 cln.clean(scheme.rawTeachingsEv1617, 2016, True)
21
22 cln.drop()

```

A.2 AGGREGAZIONE DEGLI ATTRIBUTI

Modulo Python per l'aggregazione degli attributi:

```

1 """Data aggregation utility object, strongly coupled with
   applicative logic so beware."""
2
3 from datetime import datetime
4
5
6 class Aggregator:
7     """Abstract factorization of common stuff."""
8
9     def __init__(self, source, destination):
10         self._source = source
11         self._dest = destination
12
13     def drop(self):
14         """Drop the original collection that has been aggregated."""
15         self._source.drop()
16
17     def aggregate_stud(self, coorte):

```



```

18     """Class signature function."""
19     raise NotImplementedError('Abstract method!')
20
21     def aggregate_par(self):
22         """Class signature function."""
23         raise NotImplementedError('Abstract method!')
24
25
26 class StudAggregator(Aggregator):
27     """ Data aggregation object from students instances to aggregate
        data."""
28
29     _EX_KEYS = [{'date': 'data_ASD', 'name': 'ASD'},
30                 {'date': 'data_MDL', 'name': 'MDL'},
31                 {'date': 'data_PRG', 'name': 'PRG'},
32                 {'date': 'data_ANI', 'name': 'ANI'},
33                 {'date': 'data_ARC', 'name': 'ARC'},
34                 {'date': 'data_ALG', 'name': 'ALG'},
35                 {'date': 'data_CPS', 'name': 'CPS'},
36                 {'date': 'data_MP', 'name': 'MP'},
37                 {'date': 'data_PC', 'name': 'PC'},
38                 {'date': 'data_FIS', 'name': 'FIS'},
39                 {'date': 'data_BDSI', 'name': 'BDSI'},
40                 {'date': 'data_SO', 'name': 'SO'},
41                 {'date': 'data_ANII', 'name': 'ANII'},
42                 {'date': 'data_CAL', 'name': 'CAL'},
43                 {'date': 'data_IT', 'name': 'IT'},
44                 {'date': 'data_RETI', 'name': 'RETI'},
45                 {'date': 'data_IntC', 'name': 'IntC'}
46             ]
47
48     def aggregate_stud(self, coorte):
49         exams = _init_exam_docs()
50
51         for doc in self._source.find():
52             for keys in self._EX_KEYS:
53
54                 if _exam_done_in_ref_period(doc, coorte):
55                     new_doc = exams[keys['name']]
56                     a_a = int(coorte) + new_doc['Anno'] - 1
57                     new_doc['Anno Accademico'] = str(a_a) + '-' +

```

```

        str(int(a_a) + 1)
58         _update_doc(doc, new_doc, keys['name'], keys['date'])
59
60     for i in exams:
61         if exams[i]['upd']:
62             _avg(exams[i])
63             _std_dev(exams[i])
64             _perc(exams[i])
65             _delay(exams[i])
66
67         del exams[i]['Voti']
68         del exams[i]['Date']
69         del exams[i]['Anno']
70         del exams[i]['Coorti']
71         del exams[i]['Sem']
72         del exams[i]['upd']
73
74         self._dest.insert_one(exams[i])
75
76     def aggregate_par(self):
77         raise NotImplementedError('Wrong class!')
78
79
80     def _init_exam_docs():
81         exams = {'ASD': {'Insegnamento': 'Algoritmi e strutture dati',
82                          'Anno': 1, 'Sem': 6},
83                  'MDL': {'Insegnamento': 'Matematica discreta e logica',
84                          'Anno': 1, 'Sem': 6},
85                  'PRG': {'Insegnamento': 'Programmazione', 'Anno': 1,
86                          'Sem': 6},
87                  'ANI': {'Insegnamento': 'Analisi I: calcolo differenziale
88                          ed integrale', 'Anno': 1, 'Sem': 6},
89                  'ARC': {'Insegnamento': 'Architetture degli elaboratori',
90                          'Anno': 1, 'Sem': 6},
91                  'ALG': {'Insegnamento': 'Algebra lineare', 'Anno': 2,
92                          'Sem': 1},
93                  'CPS': {'Insegnamento': 'Calcolo delle probabilita e
94                          statistica', 'Anno': 2, 'Sem': 1},
95                  'MP': {'Insegnamento': 'Metodologie di programmazione',
96                          'Anno': 2, 'Sem': 1},
97                  'PC': {'Insegnamento': 'Programmazione concorrente',

```

