UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione

TESI DI LAUREA

Progettazione e realizzazione su una campagna di data analytics per investigare le ragioni alla base dell'abbandono degli studi universitari

Design and implementation on a data analytics campaign to investigate the reasons behind university dropouts

Relatore

Prof. Domenico Ursino

Candidata

Francesca Zitoli

Correlatore

Dott. David Malizia

ANNO ACCADEMICO 2023-2024

Sommario

Negli ultimi anni il fenomeno dell'abbandono universitario ha assunto un'importanza sempre maggiore in quanto, comporta profonde implicazioni sia a livello economico che a livello di capitale umano. Per arginare questo fenomeno, le università sono interessate a cercare di capire chi molto probabilmente abbandonerà il proprio percorso di studi per mettere in atto delle misure preventive. In questa tesi sono stati analizzati i dati degli studenti dell'Università di Pisa. In particolare, è stata realizzata una fase di ETL dei dati, seguita da una fase di classificazione e di analisi dei risultati ottenuti.

Keyword: Abbandono Universitario, Extract Trasform and Load, Data Factory, Databricks, Classificazione, Alberi Decisionali, XGBoost, Overfitting, Undersampling

Indice

Introduzione **1**

1 Il problema dell'abbandono degli studi universitari **3**
1.1 Perché l'abbandono universitario è un problema . . . 3
1.2 Situazione in Italia . . . 4
1.3 Cause dell'abbandono . . . 5

2 Acquisizione dei dati per la campagna **7**
2.1 Le fonti . . . 7
2.2 Obbiettivo . . . 8
2.3 Implementazione delle soluzioni . . . 10

3 Attività di Extraction, Transformation and Loading **17**
3.1 Data Factory . . . 17
3.2 Main . . . 18
3.2.1 Load ingestion . . . 19
3.2.2 Cleaning . . . 20
3.2.3 Dimensioni . . . 21
3.2.4 Fatti . . . 22
3.3 Creazione dei metadati . . . 23

4 Progettazione degli algoritmi di predizione **26**
4.1 La classificazione . . . 26
4.2 Databricks . . . 28
4.3 Modelli . . . 29
4.3.1 Decision tree . . . 29
4.3.2 Random forest . . . 30
4.3.3 Logistic regression . . . 31
4.3.4 XGBoost . . . 31
4.3.5 LightGBM . . . 32
4.4 Le metriche . . . 33

5 Implementazione degli algoritmi di predizione **34**
5.1 Preparazione del dataset di training . . . 34
5.2 Fase di training . . . 37
5.3 Overfitting . . . 43

6 Risultati ottenuti e loro commento **46**
6.1 Risultati 46
6.2 Feature importance 48

7 Conclusioni **52**

Bibliografia **54**

Ringraziamenti **56**

Elenco delle figure

1.1 Andamento degli abbandoni in Italia tra il I e il II anno 4
1.2 Andamento degli abbandoni in Italia dal III anno 5

2.1 Flusso dei dati 8
2.2 Schema logico della carriera di uno studente 9
2.3 Schema logico di evento 10
2.4 Pipeline LOAD INGESTION DB 16

3.1 Pipeline `Main` 19
3.2 Attività di copy data per ANS1 19
3.3 Tabelle esterne dedicate ai dati dell'Osservatorio 20
3.4 Pipeline `DIM` 21
3.5 Pipeline `DIM_CALENDARIO` 21
3.6 Pipeline `DIM_STANDARD` 21
3.7 Pipeline `DIM_TITOLO` 22
3.8 Pipeline `FACT` 22
3.9 Esami durante l'Anno Accademico 24
3.10 Andamento dell'accuratezza in base ai periodi presi in considerazione per le previsioni 25
3.11 Andamento delle previsioni all'aumentare del campione di dati 25

4.1 Tecniche di classificazione 27
4.2 Esempio di albero decisionale 30
4.3 Grafico della curva logistic regression 31
4.4 Espansione albero ricerca LGBM 32

5.1 Abbandono universitario per lauree triennali 35
5.2 Abbandono universitario per lauree magistrali 35
5.3 Distribuzione degli eventi lauree triennali 36
5.4 Distribuzione degli eventi per le lauree a ciclo unico 37
5.5 Distribuzione degli eventi per le lauree magistrali 37
5.6 Pipeline del modello XGBoost 38
5.7 Confronto delle matrici di confusione per i corsi triennali 40
5.8 Confronto delle matrici di confusione per i corsi a ciclo unico 41
5.9 Confronto delle matrici di confusione per i corsi magistrali 42
5.10 Matrice di confusione dei corsi a ciclo unico dopo l'undersampling 45

5.11 Matrice di confusione dei corsi magistrali dopo l'undersampling 45

6.1 Riepilogo delle matrici di confusione per i diversi corsi di laurea 47
6.2 Distribuzione delle classi per i diversi corsi di laurea 48
6.3 Beeswarm per i corsi di laurea triennale 49
6.4 Distribuzione degli studenti per età calcolata in giorni per i corsi di laurea triennale 49
6.5 Beeswarm per i corsi di laurea a ciclo unico 50
6.6 Distribuzione degli studenti per età calcolata in giorni per i corsi di laurea a ciclo unico 50
6.7 Beeswarm per i corsi di laurea magistrale 51
6.8 Distribuzione degli studenti per età calcolata in giorni per i corsi di laurea magistrale 51

Elenco delle tabelle

3.1 Valori dell'accuratezza in base al periodo dell'anno e al tipo di corso 24

5.1 Valori delle metriche per i modelli dei corsi triennali 38
5.2 Valori delle metriche per i modelli dei corsi ciclo unico 39
5.3 Valori delle metriche per i modelli dei corsi magistrali 39
5.4 Distribuzione delle classi nel dataset 43
5.5 Distribuzione delle classi nei corsi ciclo unico dopo undersampling 44
5.6 Valori delle metriche per i corsi a ciclo unico dopo l'undersampling 45
5.7 Distribuzione delle classi nei corsi magistrali dopo l'undersampling 45
5.8 Valori delle metriche per i corsi magistrali dopo l'undersampling 45

6.1 Riepilogo delle metriche per i diversi corsi di laurea 47

Introduzione

Negli ultimi anni, il fenomeno dell'abbandono universitario è diventato oggetto di numerosi studi in tutto il mondo. La sua diffusione ha acceso i riflettori su un problema dalle profonde implicazioni, con ricadute negative sia a livello economico che di capitale umano. Le perdite generate dall'abbandono universitario si manifestano su diversi fronti; da un lato, le famiglie degli studenti si ritrovano a sostenere i costi di un percorso formativo che non viene completato, dall'altro, lo Stato investe risorse in individui che non entrano a far parte della forza lavoro qualificata, con un mancato ritorno in termini di produttività e innovazione. L'abbandono universitario rappresenta un problema anche dal punto di vista del capitale umano; un'istruzione elevata, infatti, porta ad un aumento della produttività degli individui e, in un contesto di forte cambiamento tecnologico, come quello che stiamo vivendo negli ultimi anni, la domanda di una forza lavoro istruita è sempre più pressante. Le persone con un livello di istruzione più elevato, infatti, sono in grado di apprendere nuove competenze con maggiore facilità, adattarsi alle nuove tecnologie e ai nuovi modelli di lavoro, risolvere problemi complessi e prendere decisioni in modo efficace, essere più creativi e innovativi. L'abbandono universitario priva, quindi, il Paese di una parte fondamentale del suo capitale umano, con un impatto negativo sulla crescita economica e sulla competitività del sistema produttivo.

L'abbandono universitario in Italia rappresenta una criticità, con differenze significative tra i diversi tipi di corsi di laurea. Nei corsi triennali, il fenomeno assume proporzioni preoccupanti; la percentuale di studenti che abbandona gli studi si attesta al 14.5%, un valore decisamente elevato. Situazione migliore si riscontra per i corsi di laurea a ciclo unico (5-6 anni) e magistrale (2 anni), dove i tassi di abbandono si riducono considerevolmente; le percentuali scendono, rispettivamente, al 8.2% e al 7.3%, pur evidenziando comunque la presenza del problema. Questo è quanto emerge dal rapporto ANVUR [2023] che tiene conto solo degli studenti che si ritirano dal corso di laurea e non prende in considerazione coloro che dopo l'iscrizione non effettuano più alcun tipo di attività, detti "non presentati"; considerando anche quest'ultimo caso le percentuali sarebbero molto più alte e preoccupanti.

Nella presente tesi è stato analizzato il fenomeno dell'abbandono dell'Università di Pisa. In una prima fase viene effettuata la raccolta dei dati da diverse sorgenti; su questi viene effettuata la fase di ETL. L'obbiettivo finale è cercare di prevedere quali studenti abbiano elevata probabilità di abbandono e quali abbiano una bassa probabilità di non terminare il proprio percorso di studi, in modo tale che l'università possa intervenire con delle misure preventive per evitare che ciò avvenga. Per raggiungere tali scopi vengono usati diversi modelli di classificazione e, dopo aver effettuato il training, viene scelto il migliore da utilizzare per effettuare le previsioni.

La presente tesi è composta da sette capitoli strutturati come di seguito specificato:

- Nel Capitolo 1 verrà introdotto il problema dell'abbandono universitario analizzando soprattutto la situazione italiana.
- Nel Capitolo 2 verranno presentati i metodi utilizzati per l'acquisizione dei dati.
- Nel Capitolo 3 saranno illustrate le tecnologie utilizzate e si vedrà come viene effettuata l'intera fase di ETL per pulire e caricare i dati.
- Nel Capitolo 4 verrà analizzato il problema della classificazione, la tecnologia usata per creare i modelli ed, infine, i vari modelli di classificazione che saranno utilizzati e le relative metriche usate per valutarli.
- Nel Capitolo 5 sarà illustrata la suddivisione del database utilizzato per il training dei modelli. Successivamente, verranno presentati i risultati ottenuti dai modelli addestrati e si procederà a determinare quale modello ha prodotto i risultati migliori. Verranno, poi, discussi i problemi affrontati durante il processo e le relative soluzioni adottate.
- Nel Capitolo 6 verranno esposti i risultati finali ottenuti dai classificatori, e si procederà ad un'approfondita valutazione delle performance dei modelli addestrati, identificando le feature che hanno maggiormente inciso sul processo decisionale.
- Infine, nel Capitolo 7, saranno tratte le conclusioni.