```

    'Anno': 2, 'Sem': 1},
90     'FIS': {'Insegnamento': 'Fisica generale', 'Anno': 2,
            'Sem': 6},
91     'BDSI': {'Insegnamento': 'Basi di dati e sistemi
            informativi', 'Anno': 2, 'Sem': 6},
92     'SO': {'Insegnamento': 'Sistemi operativi', 'Anno': 2,
            'Sem': 6},
93     'ANII': {'Insegnamento': 'Analisi 2: funzioni in piu
            variabili', 'Anno': 2, 'Sem': 6},
94     'CAL': {'Insegnamento': 'Calcolo numerico', 'Anno': 3,
            'Sem': 6},
95     'IT': {'Insegnamento': 'Informatica teorica', 'Anno': 3,
            'Sem': 1},
96     'RETI': {'Insegnamento': 'Reti di calcolatori', 'Anno': 3,
            'Sem': 1},
97     'IntC': {'Insegnamento': 'Interpreti e compilatori',
            'Anno': 3, 'Sem': 1}
98 }
99
100 # init other common fields
101 for i in exams:
102     exams[i]['N [istanze]'] = 0
103     exams[i]['Voti'] = []
104     exams[i]['Date'] = []
105     exams[i]['Coorti'] = []
106     exams[i]['Voto >= 24 [perc]'] = 0
107     exams[i]['upd'] = False
108
109 return exams
110
111
112 def _update_doc(old, new, field_mark, field_date):
113     if int(old[field_mark]) > 0:
114         new['upd'] = True
115
116         new['N [istanze]'] = new['N [istanze]'] + 1
117         new['Voti'].append(int(old[field_mark]))
118
119         if int(old[field_mark]) >= 24:
120             new['Voto >= 24 [perc]'] = new['Voto >= 24 [perc]'] + 1
121

```

```

122         new['Date'].append(old[field_date])
123         new['Coorti'].append(old['coorte'])
124
125
126     def _avg(doc):
127         doc['Voto [media]'] = 0
128         for voto in doc['Voti']:
129             doc['Voto [media]'] = doc['Voto [media]'] + voto
130
131         doc['Voto [media]'] = round(doc['Voto [media]'] / doc['N
            [istanze]'], 2)
132
133
134     def _std_dev(doc):
135         key = 'Voto [std dev]'
136         doc[key] = 0
137         for voto in doc['Voti']:
138             doc[key] = doc[key] + pow((voto - doc['Voto [media]']), 2)
139
140         doc[key] = round(pow(doc[key] / doc['N [istanze]'], 0.5), 2)
141
142
143     def _perc(doc):
144         doc['Voto >= 24 [perc]'] = round(100 * doc['Voto >= 24 [perc]'] /
            doc['N [istanze]'], 2)
145
146
147     def _exam_done_in_ref_period(doc, coorte):
148         return str(doc['coorte']) == str(coorte)
149
150
151     def _delay(stuff):
152         """Calculated on a yearly basis, no sense in detecting semester
            shifts."""
153
154         avg_delay = 0
155         p1year = 0
156         instances = 0
157
158         for i in range(len(stuff['Date'])):
159             correct_time = datetime(stuff['Coorti'][i] + stuff['Anno'],

```

```

        stuff['Sem'], 1)
160     exam_time = datetime.strptime(stuff['Date'][i], '%Y-%m-%d')
161     delay_sem = int((exam_time - correct_time).days / (30 * 6))
162
163     avg_delay = avg_delay + delay_sem
164     if delay_sem >= 1:
165         p1year = p1year + 1
166         instances = instances + 1
167
168     stuff['Ritardo [semestre, media]'] = round(avg_delay / instances,
169                                                2)
170     stuff['Ritardo >=1sem [percent]'] = round((p1year / instances) *
171                                                100)
172
173 class ParAggregator(Aggregator):
174     """Aggregate teachings evaluation per paragraph."""
175
176     _GEN = ['Hash Docente/i', 'Anno Accademico']
177
178     def _get_docs_to_aggregate(self):
179         return self._source.aggregate(
180             [
181                 {"$group": {"_id": {
182                     'Insegnamento': "$Insegnamento",
183                     'Paragrafo': "$Paragrafo",
184                     'Anno Accademico': '$Anno Accademico'}
185                 }
186             ]
187         )
188
189     def aggregate_par(self):
190
191         for group in self._get_docs_to_aggregate():
192
193             # weighted mean for those
194             mean = 0
195             stdev = 0
196             p6 = 0
197

```

```

198         # standard mean for this
199         n = 0
200         i = 0
201
202         last_doc_ref = None # to avoid another db query
203
204         for doc in self._source.find(group['_id']):
205             last_doc_ref = doc
206
207             if doc['N'] != '<5' and int(doc['N']) < 0:
208                 print(doc)
209                 raise Exception('negative stuff')
210
211             try:
212                 mean = mean + (doc['Media'] * doc['N'])
213                 stdev = stdev + (doc['Deviazione standard'] *
214                                 doc['N'])
215                 p6 = p6 + (doc['P>=6'] * doc['N'])
216
217                 n = n + doc['N']
218                 i = i + 1
219
220             except TypeError: # do not count missing values for means
221                 pass
222
223         try:
224             mean = round(mean / n, 2)
225             stdev = round(stdev / n, 2)
226             p6 = round(p6 / n, 2)
227             n = int(n / i)
228         except ZeroDivisionError: # it means all values are missing
229             mean = 'n.c.'
230             stdev = 'n.c.'
231             p6 = 'n.c.'
232             n = 'n.c.'
233
234         if n != 'n.c.' and n < 0:
235             print('n' + str(n))
236             print('i' + str(i))
237             raise Exception('negative stuff')

```

```

238         self._dest.insert_one(self._construct_doc(group['_id'],
239                                                     last_doc_ref, [mean, stdev, p6, n]))
240
241     def aggregate_stud(self, coorte):
242         raise NotImplementedError('Wrong class!')
243
244     def _construct_doc(self, skel, ref_lst_doc, aggr_attr):
245         newdoc = skel
246
247         for attr_gen in self._GEN:
248             newdoc[attr_gen] = ref_lst_doc[attr_gen]
249
250         newdoc['Val [media pesata]'] = aggr_attr[0]
251         newdoc['Std Dev [media pesata]'] = aggr_attr[1]
252         newdoc['Val >= 6 [percent]'] = aggr_attr[2]
253         newdoc['N [istanze]'] = aggr_attr[3]
254
255         if newdoc['N [istanze]'] != 'n.c.' and int(newdoc['N
256             [istanze]']) < 0:
257             print(newdoc)
258             print(aggr_attr[3])
259             raise Exception('negative stuff')
260
261         return newdoc

```

Script che utilizza il modulo, riferito nel makefile:

Listing A.3: teval_aggr.py

```

1  from pymongo import MongoClient
2
3  from mymodules import aggregates
4
5  scheme = MongoClient().exams
6  teval = scheme["teachEval"]
7  teval_aggr = scheme.create_collection('teachEval_aggr')
8
9  aggr = aggregates.ParAggregator(teval, teval_aggr)
10 aggr.aggregate_par()
11 aggr.drop()

```

Listing A.4: stud_aggr.py

```

1 from pymongo import MongoClient
2
3 from mymodules import aggregs
4
5 scheme = MongoClient().exams
6 sprod = scheme.create_collection("sprod")
7
8 agg = aggregs.StudAggregator(scheme.rawStudentsPr1013, sprod)
9
10 agg.aggregate_stud('2010')
11 agg.aggregate_stud('2011')
12 agg.aggregate_stud('2012')
13 agg.aggregate_stud('2013')

```

A.3 DISCRETIZZAZIONE

Modulo Python per la discretizzazione degli attributi:

```

1 """Discretization of countinous attributes in datasets."""
2
3
4 class _DiscRange:
5     def __init__(self, minimum, maximum):
6         self._label = str(minimum) + '-' + str(maximum)
7         self._min = minimum
8         self._max = maximum
9
10    def in_range(self, value):
11        """Are you happy, damned linter? What could this function
12           possibly do?"""
13        return self._min <= value <= self._max
14
15    def get_label(self):
16        """As above."""
17        return self._label
18
19 def _discretize(value, ranges):
20     if value == 'n.c.' or value == '<5':

```



```

21     return value.upper()
22
23     range_found = False
24
25     for rng in ranges:
26         if rng.in_range(float(value)):
27             value = rng.get_label()
28             range_found = True
29             break
30
31     if not range_found:
32         value = 'OUT_OF_RANGE'
33
34     return value
35
36
37 def discretize(source, dest, keys, ranges, drop):
38     """Discretization of countinuous attributes in datasets.
39     WARNING: keys and ranges must match positionally!"""
40     if len(keys) != len(ranges):
41         raise Exception('Keys and ranges length are different!')
42
43     for doc in source.find():
44         if drop:
45             source.delete_one(doc)
46
47         for i in range(len(keys)):
48             for k in list(doc.keys()):
49                 if keys[i] in k:
50                     doc[k] = _discretize(doc[k], ranges[i])
51
52     dest.insert_one(doc)
53
54     if drop:
55         source.drop()

```

Script che contiene le dichiarazioni dei *range* di discretizzazione:

Listing A.5: PREPR_PARAMS.py

```

1 """General preprocessing parameters."""

```

```

2
3 from mymodules import discretize
4
5 VAL_SCORE = {discretize._DiscRange(0, 4),
6               discretize._DiscRange(4, 6),
7               discretize._DiscRange(6, 8),
8               discretize._DiscRange(8, 10)
9               }
10
11 STD_DEV = {discretize._DiscRange(0, 1.5),
12            discretize._DiscRange(1.5, 3),
13            discretize._DiscRange(3, 5),
14            discretize._DiscRange(5, 8)
15            }
16
17 MARKS = {discretize._DiscRange(18, 22),
18          discretize._DiscRange(22, 25),
19          discretize._DiscRange(25, 28),
20          discretize._DiscRange(28, 31)
21          }
22
23 PERCENT = {discretize._DiscRange(0, 20),
24            discretize._DiscRange(20, 40),
25            discretize._DiscRange(40, 60),
26            discretize._DiscRange(60, 80),
27            discretize._DiscRange(80, 100)
28            }
29
30 STUDENTS = {discretize._DiscRange(0, 25),
31             discretize._DiscRange(25, 50),
32             discretize._DiscRange(50, 100),
33             discretize._DiscRange(100, 200)
34             }
35
36 YEARS = {discretize._DiscRange(0, 0.5),
37           discretize._DiscRange(0.5, 1),
38           discretize._DiscRange(1, 1.5),
39           discretize._DiscRange(1.5, 2),
40           discretize._DiscRange(2, 2.5),
41           discretize._DiscRange(2.5, 3)
42           }

```