CAPITOLO 1

Il problema dell'abbandono degli studi universitari

Questo capitolo introduttivo affronterà il complesso problema dell'abbandono degli studi universitari, un fenomeno in cui gli studenti iscritti a un corso accademico non riescono a completarlo, interrompendo il percorso prima di conseguire la laurea. Tale situazione costituisce una sfida significativa, poiché comporta una perdita non solo di tempo ma anche di risorse considerevoli, sia per le famiglie coinvolte sia per il sistema statale.

In particolare, ci concentreremo sull'analisi della realtà italiana riguardante gli abbandoni universitari e sulle possibili cause che inducono gli studenti a prendere questa difficile decisione. Esaminare attentamente tali motivazioni è cruciale per comprendere come affrontare e prevenire questo problema che impatta profondamente sul percorso formativo e sul futuro dei giovani.

1.1 Perché l'abbandono universitario è un problema

La questione dell'abbandono universitario merita attenzione per diverse ragioni; una di queste è che gli investimenti sul capitale umano apportano diversi benefici (Marco Delogu [2023]).

In primo luogo, un livello di istruzione elevato del capitale umano aumenta la produttività degli individui, rendendoli più preziosi sia per i datori di lavoro sia per i lavoratori, poiché aumenta il loro potenziale guadagno. In secondo luogo, l'istruzione migliora la capacità degli individui di adattarsi alle mutevoli condizioni economiche, grazie all'acquisizione di nuove competenze e conoscenze. Bisogna considerare, anche, che i posti di lavoro più ambiti richiedono delle qualifiche sempre più avanzate; per questo è fondamentale avere personale istruito come forza lavoro.

Inoltre, la domanda di una forza lavoro istruita è strettamente connessa al repentino e significativo cambiamento tecnologico degli ultimi anni. Quest'evoluzione ha portato notevoli vantaggi in termini di crescita economica e di ottimizzazione dell'efficienza; tuttavia spesso, l'adozione di tali tecnologie è molto difficile per le aziende proprio a causa dell'assenza di manodopera specializzata. Lo dimostra una ricerca effettuata della BCE sulle principali aziende dell'Eurozona focalizzate sulla digitalizzazione; in tale ricerca si è visto che il reclutamento e il mantenimento di personale ICT altamente qualificato è tra i principali ostacoli per l'adozione di tecnologie digitali. Questo avviene a causa del grande divario tra domanda e offerta di competenze in questo ambito.

Infine, l'interruzione prematura del percorso universitario non solo implica una perdita di tempo e di risorse per gli studenti e le loro famiglie, ma comporta anche una distribu-

zione inefficace dei finanziamenti pubblici, dato che l'istruzione riceve spesso un sostegno finanziario da parte dello Stato.

Nonostante questi benefici evidenti legati al conseguimento di una laurea, la percentuale di studenti che abbandonano gli studi universitari rimane sorprendentemente elevata, richiedendo l'implementazione di nuove politiche e strategie per affrontare questa sfida educativa.

1.2 Situazione in Italia

Tra i Paesi OCSE, l'Italia è, senza dubbio, uno dei paesi sviluppati con i numeri più preoccupanti. Sebbene ci siano stati alcuni miglioramenti negli ultimi anni, l'Italia è ancora uno dei paesi con il tasso di abbandono più alto tra gli studenti universitari, nonostante le numerose riforme attuate negli anni per aumentare il numero dei laureati.

Nel contesto del sistema italiano, l'abbandono universitario rimane un fenomeno predominante. Secondo il rapporto ANVUR [2023], fino all'Anno Accademico 2019/20 si è osservato un notevole miglioramento nel tasso di abbandono tra il primo e il secondo anno per tutte le tipologie di corsi. Tuttavia, a partire dall'Anno Accademico 2020/21, si è registrato un significativo aumento di questo dato, invertendo la tendenza positiva degli anni precedenti, come si può notare dalla Figura 1.1.

È opportuno notare che, per tutti i livelli di corso di studi, la percentuale di abbandoni si avvicina al dato dell'Anno Accademico 2011/12. Le lauree triennali, con il 14,5%, sono molto vicine al 15,1% dell'Anno Accademico 2011/12; analogamente per le lauree biennali, dove si ha una percentuale di abbandoni pari al 7,3%, in linea col dato di dieci anni prima.

Relativamente ai corsi di laurea a ciclo unico, il dato di abbandoni è dell'8,2%, riportando la situazione a quella registrata nell'Anno Accademico 2012/13.

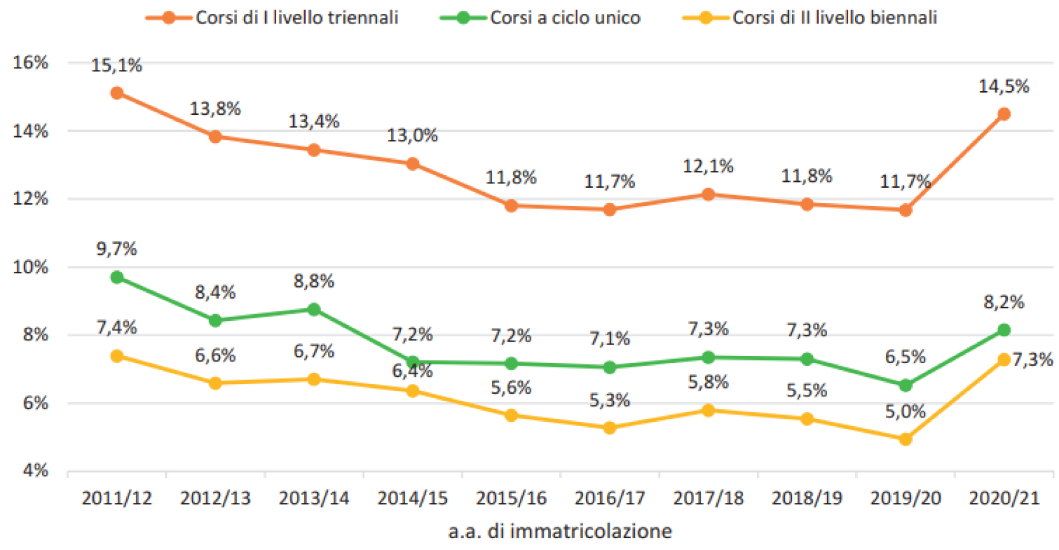


Figura 1.1: Andamento degli abbandoni in Italia tra il I e il II anno

Approfondendo l'analisi dei dati, come evidenziato nella Figura 1.2, emerge un persistente fenomeno di abbandono degli studi universitari anche nei periodi successivi al primo anno. In particolare, considerando i corsi di laurea triennale e focalizzandoci sulla coorte degli studenti immatricolati nell'Anno Accademico 2017/18, si osserva che il 20,1% di essi ha interrotto il percorso dopo tre anni dall'iscrizione. Tale percentuale aumenta leggermente al 22,6% per

coloro che hanno abbandonato il corso dopo quattro anni. È importante sottolineare che per "coorte" si intende l'anno di ingresso dello studente nel corso di studi. Ad esempio, se uno studente si iscrive al primo anno nell'Anno Accademico 2022-2023, la coorte di riferimento sarà il 2022.

Proseguendo nell'analisi dei tassi di abbandono, si nota che il 24,2% degli studenti appartenenti alla coorte iscritta nell'Anno Accademico 2015/16 ha interrotto il percorso universitario entro sei anni dall'immatricolazione.

Di conseguenza, si può affermare che, entro il periodo di durata standard del corso, approssimativamente uno studente su cinque abbandona gli studi universitari. Questo dato aumenta ad uno studente su quattro entro il periodo di sei anni dall'inizio del percorso accademico.

Quelli analizzati sono dati preoccupanti, e dimostrano come il fenomeno degli abbandoni non sia limitato solo al passaggio tra il primo e il secondo anno, ma è molto presente anche negli anni successivi. È essenziale concentrare l'attenzione non solo sull'orientamento iniziale, ma anche sulle politiche e sulle iniziative di tutorato lungo l'intero percorso accademico. Questo approccio è cruciale per contrastare l'abbandono universitario e supportare gli studenti lungo l'intero ciclo di studi.

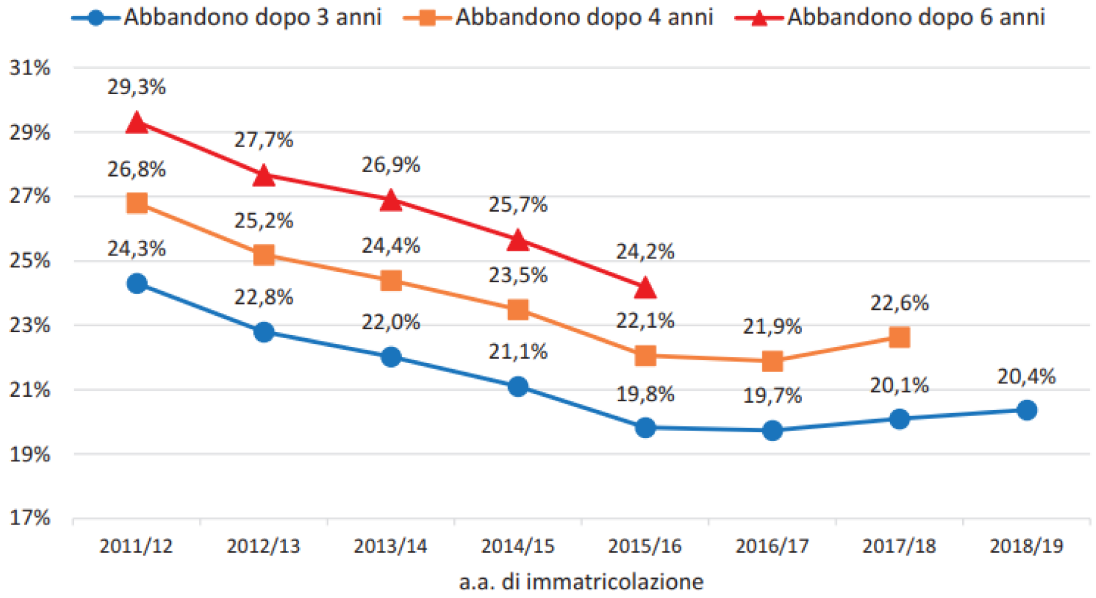


Figura 1.2: Andamento degli abbandoni in Italia dal III anno

1.3 Cause dell'abbandono

In questo paragrafo, si esamineranno le principali cause associate all'interruzione degli studi universitari.