```

43
44
45 # eventually dropped instance pruning in preprocessing
46 def aggregation_diff_pruning(n_1, n_2):
47     """Returns if aggregated instance is to be pruned,
48     based on number of single instance considered for aggregation."""
49     diff = abs(n_1 - n_2)
50     return diff > min(n_1, n_2)
51     # return False

```

Script che utilizza il modulo, riferito nel makefile:

Listing A.6: dataset_discretize.py

```

1 from pymongo import MongoClient
2
3 from mymodules import discretize
4
5 import PREPR_PARAMS as PP
6
7 scheme = MongoClient().exams
8 scheme.create_collection("minable_discretized")
9 scheme.create_collection("minable_min_discretized")
10
11 keys_substrings = ['Val [media pesata]', 'Std Dev [media pesata]',
12                   '[std dev]', 'N [istanze]',
13                   '>= 6', '>= 24', 'Voto [media]', 'Ritardo [semestre,
14                   media]', '>=1sem']
15
16 ranges = [PP.VAL_SCORE, PP.STD_DEV, PP.STD_DEV, PP.STUDENTS,
17           PP.PERCENT, PP.PERCENT, PP.MARKS, PP.YEARS, PP.PERCENT]
18
19 discretize.discretize(scheme.minable, scheme.minable_discretized,
20                       keys_substrings, ranges, False)
21 discretize.discretize(scheme.minable_min,
22                       scheme.minable_min_discretized, keys_substrings, ranges, False)

```

A.4 JOIN

Modulo Python per effettuare *join* e *merge* fra due *collection*:

```

1  """Attribute merging for producing a single binarizable
   collection."""
2
3
4  class Merger:
5      """Attribute merging for producing a single binarizable
   collection."""
6
7      def __init__(self, keys, delete):
8
9          if len(keys) > 3:
10             raise Exception('This is application specific: what are you
               trying to do?')
11
12         self._keys = keys
13         self._delete = delete
14
15         self._bounce_gen = None
16         self._bounce_spec = None
17         self._discriminant = None
18         self._pref = None
19
20     def set_specific_keys(self, bounce_spec, discriminant, pref):
21         """Self explaining method."""
22         self._bounce_spec = bounce_spec
23         self._discriminant = discriminant
24         self._pref = pref
25
26     def set_gen_keys(self, bounce_gen):
27         """Self explaining method."""
28         self._bounce_gen = bounce_gen
29
30     def merge_attributes(self, coll):
31         """Self explaining method."""
32         for k_0 in coll.distinct(self._keys[0]):
33             for k_1 in coll.distinct(self._keys[1], {self._keys[0]:
               k_0}):

```

```

34
35         teach_docs = coll.find(self._get_filter(k_0, k_1))
36         newdoc = self._peek_generalities(teach_docs.next(), k_0,
37                                         k_1)
38         teach_docs.rewind()
39
40         for doc in teach_docs:
41             self._peek_specifics(doc, newdoc)
42
43         if self._delete:
44             coll.delete_many(self._get_filter(k_0, k_1))
45             coll.insert_one(newdoc)
46
47 def merge_collections(self, coll1, coll2, label2, dest):
48     """Finally merge two datasets into a single (minable)
49     collection."""
50
51     for teach_doc in coll1.find():
52         for prod_doc in coll2.find({self._keys[0]:
53                                     teach_doc[self._keys[0]]}):
54
55             if prod_doc[self._keys[1]].upper() ==
56                 teach_doc[self._keys[1]].upper():
57                 newdoc = teach_doc
58                 for key in self._bounce_spec:
59                     newdoc[label2 + ' - ' + key] = prod_doc[key]
60
61                 dest.insert_one(newdoc)
62
63                 if self._delete:
64                     coll2.delete_one(prod_doc)
65                     break
66
67         if self._delete:
68             coll1.delete_one(teach_doc)
69
70     if self._delete:
71         coll1.drop()
72         coll2.drop()
73
74 def _get_filter(self, k_0, k_1):
75     return {self._keys[1]: k_1, self._keys[0]: k_0}

```



```

18         None, None)
19
20 mrg.merge_collections(teval, sprod, 'Produttivita Studenti', dest)

```

Listing A.8: teval_merge.py

```

1 from pymongo import MongoClient
2 from mymodules import merge
3
4 mrg = merge.Merger(['Anno Accademico', 'Insegnamento'], True)
5
6 mrg.set_gen_keys(['Hash Docente/i'])
7
8 mrg.set_specific_keys(['Val [media pesata]',
9                        'Std Dev [media pesata]',
10                       'Val >= 6 [percent]',
11                       'N [istanze]'],
12
13                       'Paragrafo',
14                       'Valutazione Insegnamento - ')
15
16 mrg.merge_attributes(MongoClient().exams['teachEval_aggr'])

```

A.5 PRODUZIONE DATA SET

A.5.1 Minimizzazione degli Attributi sul Data Set Unito

Script per l'aggregazione totale della valutazione degli insegnamenti sul data set unito, riferito nel makefile:

Listing A.9: dataset_min.py

```

1 from pymongo import MongoClient
2
3 scheme = MongoClient().exams
4 new_coll = scheme.create_collection('minable_min')
5 coll = scheme['minable']
6
7 v = 'Valutazione Insegnamento'
8
9 for doc in coll.find():

```

```

10     n = 0
11     val = 0
12     std = 0
13     p6 = 0
14     i = 0
15     for key in list(doc.keys()):
16         if v in key:
17             if 'Val [media pesata]' in key:
18                 val = val + doc[key]
19             if 'Std Dev [' in key:
20                 std = std + doc[key]
21             if 'Val >= 6 [' in key:
22                 p6 = p6 + doc[key]
23             if 'N [istanze]' in key:
24                 n = n + doc[key]
25                 i = i + 1
26
27         del doc[key]
28
29     if i != 0:
30         doc[v + ' - N [media, istanze]'] = int(round(n / i, 0))
31         doc[v + ' - Std Dev [media pesata]'] = round(std / i, 2)
32         doc[v + ' - Val >= 6 [media, percent]'] = round(p6 / i, 2)
33         doc[v + ' - Val [media pesata]'] = round(val / i, 2)
34
35     new_coll.insert_one(doc)

```

A.5.2 Condensazione delle Valutazioni sui Insegnamenti

Script per l'aggregazione della valutazione degli insegnamenti in un data set minimale, riferito nel makefile:

Listing A.10: dataset_eval_gen.py

```

1 from pymongo import MongoClient
2
3 scheme = MongoClient().exams
4 eval_gen = scheme.create_collection('eval_gen')
5 teval = scheme['teachEval_aggr']
6
7 val = 'Valutazione Complessiva [media pesata]'

```



```

8 sdv = 'Deviazione Standard Complessiva [media pesata]'
9 prc = 'Percentuale Valutazioni Sufficienti [media pesata]'
10 n = 'Numero Valutazioni [media]'
11
12 for group in teval.aggregate([{"$group": {"_id": {"Anno Accademico":
    '$Anno Accademico'}}}]):
13
14     aggr = {val: 0, sdv: 0, prc: 0, n: 0}
15
16     i = 0
17     for doc in teval.find(group['_id']):
18         if doc['Val [media pesata]'] != 'n.c.':
19             aggr[val] = aggr[val] + doc['Val [media pesata]'] * doc['N
                [istanze]']
20             aggr[sdv] = aggr[sdv] + doc['Std Dev [media pesata]'] *
                doc['N [istanze]']
21             aggr[prc] = aggr[prc] + doc['Val >= 6 [percent]'] * doc['N
                [istanze]']
22             aggr[n] = aggr[n] + doc['N [istanze]']
23             i = i + 1
24
25     aggr[val] = round(aggr[val] / aggr[n], 2)
26     aggr[sdv] = round(aggr[sdv] / aggr[n], 2)
27     aggr[prc] = round(aggr[prc] / aggr[n], 2)
28     aggr[n] = int(round(aggr[n] / i, 0))
29
30     aggr['Anno Accademico'] = group['_id']['Anno Accademico']
31
32     eval_gen.insert_one(aggr)

```

A.5.3 Condensazione della Produttività degli Studenti

Script per l'aggregazione della produttività degli studenti in un data set minimale, riferito nel makefile:

Listing A.11: dataset_stud_gen.py

```

1 from pymongo import MongoClient
2
3 scheme = MongoClient().exams
4 stud_gen = scheme.create_collection('stud_gen')

```

```

5
6 prod = scheme.rawStudentsPr1013
7 stud = scheme.sprod
8
9 # aggr doc keys
10 val = 'Valutazione Test Ingresso [media]'
11 pl = 'Studenti Laureati [percent]'
12 ni = 'N [istanze]'
13 mark = 'Voto [media pesata]'
14 delay = 'Ritardo [semestre, media pesata]'
15
16 for group in prod.aggregate([{"$group": {"_id": {'coorte':
    '$coorte'}}}]):
17
18     aggr = {val: 0, pl: 0, ni: 0, 'Coorte Immatricolazione':
        group['_id']['coorte']}
19
20     for doc in prod.find(group['_id']):
21         aggr[ni] = aggr[ni] + 1
22         aggr[val] = aggr[val] + doc['test']
23         if doc['crediti_totali'] == 180:
24             aggr[pl] = aggr[pl] + 1
25
26     aggr[val] = round(aggr[val] / aggr[ni], 2)
27     aggr[pl] = round((aggr[pl] / aggr[ni]) * 100, 2)
28
29     # weighted mean
30     aggr[mark] = 0
31     aggr[delay] = 0
32     n = 0
33
34     a_a = str(group['_id']['coorte']) + '-' +
        str(group['_id']['coorte'] + 1)
35     for doc in stud.find({'Anno Accademico': a_a}):
36         aggr[mark] = aggr[mark] + doc['Voto [media]'] * doc[ni]
37         aggr[delay] = aggr[delay] + doc['Ritardo [semestre, media]'] *
            doc[ni]
38         n = n + doc[ni]
39
40     aggr[mark] = round(aggr[mark] / n, 2)
41     aggr[delay] = round(aggr[delay] / n, 2)

```