Uno studio condotto da Filippo Belloc e Petrella [2010] ha investigato le ragioni dell'abbandono universitario in Italia, concentrandosi sul background formativo degli studenti, sul loro rendimento accademico e sulle caratteristiche personali. Secondo questo studio, l'abbandono spesso si correla a un atteggiamento "orientato al consumo" da parte degli studenti: molti abbandonano l'università non appena si rendono conto di non apprezzare il percorso di studio scelto. Pertanto, una scelta iniziale errata dell'istituto o del programma accademico aumenta notevolmente il rischio di abbandono.

Secondo Becker [2001], è comune che gli studenti si iscrivano all'università poiché incontrano difficoltà nel trovare lavoro. Di conseguenza, quando ricevono un'offerta occupazionale, tendono a interrompere gli studi universitari.

Un ulteriore studio condotto da Lara Gittoa [2016] ha evidenziato che il tasso di abbandono degli studenti non è soltanto influenzato dal loro background, ma spesso è legato anche alla distanza tra la residenza dello studente e l'università.

L'esistenza di un campus universitario nelle vicinanze può spingere alcuni studenti a iscriversi, dato che potrebbero non averlo fatto se l'università fosse stata situata lontano dalla loro città natale. Questi studenti potrebbero risultare meno motivati e incontrare maggiori difficoltà nel conseguire crediti universitari rispetto a coloro che frequentano istituti distanti (affrontando, dunque, spese più elevate).

Una struttura universitaria più concentrata, focalizzata attorno a un'unità centrale, potrebbe offrire un ambiente accademico più efficace per gli studenti, contribuendo, così, a ridurre i tassi di abbandono.

Secondo le indagini condotte da Burgalassi [2016], gli studenti con un'età più avanzata mostrano una maggiore tendenza all'abbandono universitario, spesso legata alla presenza di responsabilità familiari come coniuge e figli, o all'impegno in un lavoro a tempo pieno.

Per quanto riguarda l'approccio all'esperienza universitaria, l'abbandono precoce è fortemente correlato a una scarsa partecipazione alle attività accademiche. Questo si osserva particolarmente tra coloro che interrompono il corso durante il primo anno, i quali spesso non hanno mai frequentato regolarmente le lezioni programmate o hanno partecipato solo nelle prime settimane del semestre.

Un'altra causa di abbandono va individuata nelle sfide che l'ambiente accademico presenta ai nuovi studenti, le quali influenzano significativamente un segmento degli iscritti: molti attribuiscono questa decisione a difficoltà di adattamento legate all'organizzazione dell'ambiente universitario e alle relazioni con i coetanei e i docenti.

CAPITOLO 2

Acquisizione dei dati per la campagna

Questo capitolo approfondisce la fase iniziale del progetto, incentrata sull'acquisizione dei dati della campagna, offrendo un'analisi dettagliata del processo di raccolta dati che costituiranno la base per le operazioni di ETL nella fase successiva.

Il primo paragrafo delinea i dati che devono essere acquisiti, le loro fonti e il relativo contenuto.

Viene stabilito l'obiettivo primario, che è quello di raccogliere dati pertinenti e accurati, fondamentali per la creazione del Data Warehouse che sarà utilizzato nelle fasi successive del progetto.

Infine, vengono illustrati i metodi di estrazione dei dati impiegati per raggiungere tale obiettivo.

2.1 Le fonti

Per condurre questa analisi sono stati acquisiti dati provenienti da due fonti distinte. La prima consiste in un database esterno fornito dall'Università di Pisa, mentre la seconda è costituita dal sito dell'Osservatorio CINECA.

Nel dettaglio, dal database dell'Università di Pisa sono stati prese le seguenti tabelle:

- ANS1: contenente gli ingressi; questi rappresentano il numero di nuovi studenti che iniziano il loro percorso accademico nell'università, iscrivendosi a programmi di laurea triennale, magistrale, dottorato, o altri corsi di studio.
- ANS2: questa tabella contiene l'insieme dei titoli universitari.
- ANS3: è la tabella contenente i debiti e i crediti degli studenti iscritti all'università.
- ANS4: contiene tutti gli eventi effettuati durante la carriera degli studenti iscritti. Tali eventi sono: variazione di sede, iscrizioni e ricognizioni, variazioni di corso, trasferimenti in uscita e trasferimenti in ingresso.
- ANS5: è la tabella contenente le variazioni amministrative. Ad esempio "Periodi di mobilità studenti incoming", cioè quegli studenti che arrivano da altre istituzioni o altri paesi per studiare o svolgere attività accademiche, oppure "Periodo di specifica attività di ricerca", cioè un ricercatore, un dottorando o uno studente che svolge attività di ricerca specifiche, ad esempio per un progetto di ricerca, la scrittura di una tesi, lo svolgimento di esperimenti o lo studio approfondito di un argomento specifico.
- ANS6: contiene tutti gli esami effettuati da tutti gli studenti iscritti, con i relativi CFU e voti conseguiti.

- ANS7: contiene tutti gli eventi riguardanti la chiusura della carriera degli studenti; questa può essere chiusa a seguito di una laurea, di un ritiro o per altri motivi.
- CORSI: contiene l'anagrafica dei corsi di studi con tutte le informazioni di carattere interno.
- ESAMI: contiene l'anagrafica degli esami/insegnamenti o attività che danno seguito a debiti o crediti.
- STUDENTI: contiene le informazioni relative a tutti gli studenti che si sono iscritti all'università, come nome, cognome, codice fiscale, data di nascita, comune di residenza etc.

Dal sito dell'Osservatorio del CINECA, invece, vengono scaricati i dati che servono per formare le diverse anagrafiche, come:

- COMUNE: contiene l'anagrafica della geografia italiana con livello di dettaglio comunale. Contiene, anche, le informazioni gerarchiche di Provincia e Regione.
- DENOMINAZIONE DIPLOMI: contiene le anagrafiche delle tipologie di diploma.
- CORSI DI LAUREA: contiene l'anagrafica delle classi di laurea.
- UNIVERSITÀ: contiene l'anagrafica degli atenei universitari.
- NAZIONI: contiene l'anagrafica delle nazioni.

2.2 Obbiettivo

L'obbiettivo, in questa fase del progetto, è di estrarre regolarmente i dati dall'Università di Pisa e dal sito dell'Osservatorio CINECA, garantendo un aggiornamento costante per evitare che diventino obsoleti, e quindi non affidabili.

I file menzionati nel paragrafo precedente risultano fondamentali per il progresso del progetto, poiché costituiranno la base per l'attività di previsione degli abbandoni, ed è indispensabile che questi dati siano costantemente aggiornati.

Come evidenziato nella Figura 2.1, il processo inizia con l'estrazione dei file dal sito dell'Osservatorio CINECA e con l'acquisizione dei dati dal Data Warehouse dall'Università di Pisa. I dati saranno automaticamente caricati sulla piattaforma Cloud di Azure.

Successivamente, verrà eseguita la fase di ETL (Extract, Transform, Load) tramite Data Factory; questo processo sarà dettagliato nel capitolo successivo.

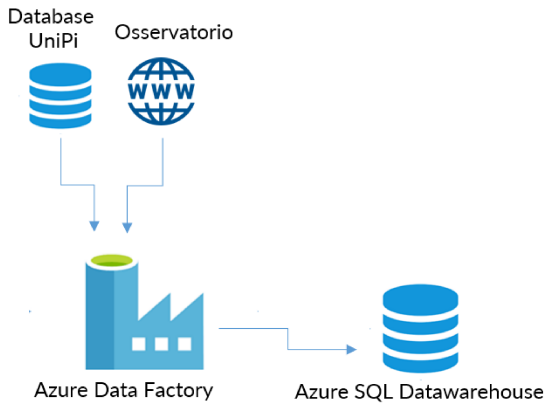


Figura 2.1: Flusso dei dati

Infine, i dati costituiranno il dataset fondamentale per l'analisi predittiva degli abbandoni. Il Data Warehouse (DWH) è stato modellato identificando le entità sotto forma di tabelle dei fatti e delle dimensioni.

Di seguito, sono presentati i modelli logici dello schema di destinazione. In particolare, la Figura 2.2 illustra l'analisi della carriera derivata dai dati raccolti in questa fase. Le dimensioni e i fatti ottenuti dai dati che si otterranno in questa fase sono rappresentati in modo grafico, offrendo una visione dettagliata delle correlazioni e delle principali caratteristiche presenti nei dati.

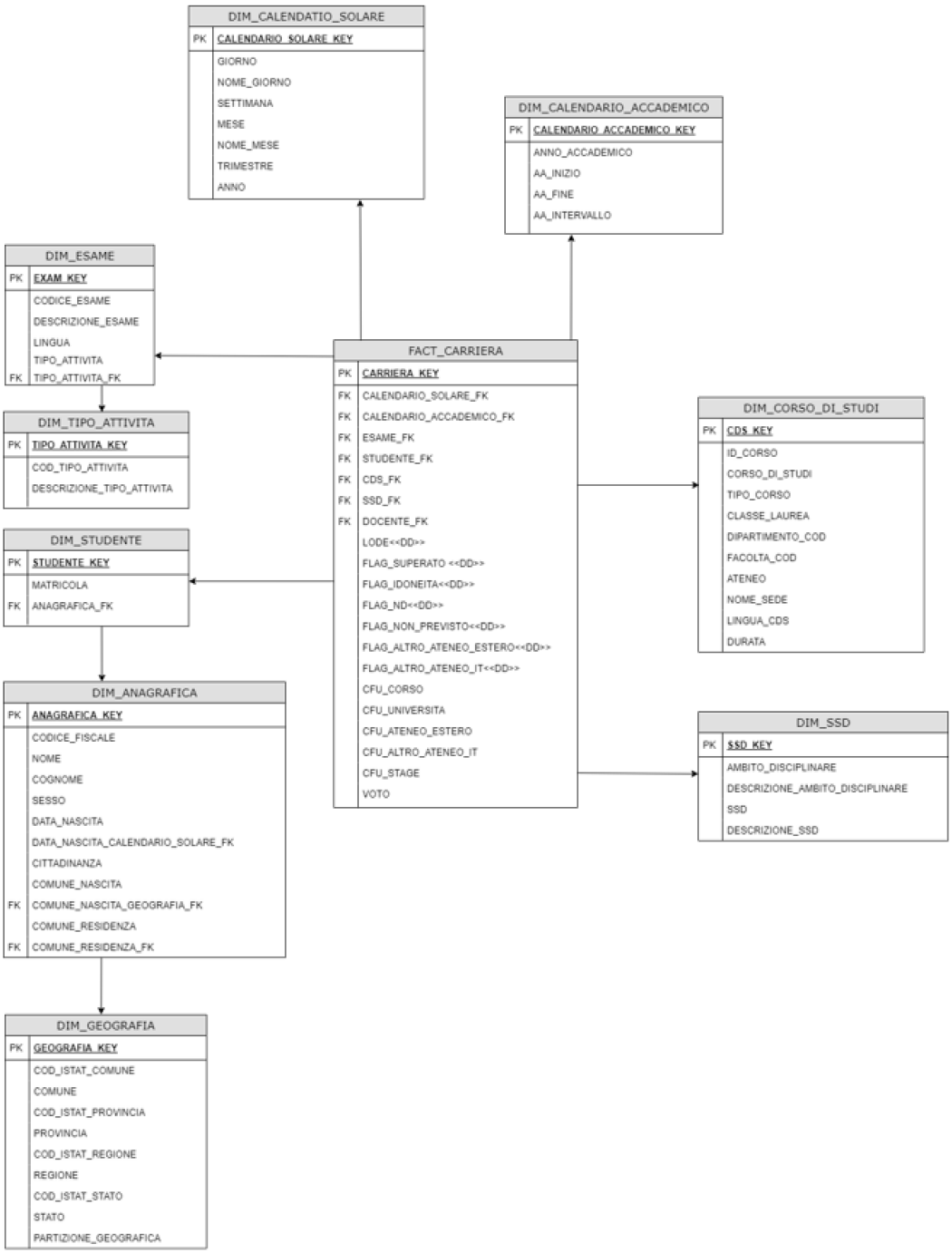


Figura 2.2: Schema logico della carriera di uno studente

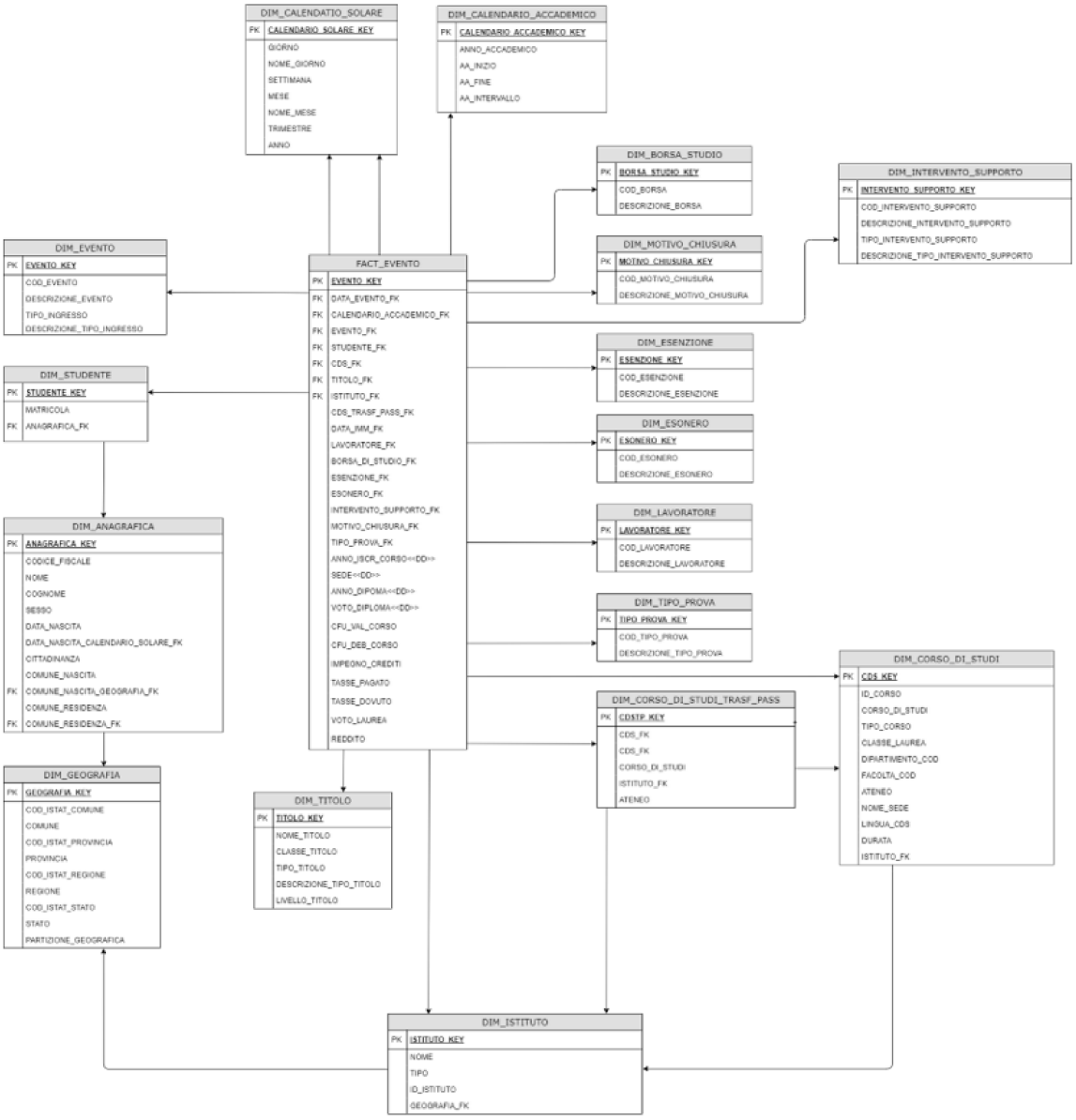


Figura 2.3: Schema logico di evento

Nella Figura 2.3 viene rappresentato il fatto "Evento", con le relative dimensioni ad esso connesse; anche queste informazioni sono state ricavate dai file e dalle tabelle scaricati.

Per conseguire tali obiettivi, è stato implementato uno script usando il linguaggio di programmazione Python per il download dei file dal sito dell'Osservatorio CINECA. Dall'altra parte, per acquisire i dati dalle tabelle dal Data Warehouse dell'Università di Pisa, è stata implementata una pipeline all'interno di Data Factory.

2.3 Implementazione delle soluzioni

Come visto nelle sezioni precedenti, bisogna estrarre i dati da due fonti diverse.

Per prima cosa viene illustrato come effettuare l'attività di scraping dal sito dell'Osservatorio CINECA. Per questa attività è stato realizzato un script in Python, di cui viene riportato il codice in seguito.

```
from bs4 import BeautifulSoup
import requests
import logging
import os
import json
from zipfile import ZipFile
from azure.storage.blob import BlobServiceClient
from azure.storage.blob import BlobClient


url_ingresso = "https://osservatorio.cineca.it/php5/osd_sito_load.php"

username = "aaaaa.bbbbb"
password = "*******"
ateneo = "24"

payload = {
   "code_un": ateneo,
   "username": username,
   "password": password,
   "first": "1",
   "submit": "E N T R A" }
```

In questa prima parte di codice sono state importate tutte le librerie necessarie per eseguire lo script:

- `from bs4 import BeautifulSoup`: importa la classe `BeautifulSoup` dalla libreria `bs4`, che è comunemente utilizzata per analizzare ed estrarre dati da pagine HTML e XML.
- `import requests`: importa il modulo `requests`, che permette di inviare richieste HTTP facilmente. Tale modulo è usato per scaricare il contenuto di pagine web.
- `import logging`: importa il modulo `logging` di Python, che consente di registrare messaggi informativi, di debug, avvertenze, errori, etc., durante l'esecuzione del programma.
- `import os`: importa il modulo `os` che fornisce funzioni per l'interazione con il sistema operativo, come operazioni di gestione dei file, manipolazione del percorso, variabili di ambiente, etc.
- `import json`: importa il modulo per la manipolazione dei dati in formato JSON, che offre funzioni per convertire oggetti Python in JSON, e viceversa.
- `from zipfile import ZipFile`: importa la classe `ZipFile` dal modulo `zipfile` che permette di lavorare con file `.zip`; questa sarà necessaria perché i file verranno scaricati in formato `.zip` e sarà necessario estrarli primi di caricarli sul Cloud Azure.
- `from azure.storage.blob import BlobServiceClient`: importa la classe `BlobServiceClient` dal modulo `azure.storage.blob`, che consente l'interazione con il servizio di archiviazione di Azure per gestire contenuti all'interno di contenitori Blob, dove veranno caricati i file.
- `from azure.storage.blob import BlobClient`: importa la classe `BlobClient` dal modulo `azure.storage.blob`, che consente di interagire con un singolo blob all'interno del servizio di archiviazione di Azure.

La seconda parte del programma Python serve per effettuare l'accesso al sito dell'Osservatorio CINECA, ossia quello indicato in `url_ingresso`.

Sono state definite le variabili `username`, `password` e `ateneo` (24 è il codice ateneo che identifica l'Univerità di Pisa).

Infine, è stato definito il `payload`; questo è un dizionario Python contenente i dati da inviare al server.

I campi del `payload` sono tipicamente quelli richiesti dal sistema per autenticare l'utente. Nel caso specifico, ci sono i campi: `code_un`, `username`, `password`, `first`, e `submit`. Questi campi rappresentano i dettagli dell'utente e l'azione da intraprendere, in questo caso un login.

```
s = requests.session()
response = s.post(url_ingresso , data=payload)
print(response.status_code)
soup = BeautifulSoup(response.content, "html.parser")
req =
s.get('https://osservatorio.cineca.it/osd.php?sez=ADWN&pag=DGEN&liv=0')
print(req.status_code)
soup2 = BeautifulSoup(req.content, "html.parser")
estrazione = soup2.find_all('a')

for linko in estrazione:
   if linko.get("href").find("CODICI_GENERALI") > 0:
   url = 'https://osservatorio.cineca.it'+linko.get("href").strip()
url = 'https://osservatorio.cineca.it'+estrazione[40].get("href").strip()
local_filename = 'CODICI_GENERALI.zip'
r = s.get(url)
open(local_filename, 'wb').write(r.content)
```

Nel pezzo di codice Python riportato in precedenza vengono effettuate una serie di operazioni HTTP, dopo aver effettuato l'autenticazione.

Per prima cosa è stata creata una sessione HTTP utilizzando il modulo `requests`; questo permette di mantenere la persistenza dei dati della sessione, come le informazioni di autenticazione, tra diverse richieste HTTP.