```
42
43 stud_gen.insert_one(aggr)
```

A.5.4 Sequenze Ordinate di Esami Superati

Script per estrarre delle sequenze ordinate di esami superati da ogni studente, in un formato direttamente utilizzabile da Weka:

Listing A.12: dataset_stud_gen.py

```
1 import operator
2 import os
3
4 from datetime import datetime
5 from pymongo import MongoClient
6
7 def only_the_dates(doc):
8     keys_to_delete = []
9
10    for key in doc.keys():
11        if "data_" not in key:
12            keys_to_delete.append(key)
13
14    for key in keys_to_delete:
15        del doc[key]
16
17    return doc
18
19
20 def datetime_sorted(doc):
21     keys_to_delete = []
22
23     for key in doc.keys():
24         if doc[key] != 0 and doc[key] != '0000-00-00':
25             doc[key] = datetime.strptime(doc[key], '%Y-%m-%d')
26         else:
27             keys_to_delete.append(key)
28
29     for key in keys_to_delete:
30         del doc[key]
31
```

```

32     sorted_list = sorted(doc, key=doc.get)
33
34     return sorted_list
35
36
37 def fix_names(filename):
38     with open(filename, 'r') as file :
39         filedata = file.read()
40
41     filedata = filedata.replace('data_ANII', '2_2_AN2')
42     filedata = filedata.replace('data_FIS', '2_2_FIG')
43     filedata = filedata.replace('data_BDSI', '2_2_BDS')
44     filedata = filedata.replace('data_SO', '2_2_SOP')
45
46     filedata = filedata.replace('data_ASD', '1_ASD')
47     filedata = filedata.replace('data_ARC', '1_ADE')
48     filedata = filedata.replace('data_ANI', '1_AN1')
49     filedata = filedata.replace('data_PRG', '1_PRG')
50     filedata = filedata.replace('data_MDL', '1_MDL')
51     filedata = filedata.replace('data_INGLESE', '1_ENG')
52
53     filedata = filedata.replace('data_ALG', '2_1_ALG')
54     filedata = filedata.replace('data_MP', '2_1_MDP')
55     filedata = filedata.replace('data_CPS', '2_1_CPS')
56     filedata = filedata.replace('data_PC', '2_1_PRC')
57
58     filedata = filedata.replace('data_RETI', '3_1_REC')
59     filedata = filedata.replace('data_IntC', '3_1_INC')
60     filedata = filedata.replace('data_CAz', '3_1_CAZ')
61
62     filedata = filedata.replace('data_CAL', '3_2_CAN')
63     filedata = filedata.replace('data_IT', '3_2_ITE')
64     filedata = filedata.replace('data_CS', '3_2_CES')
65
66     with open(filename, 'w') as file:
67         file.write(filedata)
68
69 def add_ids(filename, id):
70     with open(filename, 'r') as file :
71         filedata = file.read()
72

```

```

73     id_string = '{'
74
75     for i in range(1, id):
76         id_string = id_string + str(i) + ', '
77
78     id_string = id_string[:-2] # remove last comma
79     id_string = id_string + '}'
80
81     filedata = filedata.replace('{--ADD IDs--}', id_string)
82
83     with open(filename, 'w') as file:
84         file.write(filedata)
85
86
87     def init_file(filename):
88         with open(filename, 'w') as the_file:
89             the_file.write('@relation students_career\n')
90             the_file.write('\n')
91             the_file.write('@attribute StudentID {--ADD IDs--}\n')
92             the_file.write('@attribute ExamID {1_ASD, 1_ADE, 1_AN1, 1_PRG,
93                 1_MDL, 1_ENG, 2_1_ALG, 2_1_MDP, 2_1_CPS, 2_1_PRC, 2_2_AN2,
94                 2_2_FIG, 2_2_BDS, 2_2_SOP, 3_1_REC, 3_1_INC, 3_1_CAZ,
95                 3_2_CAN, 3_2_ITE, 3_2_CES}\n')
96             the_file.write('\n')
97             the_file.write('@data\n')
98
99     os.chdir('../datasets')
100
101     init_file('seq_stud_all.arff')
102     init_file('seq_stud_2010.arff')
103     init_file('seq_stud_2011.arff')
104     init_file('seq_stud_2012.arff')
105     init_file('seq_stud_2013.arff')
106
107     scheme = MongoClient().exams
108     coll = scheme['rawStudentsPr1013']
109     studentID = {'ALL': 1, '2010': 1, '2011': 1, '2012': 1, '2013': 1}
110
111     for doc in coll.find():

```