Successivamente è stata effettuata una richiesta POST all'URL di accesso definito nella prima parte del codice visto in precedenza (`url_ingresso`) con i dati del payload (che contengono le credenziali di accesso).

La risposta dal server viene memorizzata nella variabile `response`. Tale risposta verrà analizzata con `BeautifulSoup`.

Deve essere effettuata una seconda richiesta GET per arrivare ad una pagina intermedia da cui scaricare i tag contenuti in essa.

All'interno di quest'ultima, infatti, vengono estratti tutti i link e, tra questi, si cerca quello che contiene il testo `"CODICI_GENERALI"` nell'URL.

Una volta individuato, viene restituito il codice URL completo che è quello da cui scaricare il file.

Infine, viene effettuata l'ultima richiesta GET all'URL restituito in precedenza e viene aperto il file locale in modalità binaria dove viene scritto il contenuto ricevuto dalla richiesta GET, ovvero il file `CODICI_GENERALI.zip`.

All'interno di questo file zip sono contenuti tutti i file che servono per formare le anagrafiche viste nella Sezione 2.1.

```
try:
   try:
      os.mkdir(os.getcwd()+'/files')
   except:
```

```
print('error creating directory: '+os.getcwd()+'/files')

try:
os.mkdir(os.getcwd()+'/files/utf')
except:
   print('error creating directory: '+os.getcwd()+'/files/utf')

with ZipFile(local_filename, 'r') as zip:
zip.extractall('./files')

for root, dirs, files in os.walk(os.getcwd()+'/files'):
for f in files:
   if os.path.splitext(f)[1] == '.dat':filepath = root + os.sep + f
         fileout = root + os.sep + 'utf/' + os.path.splitext(f)[0]+ '.txt'
         fileb = open(filepath, 'rb').read()
         try:
            filed=fileb.decode('utf-8')
         except:
            filed=fileb.decode('iso-8859-1')
         with open(fileout,'w', encoding='utf-8') as fout:
            fout.write(filed)
```

La parte di codice sopra riportata serve, essenzialmente, per effettuare l'estrazione dei file `.zip` scaricati al passo precedente.

Nel dettaglio si creano due cartelle nelle quali verranno salvati i file. Viene creata una prima cartella `files` e al suo interno un'altra cartella `utf`. Se ci sono errori viene stampato un messaggio.

Si procede con l'estrazione dei file `.zip`; in particolare viene usato `ZipFile` per estrarre il contenuto del file `.zip` scaricato al passo precedente; questi file vengono estratti nella cartella `files`.

Per ogni file `.dat` presente nella cartella `files`, il codice legge il contenuto binario del file e tenta prima di decodificarlo come UTF-8. Se la decodifica fallisce, prova a decodificarlo come ISO-8859-1.

Il contenuto del file decodificato viene, infine, salvato come file di testo con estensione `.txt` nella sottocartella `utf`.

```
service = BlobServiceClient(account_url="https://acsunipi.blob.
core.windows.net",credential="***")
print('Creato BlockBlobService ')
connection_string="DefaultEndpointsProtocol=https;
AccountName=acsunipi;AccountKey=*****"
service = BlobServiceClient.from_connection_string(conn_str=connection_string)
print('Connection string')
```

In questa fase si crea un'istanza di `BlobServiceClient` dal modulo `azure.storage.blob` per interagire con un account di archiviazione di Azure Blob.

```
for root, dirs, files in os.walk(os.getcwd() + os.sep + "files/utf/"):
   for f in files:
      if os.path.splitext(f)[1] == '.txt':
         blob =
         BlobClient.from_connection_string(conn_str=connection_string,
         container_name="unzippedconverted",
         blob_name=f)
         filepath = os.path.join(root, f)
      with open(filepath, "rb") as data:blob.upload_blob(data, overwrite=True)
      print('Creato Blob '+f)
      print('SUCCESS')
```

```
except Exception as e:
print(str(e))
```

In quest'ultima parte dello script per effettuare lo scraping vengono caricati prima tutti i file con estensione `.txt` nell'archivio Azure Blob.

Per ciascun file `.txt` viene creato un oggetto BlobClient utilizzando la stringa di connessione specificata, dove vengono indicati il contenitore di destinazione (`container_name`) e il nome del blob (`blob_name`).

In seguito, vengono aperti i file `.txt` in modalità binaria (`rb`) e si usa `upload_blob` di `BlobClient` per caricare il contenuto del file nell'archivio di Azure Blob; `overwrite=True` indica che, se il file è già presente con lo stesso nome, deve essere sovrascritto.

Dopo aver completato il caricamento di tutti i file `.txt`, viene stampato un messaggio che indica il completamento del caricamento dei file e il successo dell'operazione.

In sintesi, questo pezzo di codice carica i file `.txt` presenti nella cartella `files/utf/` nell'archivio Azure Blob specificato.

Una volta realizzato lo script, bisogna impostare la sua esecuzione automaticamente. Per questo passaggio è stato usato *MobaXterm*. Essa è un'applicazione che offre un terminale avanzato per Windows con molte funzionalità utili per gli sviluppatori e gli amministratori di sistema, inclusi strumenti per l'esecuzione di comandi su server remoti tramite SSH e per la gestione dei file.

MobaXterm semplifica l'interazione con server remoti grazie alla sua interfaccia utente e alle funzionalità integrate, consentendo di eseguire operazioni quali l'esecuzione di script e la gestione l'output senza dover utilizzare direttamente il terminale di Linux.

Per prima cosa è stata semplicemente creata la cartella in cui salvare lo script con il seguente comando

```
mk dir scraper_new
```

Spostandosi nella cartella appena creata bisogna inserire lo script che verrà chiamato con il nome di `scraper_new`

```
cd scraper_new
```

```
vi scraper_new.py
```

Nella schermata che apparirà viene inserito il codice dello script per effettuare web scraping. Bisogna, poi, creare il file che verrà usato per eseguire lo script. A tal fine, si apre nell'editor "vi" il file `lancia_scraper.sh`

```
vi lancia_scraper.sh
```

All'interno di questo file bisogna inserire il codice che eseguirà l'intero script. Bisogna, quindi, prima indicare la directory in cui il file da eseguire si trova, poi viene attivato l'ambiente virtuale Python ed, infine, viene eseguito il file `scraper_new.py` utilizzando l'interprete Python 3.9.

```
cd /home/ups/scraper_new/
source /home/ups/scraper_new/bin/activate
python3.9 scraper_new.py
```

L'ultimo passo è stabilire quando deve essere eseguito lo script, in questo caso si è deciso di farlo eseguire giornalmente alle ore 04:00 del mattino.

Per eseguire questa operazione è stato usato prima`crontab`, per aprire il file `crontab` dell'utente corrente in un editor di testo.

`Crontab` è un file di configurazione che contiene le istruzioni per l'esecuzione periodica di comandi o script.

```
crontab -e
```

L'ultimo passo è quello di configurare il cron job. Questa riga dice al sistema di eseguire il comando `/home/upvs/lancia_sraper.sh` ogni giorno alle 4:00 AM. Inoltre, reindirizza l'output (standard output e standard error) del comando a un file di log chiamato `lancia_scraper.log` nella directory `/home/upvs`

Riassumendo, questa configurazione crontab attiva lo script `lancia_scraper.sh` ogni giorno alle 4 del mattino e salva l'output (compresi eventuali messaggi di errore) in un file di log.

```
00 04 * * * /home/upvs/lancia_sraper.sh > /home/upvs/lancia_scraper.log 2>&1
```

L'altra fase è quella di estrazione dei dati dal database dell'Università di Pisa.

In questo caso il sistema utilizza un Database SQL Server di tipo Express, che è gratuito, come supporto per i dati, mentre il sistema sottostante è ESSE3 eseguito su Oracle.

Per evitare di concedere accesso al sistema principale, vengono duplicate le tabelle necessarie da Oracle a SQL Server.

Per acquisire i dati da questo database Express è stata creta una pipeline in Data Factory.

Microsoft Data Factory è un servizio di integrazione dei dati basato su cloud fornito da Microsoft all'interno della suite di servizi di Azure.

È progettato per automatizzare, orchestrare e gestire il movimento, la trasformazione e l'elaborazione di dati da diverse fonti a diverse destinazioni.

Le sue principali funzioni sono:

- *Integrazione dei dati*: consente di connettere e integrare dati da varie sorgenti, come database locali, dati cloud, servizi Software as a Service (SaaS), e altro ancora.
- *ETL (Extract, Transform, Load)*: consente di estrarre dati da diverse sorgenti, trasformarli in formati utilizzabili e caricarli in un repository o un data lake.
- *Automazione dei flussi di lavoro*: permette la creazione di flussi di lavoro (detti, anche, "pipeline"), che orchestrano e automatizzano il movimento e la trasformazione dei dati in un determinato ordine.
- *Monitoraggio e gestione*: offre strumenti per il monitoraggio delle attività, il controllo dei flussi di lavoro e la gestione delle risorse utilizzate per l'elaborazione dei dati.

In questa fase è usata la funzione di automatizzazione dei flussi attraverso l'uso di una pipeline per permettere di integrare i dati provenienti da una fonte esterna.

Per importare i dati dal database dell'Università di Pisa è stata creata una pipeline a cui è stato dato il nome di `LOAD_INGESTION_DB`. Questa pipeline, non fa altro che prendere i dati dal Database SQL Server di tipo Express dell'Università e copiarli nel database su cui, in un secondo momento, verranno effettuate le modifiche per ottenere gli schemi dei fatti e delle dimensioni visti nelle Figure 2.2 e 2.3. La Figura 2.4 riporta il contenuto della pipeline `LOAD_INGESTION_DB`; questa è composta da una serie di "execute pipeline", una per ogni tabella vista nella Sezione 2.1, che deve essere importata.

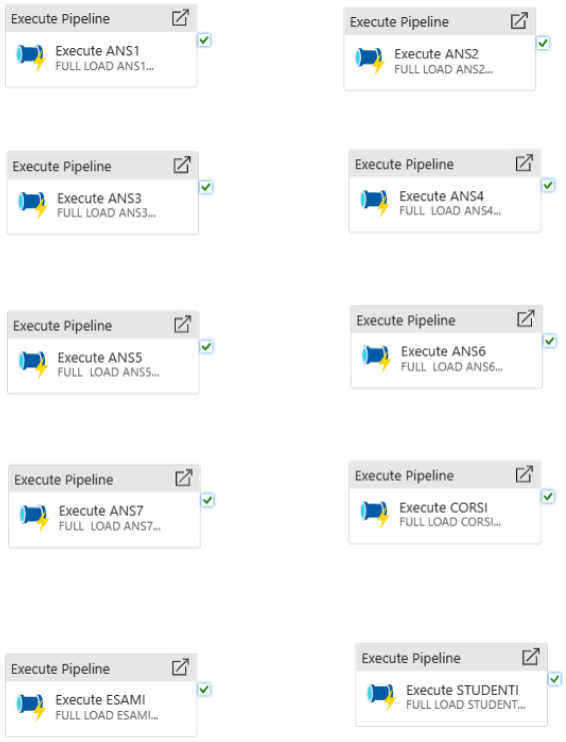


Figura 2.4: Pipeline LOAD INGESTION DB

Le execute pipeline contengono al loro interno l'attività di "copy data"; tale attività permette di trasferire i dati dal database dell'Università di Pisa a quello su cui verranno effettuate le analisi.

Le execute pipeline non sono collegate tra loro; questo significa che vengono eseguite in parallelo.

CAPITOLO 3

Attività di Extraction, Transformation and Loading

In questo capitolo viene trattata tutta la fase di ETL (Extraction, Trasformation and Loadig) eseguita prevalentemente con Data Factory.

Per l'esecuzione di questa fase viene definita una pipeline principale, denominata `Main`*, all'interno della quale sono comprese diverse sotto-pipeline che vanno a coordinare il processo di ETL. All'interno della pipeline* `Main` *sono presenti: la pipeline* `Load Ingestion` *per l'estrazione dei dati,* `Cleaning` *usata per effettuare diverse trasformazioni sui dati,* `Dim` *e* `Fact` *per la trasformazione e il caricamento delle tabelle dei fatti e delle dimensioni.*

Infine, grazie all'aiuto di una procedura in SQL Server, viene creato il dataset finale da utilizzare per effettuare le previsioni.

3.1 Data Factory

Dopo aver acquisito i dati durante la fase precedente, in questo capitolo viene affrontata la fase di ETL del progetto, cioè di estrazione, trasformazione e caricamento dei dati.

L'intera fase è stata effettutata con un software parte dei Microsoft Azure Service; questi ultimi mettono a disposizione un'ampia gamma di servizi per acquisire, memorizzare e trasformare i dati di diverso tipo e di grandi dimensioni. Tra i servizi messi a disposizione rientra Data Factory.

Data Factory rappresenta un servizio basato su cloud progettato per agevolare le operazioni di ETL e l'integrazione di informazioni. Esso consente di creare flussi di lavoro orientati ai dati orchestrando con precisione il trasferimento e la trasformazione di informazioni su vasta scala. Esso offre un'interfaccia utente senza codice per la creazione intuitiva di contenuti e una singola console per il monitoraggio e la gestione.

Azure Data Factory è costituito dai seguenti componenti chiave:

- *Pipeline*: è un raggruppamento logico di attività che esegue un'unità di lavoro. L'insieme delle attività di una pipeline esegue un'operazione. Essa offre il vantaggio di poter gestire le attività come un set, anziché singolarmente; Le attività possono essere eseguite in parallelo o in modo sequenziale; in quest'ultimo caso le pipeline devono essere concatenate.
- *Attività*: le attività rappresentano un passaggio di elaborazione in una pipeline. Ad esempio, è possibile usare un'attività di copia per copiare i dati da un archivio dati all'altro.

- *Set di dati*: rappresentano strutture dati all'interno degli archivi. Fanno semplicemente riferimento ai dati da usare nelle attività come input o output.
- *Linked service*: costituisce la configurazione di connessione a una specifica sorgente di dati. Ad esempio, consideriamo un'applicazione di Data Factory che necessita di interagire con un database SQL Server; in questo contesto, il servizio collegato rappresenterebbe la definizione delle credenziali di accesso, l'URL del server e altri parametri necessari per stabilire e gestire la connessione al database SQL Server.
- *Flussi di dati*: rappresentano la definizione logica delle operazioni di trasformazione dei dati all'interno del servizio. Un flusso di dati definisce come questi ultimi vengono estratti, trasformati e caricati (ETL) all'interno delle pipeline.
- *Runtime di integrazione*: in Data Factory, un'attività definisce l'azione da eseguire, un servizio collegato definisce un archivio di dati o un servizio di calcolo di destinazione, mentre un runtime di integrazione funge da ponte tra l'attività e i servizi collegati.

Questi sono i componenti usati dalla piattaforma di Azure Data Factory (ADF) nella quale è possibile comporre flussi di lavoro basati sui dati con passaggi per lo spostamento e la trasformazione dei dati stessi.

Le attività usate in questo progetto per effettuare la fase di ETL sono:

- *Copy data*: questa attività permette di copiare dati da una vasta gamma di sorgenti e di caricarli in diverse destinazioni, consentendo di spostare dati tra sorgenti e destinazioni diverse, configurare mapping ed effettuare trasformazioni semplici. L'attività permette di mappare i campi tra sorgente e destinazione; questo può includere la selezione di colonne specifiche o la trasformazione dei tipi di dati.
- *Stored procedure*: questa attività consente di chiamare una stored procedure di SQL Server.
- *Execute pipeline*: è un componente che consente di eseguire un'altra pipeline all'interno dello stesso servizio ADF. Essa è utile quando si costruiscono flussi di lavoro complessi che coinvolgono più pipeline e si desidera orchestrarne l'esecuzione.

Le fonti usate in questa fase sono:

- *SQL server*: è un sistema di gestione di database relazionali (RDBMS) sviluppato da Microsoft.
- *Azure Synapse Analytics*: i dati elaborati da una pipeline di dati di ADF sono inviati ad un'area di archiviazione o a una tabella all'interno di Azure Synapse Analytics. In termini più semplici, Synapse Analytics funge da destinazione per i dati che vengono spostati o elaborati all'interno di Azure Data Factory.

L'obbiettivo è quello di partire con i dati acquisiti nella fase precedente dal sito dell'Osservatorio del CINECA e dal database dell'Università per ottenere, alla fine di questa fase, gli schemi a stella riportati nelle Figure 2.2 e 2.3 del capitolo precedente.

3.2 Main

L'intera fase di ETL viene eseguita da un'unica pipeline chiamata `Main` rappresentata in Figura 3.1. Questa pipeline orchestra l'esecuzione di altre pipeline che effettuano operazioni più specifiche.

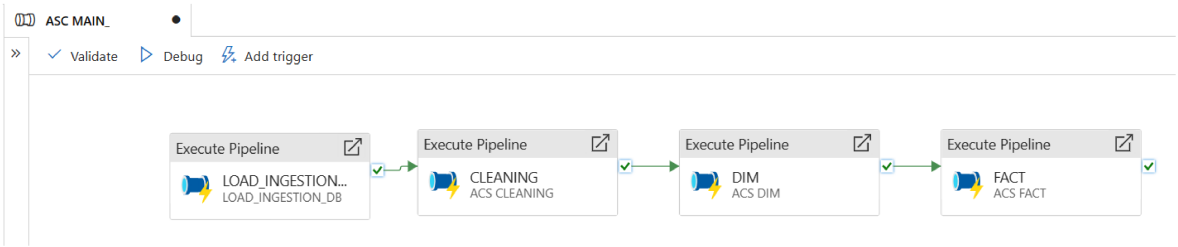


Figura 3.1: Pipeline `Main`

La pipeline `Main`, come primo passo, avvia il caricamento nell'area staging delle tabelle ottenute dal database dell'Università per poi effettuare il caricamento delle tabelle delle dimensioni e dei fatti.

Le attività sono collegate in un certo ordine; questo significa che verranno svolte in sequenza e nell'ordine in cui sono state inserite, non in parallelo, come visto nella Sezione 2.3. Tutte le attività hanno la condizione posta a `on success`, rappresentata dalla freccia verde; ciò significa che l'attività successiva viene eseguita solo se la precedente è andata a buon fine.

Nelle prossime sezioni vengono esaminate tutte le parti della pipeline `Main`.

3.2.1 Load ingestion

La pipeline `LOAD_INGESTION_DB` è essenzialmente quella che è stata vista nella sezione 2.3; essa permette di trasferire i dati dal database dell'Università di Pisa al nostro databese gestito tramite SQL Server.

Poiché i dati in questione sono già su un database e sono già in formato relazionale, per effettuare questo passaggio è bastato effettuare un'attività di copy data, come mostra la Figura 3.2.

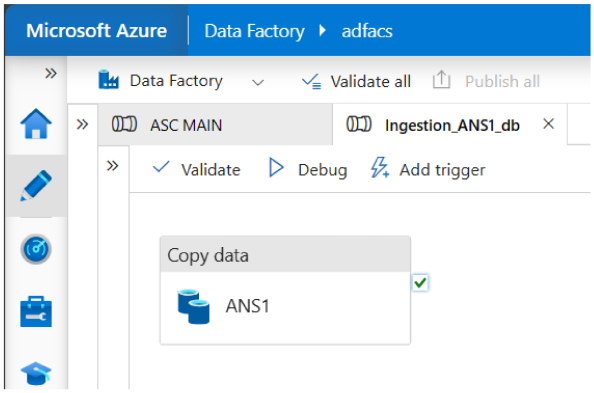


Figura 3.2: Attività di copy data per ANS1

In questa attività è stato impostata come `source` una connessione a SQL Server, corrispondente al database dell'Università di Pisa; questa è la fonte da cui estrarre i dati. Come `sink` è stato impostato un Azure Synapse Analytics; questa è la destinazione su cui vengono salvati i dati. Al termine del processo, è stata eseguita l'operazione di mapping tra la tabella proveniente dalla sorgente e la corrispondente tabella di destinazione.

Queste operazioni sono state effettuate per tutte le tabelle che devono essere importate dal database dell'Università di Pisa e vengono effettuate in parallelo tra loro perché sono indipendenti l'una dall'altra; infatti, in `LOAD_INGESTION_DB`, le execute pipeline non sono collegate tra loro.

Tutte le pipeline in questione sono state sviluppate utilizzando un approccio parametrico. Questo metodo di progettazione consente di migliorare la flessibilità e la configurabilità delle pipeline, facilitando l'adattamento dinamico del comportamento del flusso di lavoro senza richiedere modifiche dirette al codice sorgente.