```

111     coorte = doc['coorte']
112     cfu = int(doc['crediti_totali'])
113
114     min_cfu = 60
115
116     lst = datetime_sorted(only_the_dates(doc))
117
118     if cfu >= min_cfu:
119
120         for exam in lst:
121             with open('seq_stud_all.arff', 'a') as the_file:
122                 the_file.write(str(studentID['ALL']) + ',' + exam +
123                               '\n')
124
125             studentID['ALL'] = studentID['ALL'] + 1
126
127         if coorte == 2010:
128             for exam in lst:
129                 with open('seq_stud_2010.arff', 'a') as the_file:
130                     the_file.write(str(studentID['2010']) + ',' + exam +
131                                   '\n')
132
133                     studentID['2010'] = studentID['2010'] + 1
134
135         if coorte == 2011:
136             for exam in lst:
137                 with open('seq_stud_2011.arff', 'a') as the_file:
138                     the_file.write(str(studentID['2011']) + ',' + exam +
139                                   '\n')
140
141                     studentID['2011'] = studentID['2011'] + 1
142
143         if coorte == 2012:
144             for exam in lst:
145                 with open('seq_stud_2012.arff', 'a') as the_file:
146                     the_file.write(str(studentID['2012']) + ',' + exam +
147                                   '\n')
148
149                     studentID['2012'] = studentID['2012'] + 1
150
151         if coorte == 2013:
152             for exam in lst:
153                 with open('seq_stud_2013.arff', 'a') as the_file:
154                     the_file.write(str(studentID['2013']) + ',' + exam +
155                                   '\n')
156
157                     studentID['2013'] = studentID['2013'] + 1

```

```

147         studentID['2013'] = studentID['2013'] + 1
148
149 fix_names('seq_stud_all.arff')
150 add_ids('seq_stud_all.arff', studentID['ALL'])
151 fix_names('seq_stud_2010.arff')
152 add_ids('seq_stud_2010.arff', studentID['2010'])
153 fix_names('seq_stud_2011.arff')
154 add_ids('seq_stud_2011.arff', studentID['2011'])
155 fix_names('seq_stud_2012.arff')
156 add_ids('seq_stud_2012.arff', studentID['2012'])
157 fix_names('seq_stud_2013.arff')
158 add_ids('seq_stud_2013.arff', studentID['2013'])

```

A.5.5 *Pattern Sequenziali Inusuali*

Script per filtrare i pattern sequenziali frequenti significativi:

Listing A.13: out_of_place.py

```

1 from collections import deque
2 import re
3
4 out_of_place = {}
5 debug = False
6
7
8 def prepare_list(line):
9     line = line.split('{')
10    line = deque(line)
11    line.popleft()
12    line.pop()
13    return list(line)
14
15
16 def get_order(list, index):
17     sliced = list[index]
18     sliced = sliced[:4]
19     return re.sub("[^0-9]", "", sliced)
20
21
22 def proportion():

```

```

23     tot = 0
24
25     for k in out_of_place:
26         tot = tot + out_of_place[k]
27
28     for k in out_of_place:
29         out_of_place[k] = round(out_of_place[k] * 100 / tot, 2)
30
31
32 with open("unusual_pattern.res") as fileobj:
33     for line in fileobj:
34
35         if debug:
36             print('-----')
37             print('line:')
38             print(line)
39
40         if '{' in line:
41             seq = prepare_list(line)
42
43             if debug:
44                 print('seq:')
45                 print(seq)
46
47             first = get_order(seq, 0)
48
49             for i in range(1, len(seq)):
50                 if debug:
51                     print('first:')
52                     print(first)
53                     print('current:')
54                     print(get_order(seq, i))
55                     print('current<first:')
56                     print(get_order(seq, i) < first)
57                     print('key:')
58                     print(seq[i])
59
60                 if get_order(seq, i) < first:
61                     key = re.sub("}", "", seq[i])
62                     out_of_place[key] = out_of_place.get(key, 0) + 1
63             else:

```



```

64         first = get_order(seq, i)
65
66     proportion()
67     print(out_of_place)

```

A.5.6 *Esami più Frequentemente Fuori Posto*

Script per sintetizzare le informazioni presenti nei pattern sequenziali significativi:

Listing A.14: out_of_place.py

```

1  from collections import deque
2  import re
3
4  out_of_place = {}
5  debug = False
6
7
8  def prepare_list(line):
9      line = line.split('{')
10     line = deque(line)
11     line.popleft()
12     line.pop()
13     return list(line)
14
15
16  def get_order(list, index):
17      sliced = list[index]
18      sliced = sliced[:4]
19      return re.sub("[^0-9]", "", sliced)
20
21
22  def proportion():
23      tot = 0
24
25      for k in out_of_place:
26          tot = tot + out_of_place[k]
27
28      for k in out_of_place:
29          out_of_place[k] = round(out_of_place[k] * 100 / tot, 2)

```