L'implementazione di parametri offre un meccanismo robusto per la personalizzazione delle operazioni di estrazione e caricamento dei dati, garantendo che la pipeline possa essere adattata a diverse configurazioni e requisiti senza la necessità di riscrivere o modificare manualmente il codice sorgente.

L'adozione di questa strategia risponde alla necessità di un'architettura flessibile e dinamica, fornendo un elevato grado di personalizzazione durante l'esecuzione della pipeline.

Nel caso in cui non si fosse adottato un approccio parametrico, si sarebbe stati costretti a creare sorgenti e destinazioni distinte per ciascuna tabella coinvolta nel processo.

Per quanto riguarda i dati che sono stati scaricati dal sito dell'Osservatorio del CINECA, una volta estratti e salvati nel container, vengono richiamati dalla base di dati utilizzando external tables. In tale caso lo storage container è stato definito come external source nella base di dati e, per ogni file, è stata creata la tabella esterna sullo schema `stg_e`, come mostrato dalla Figura 3.3. Le tabelle esterne rappresentano solo il link ai dati salvati nello storage; lo spostamento reale avviene tramite le stored procedure che verranno esposte nella Sezione 3.2.6.

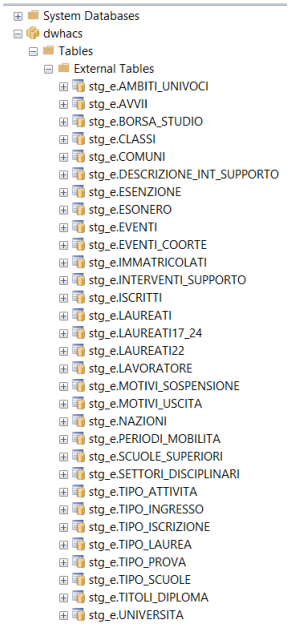


Figura 3.3: Tabelle esterne dedicate ai dati dell'Osservatorio

3.2.2 Cleaning

La pipeline successiva è denominata `CLEANING`; al suo interno è presente solo una stored procedure. Per eseguire la procedura, si usa un linked service dove è riportato il nome specifico della procedura che la pipeline è incaricata di eseguire. La procedura è stata scritta in Transact-SQL (T-SQL), un linguaggio sviluppato da Microsoft e utilizzato principalmente per interagire con i database relazionali di Microsoft SQL Server. Nello specifico, la procedura in questione crea una serie di tabelle intermedie riguardanti gli studenti, che serviranno per effettuare altri passaggi illustrati nella Sezione 3.2.6.

3.2.3 Dimensioni

L'execute pipeline DIM modella le dimensioni del Data Warehouse. Al suo interno sono presenti altre execute pipeline, come mostra la Figura 3.4.

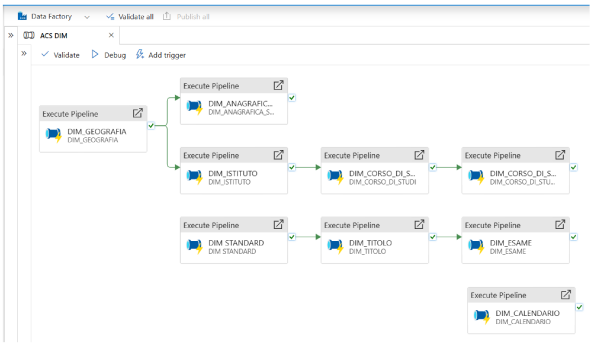


Figura 3.4: Pipeline DIM

Come evidenziato, alcune di tali operazioni devono essere eseguite sequenzialmente, mentre altre possono essere eseguite in parallelo. Questa distinzione dipende dalle dipendenze intrinseche delle operazioni stesse. Ad ogni modo, tutte le operazioni sono interconnesse mediante un vincolo di "on success"; ciò implica che, in caso di errore, durante l'esecuzione di una di esse, le operazioni successive non saranno avviate.

Partendo dal basso, nella execute procedure DIM_CALENDARIO sono presenti due procedure (Figura 3.5).

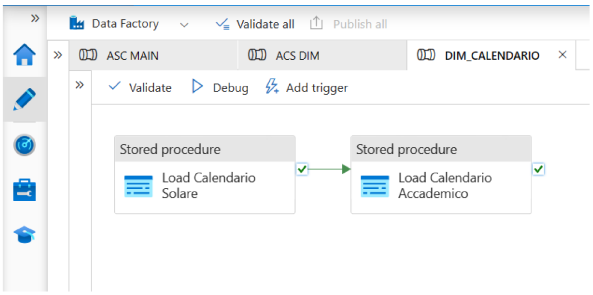


Figura 3.5: Pipeline DIM_CALENDARIO

Le due procedure illustrate nella Figura 3.5 sono quelle che vanno ad inserire i dati nelle rispettive tabelle con informazioni relative all'anno solare e all'Anno Accademico.

Tornando all'execute pipeline DIM, in essa sono presenti tre execute pipeline in serie; queste hanno come obbiettivo finale il popolamento della tabella DIM_ESAME.

Partendo dalla prima execute pipeline, la DIM_STANDARD, al suo interno sono presenti le execute pipeline illustrate in Figura 3.6.

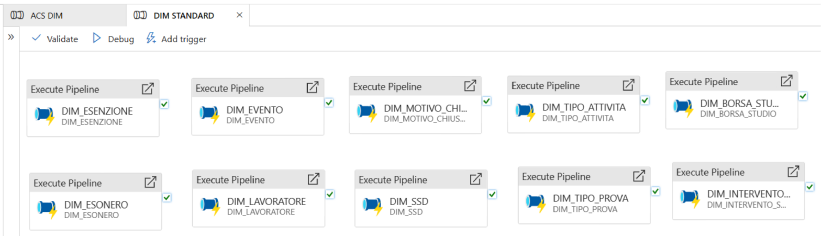


Figura 3.6: Pipeline DIM_STANDARD

Tutte queste execute pipeline al loro interno hanno una struttura simile. È presente una prima procedura che si occupa di estrarre i dati dalle tabelle esterne a quelle di staging. Le tabelle esterne sono quelle viste nella Sezione 3.2.1, ed erano state scaricate dal sito dell'Osservatorio CINECA. La seconda stored procedure effettua delle piccole modifiche alle tabelle di staging create formando le relative tabelle delle dimensioni.

All'interno della pipeline `DIM_TITOLO` vengono eseguite una serie di stored procedure che portano alla formazione della tabella delle dimensioni `DIM_TITOLO` contenente l'insieme dei titoli sia universitari che delle scuole superiori. In questo caso, la particolarità come si può notare dalla Figura 3.7, è che le procedure vengono eseguite tutte in parallelo, e solo se tutte vanno a buon fine può essere creata la tabella `DIM_TITOLO`.

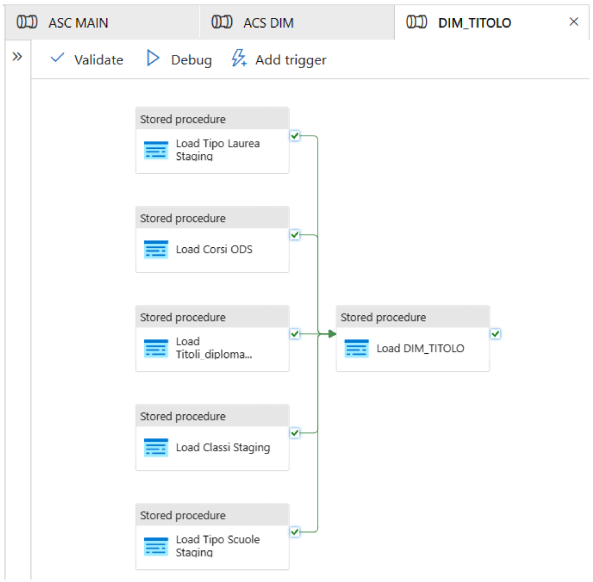


Figura 3.7: Pipeline `DIM_TITOLO`

Infine, la pipeline `DIM_ESAME` richiama al suo interno la stored procedure per popolare/aggiornare il contenuto della relativa tabella.

L'ultima parte della pipeline `DIM` è quella che popola la tabella `DIM_GEOGRAFIA`, eseguendo delle procedure che ricevono i dati dai file scaricati dal sito dell'Osservatorio del CINECA. La `DIM_GEOGRAFIA` serve per inserire i dati nella tabella `DIM_ANAGRAFICA`, contenente tutte le informazioni anagrafiche degli studenti, e nella tabella `DIM_CORSO_DI_STUDI`, contenente tutti i corsi di studi erogati dall'università.

3.2.4 Fatti

L'ultima pipeline che viene eseguita in `Main` è chiamata `FACT`, ha lo scopo di costituire le tabelle dei fatti; essa è illustrata in Figura 3.8.

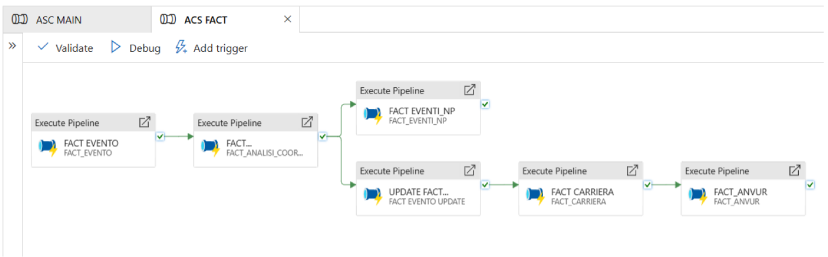


Figura 3.8: Pipeline `FACT`

La prima pipeline esegue al suo interno delle procedure in serie che vanno a popolare la tabella dei fatti `FACT_EVENTO`, rappresentata in Figura 2.3; questa tabella farà anche da base per la creazione dell'altra tabella dei fatti, `FACT_CARRIERA`, vista nella Figura 2.2.

La pipeline `FACT EVENTO UPDATE` consente di aggiornare la tabella `FACT_EVENTO` e di assegnare ai vari eventi (iscrizioni, lauree, etc.) i flag necessari per il calcolo degli indicatori.

Il caricamento della tabella dei fatti `ANALISI_COORTE` prevede l'esecuzione di una sequenza di procedure necessarie per effettuare il calcolo dello stato dello studente.