```

30
31
32 with open("unusual_pattern.res") as fileobj:
33     for line in fileobj:
34
35         if debug:
36             print('-----')
37             print('line:')
38             print(line)
39
40         if '{' in line:
41             seq = prepare_list(line)
42
43             if debug:
44                 print('seq:')
45                 print(seq)
46
47             first = get_order(seq, 0)
48
49             for i in range(1, len(seq)):
50                 if debug:
51                     print('first:')
52                     print(first)
53                     print('current:')
54                     print(get_order(seq, i))
55                     print('current<first:')
56                     print(get_order(seq, i) < first)
57                     print('key:')
58                     print(seq[i])
59
60                 if get_order(seq, i) < first:
61                     key = re.sub("}", "", seq[i])
62                     out_of_place[key] = out_of_place.get(key, 0) + 1
63                 else:
64                     first = get_order(seq, i)
65
66 proportion()
67 print(out_of_place)

```

A.6 MAKEFILE

Contenuto del *makefile* usato per automatizzare parte delle operazioni:

```

1 DB = exams
2 SCHEME= MongoClient().exams
3 PRDIR= cd prepr &&
4 TDIR= cd thesis &&
5 PY= python3
6 TEX= pdflatex -shell-escape -interaction=nonstopmode -file-line-error
7
8 # useful to spot bad coded modules
9 TIME= /usr/bin/time --format=%e
10
11 .DEFAULT= all
12 all: prepr pdf
13
14
15 #####
16 # LATEX THESIS #
17 #####
18 pdf:
19     $(TDIR) $(TEX) Thesis.tex
20
21 #####
22 # PREPROCESSING STEP #
23 #####
24
25 prep: stud_seq exp_eval_gen exp_stud_gen exp_full exp_min exp_d
26
27 reset_db:
28     mongo $(DB) --eval "db.dropDatabase()"
29
30 import: reset_db
31     $(PRDIR) sh import.sh
32
33 #
34 # teaching evaluation recipes
35 #
36 teval: teval_merge
37

```

```
38 teval_clean: import
39     $(PRDIR) $(TIME) $(PY) teval_clean.py
40
41 teval_aggr: teval_clean
42     $(PRDIR) $(TIME) $(PY) teval_aggr.py
43
44 teval_merge: teval_aggr
45     $(PRDIR) $(TIME) $(PY) teval_merge.py
46
47 teval_gen: teval_aggr
48     $(PRDIR) $(TIME) $(PY) dataset_eval_gen.py
49
50 #
51 # students productivity recipes
52 #
53 stud: stud_aggr
54
55 stud_aggr: import
56     $(PRDIR) $(TIME) $(PY) stud_aggr.py
57
58 stud_gen: stud_aggr
59     $(PRDIR) $(TIME) $(PY) dataset_stud_gen.py
60
61 stud_seq: import
62     $(PRDIR) $(TIME) $(PY) dataset_stud_seq.py
63
64 #
65 # finalize
66 #
67 merged: stud teval
68     $(PRDIR) $(TIME) $(PY) dataset_merge.py
69
70 cleaned: merged
71     $(PRDIR) $(TIME) $(PY) dataset_clean.py
72
73 minified: cleaned
74     $(PRDIR) $(TIME) $(PY) dataset_min.py
75
76 discretized: minified
77     $(PRDIR) $(TIME) $(PY) dataset_discretize.py
78
```

```

79 #
80 # export
81 #
82 EXP= mongoexport
83 FIELDS_FULL= --type=csv --fieldFile=prepr/_exp_fields.txt
84 FIELDS_MIN= --type=csv --fieldFile=prepr/_exp_fields_min.txt
85 FIELDS_STUD= --type=csv --fieldFile=prepr/_exp_fields_stud_gen.txt
86 FIELDS_EVAL= --type=csv --fieldFile=prepr/_exp_fields_eval_gen.txt
87
88 # out of recipes tree, run it manually when needed
89 list_fields:
90     $(PRDIR) sh list_fields.sh
91
92 prep_exp:
93     yes | rm -rf datasets && mkdir datasets
94
95 exp_stud_gen: stud_gen prep_exp
96     $(EXP) --db $(DB) --collection stud_gen $(FIELDS_STUD) >
        datasets/gen_stud.csv
97
98 exp_eval_gen: teval_gen prep_exp
99     $(EXP) --db $(DB) --collection eval_gen $(FIELDS_EVAL) >
        datasets/gen_eval.csv
100
101 exp_full: cleaned prep_exp
102     $(EXP) --db $(DB) --collection minable $(FIELDS_FULL) >
        datasets/full.csv
103
104 exp_min: minified prep_exp
105     $(EXP) --db $(DB) --collection minable_min $(FIELDS_MIN) >
        datasets/min.csv
106
107 exp_d: discretized prep_exp
108     $(EXP) --db $(DB) --collection minable_discretized $(FIELDS_FULL)
        > datasets/full_d.csv
109     $(EXP) --db $(DB) --collection minable_min_discretized
        $(FIELDS_MIN) > datasets/min_d.csv

```