È presente una tabella `FACT_EVENTO_NP` all'interno della quale sono presenti tutti gli studenti che si sono iscritti a un corso di studi e non hanno effettuato alcuna attività didattica o amministrativa da un periodo di tempo predefinito. La tabella `FACT_ANVUR` viene usata per il calcolo degli indici Anvur; per ogni studente indica come esso incide su ogni singolo indice Anvur, all'interno di essa è presente anche la colonna che riporta i risultati delle previsioni per ogni studente.

3.3 Creazione dei metadati

Con la pipeline `ASC_MAIN`, si procede alla creazione e al popolamento di tutte le tabelle necessarie per la costruzione del dataset di feature. Tale dataset fungerà da base per l'addestramento del modello di machine learning che avrà il compito di predire la probabilità di ritiro di uno studente. Per la preparazione del dataset, è stata creata la procedura `creazione_metadati`. Essa aggrega i dati di tutti gli studenti in un'unica tabella, utile per l'analisi predittiva. Quest'ultima conterrà tutte le informazioni dettagliate degli studenti che possono servire per la previsione, tra cui il corso a cui sono iscritti, l'età, la scuola di provenienza, se sono studenti lavoratori, i contributi dovuti all'università, i CFU conseguiti etc.

In particolare le informazioni relative ai CFU conseguiti sono state acquisite con un elevato livello di granularità, analizzando i dati per ogni Anno Accademico e suddividendo il primo anno nei seguenti periodi:

- S0: è il periodo che va da settembre/ottobre fino alla prima sessione di esami, in cui lo studente effettua l'immatricolazione e segue le lezioni.
- S1: sessione invernale, da gennaio a marzo.
- S2: sessione estiva, da maggio a fine luglio.
- S3: sessione autunnale, da settembre a ottobre.
- S4: periodo successivo all'autunno del primo Anno Accademico.

Questa suddivisione dettagliata del primo anno consente di:

- analizzare con maggiore precisione il percorso formativo degli studenti;
- identificare eventuali difficoltà o ritardi nell'acquisizione dei CFU;
- intervenire tempestivamente con misure di supporto mirate.

In aggiunta, per gli anni successivi al primo, i CFU sono stati raggruppati per Anno Accademico. Vengono fornite informazioni complete e precise sui CFU conseguiti, garantendo una base solida per l'analisi e la valutazione del percorso formativo degli studenti.

La scelta di questi quattro periodi temporali deriva dall'analisi del grafico in Figura 3.9:

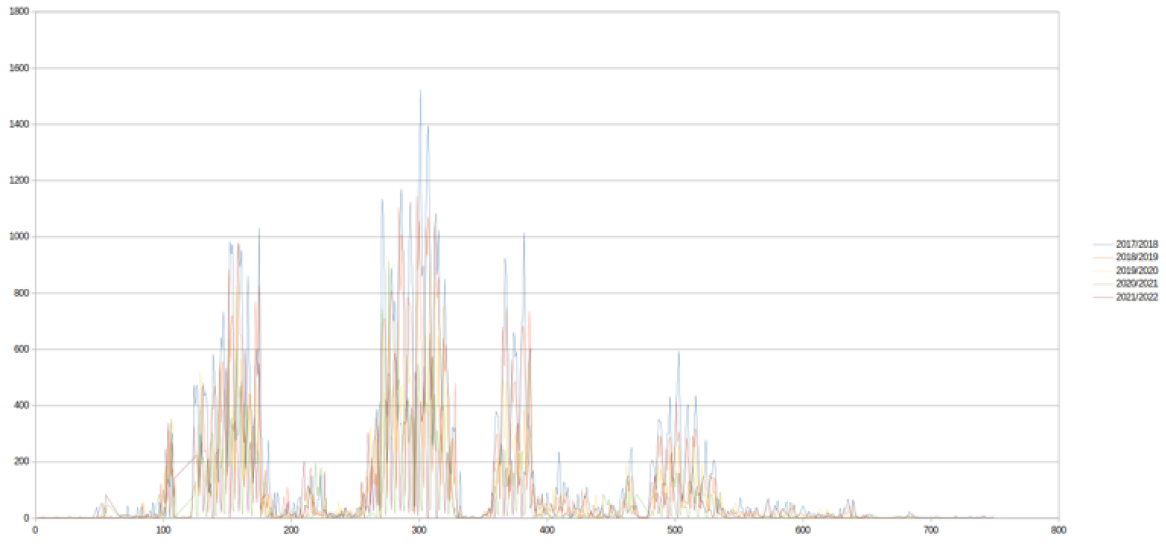


Figura 3.9: Esami durante l'Anno Accademico

Questa figura riporta il numero di esami che vengono effettuati dagli studenti durante il primo anno accademico; come si può notare c'è un primo periodo in cui non avviene nulla, perché subito dopo le immatricolazioni non vengono effettuati esami, ma vengono solo seguite le lezioni. Segue un primo picco di esami rappresentato dalla sessione invernale che va da gennaio a fine febbraio o inizio marzo, e che è stato indicato con S1, seguito da un periodo di diminuzione di eventi e, subito dopo, da un altro picco rappresentato dalla sessione estiva che va da giugno a luglio; questo periodo è stato contrassegnato con S2. Nel mese di agosto non ci sono attività; queste vengono riprese nel periodo di settembre - ottobre, chiamato S3. Tutto ciò che viene dopo è riportato come S4.

Nella fase di predizione del successo universitario, l'attenzione si concentra principalmente sul periodo che va da ottobre fino alla fine della prima sessione invernale, nonostante un'analisi ha dimostrato che, in realtà, dei risultati molto più accurati si ottengono prendendo in considerazione un periodo più lungo.

	Step	Triennale	Ciclo Unico 5	Magistrale	Ciclo Unico 6
Immatricolazione	0	70,88%	75,64%	79,33%	82,35%
Sessione Invernale	1	79,96%	79,96%	83,68%	87,42%
Sessione Estiva	2	83,74%	85,52%	86,06%	91,01%
Sessione Autunnale	3	84,03%	87,35%	87,27%	91,50%
Totale Anno	4	85,81%	89,34%	88,86%	91,01%

Tabella 3.1: Valori dell'accuratezza in base al periodo dell'anno e al tipo di corso

Come si può notare dalla Figura 3.10 e dalla Tabella 3.1 si otterrebbe un'accuratezza superiore, addirittura pari al 91% per le lauree a ciclo unico, se venisse preso in considerazione tutto il primo Anno Accademico, fino al periodo S4. Purtroppo non è possibile effettuare questa scelta, ma c'è la necessità di fermarsi al periodo S1 perché solo in questo caso l'Università può mettere in atto misure di supporto mirate, come corsi di recupero o attività di tutoraggio, per facilitare l'adattamento degli studenti e prevenire l'abbandono.

L'analisi di S1 assume, quindi, un ruolo chiave per la progettazione di interventi efficaci e tempestivi a sostegno del successo degli studenti.

Altro problema da prendere in considerazione è verificare se è possibile prendere tutti i dati presenti nel dataset.

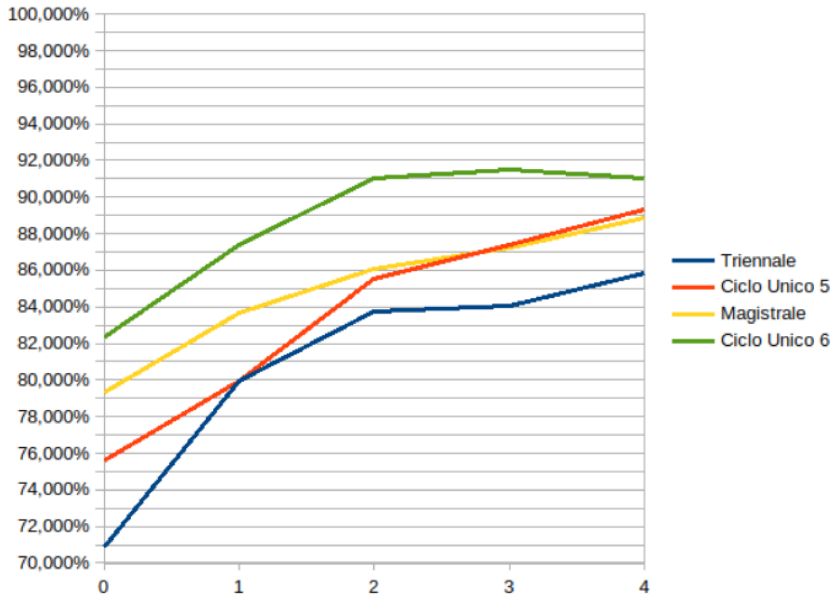


Figura 3.10: Andamento dell'accuratezza in base ai periodi presi in considerazione per le previsioni

Accuracy	Anno Inizio															
Anno Fine	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022
2008	71%															
2009	71%	70%														
2010	73%	71%	71%													
2011	72%	73%	73%	75%												
2012	74%	72%	73%	74%	74%											
2013	73%	74%	73%	75%	74%	74%										
2014	74%	73%	74%	74%	74%	74%	72%									
2015	74%	74%	75%	74%	74%	74%	73%	72%								
2016	74%	74%	75%	75%	75%	74%	73%	74%	74%							
2017	75%	76%	75%	76%	76%	76%	75%	75%	77%	75%						
2018	75%	75%	76%	76%	77%	76%	75%	76%	78%	78%	77%					
2019	76%	76%	76%	78%	76%	77%	77%	78%	79%	79%	81%	82%				
2020	77%	77%	78%	79%	78%	78%	79%	79%	81%	81%	82%	83%	88%			
2021	77%	78%	78%	79%	80%	79%	80%	80%	81%	83%	85%	86%	91%	95%		
2022	79%	79%	79%	79%	79%	80%	80%	82%	82%	83%	86%	87%	91%	96%	100%	
2023	79%	79%	79%	80%	81%	80%	81%	82%	84%	84%	86%	88%	92%	96%	100%	100%

Figura 3.11: Andamento delle previsioni all'aumentare del campione di dati

La Figura 3.11 illustra la relazione tra la dimensione del campione analizzato e la precisione delle previsioni per un modello di predizione del conseguimento della laurea. Si osserva una netta correlazione positiva tra la dimensione del campione e la precisione delle previsioni. In altre parole, all'aumentare del numero di studenti analizzati, la precisione delle previsioni migliora. Questo è un risultato intuitivo, in quanto un campione più ampio fornisce all'algoritmo di training una maggiore varietà di dati su cui basare le sue previsioni. La curva che rappresenta la precisione delle previsioni aumenta gradualmente dal 2007 al 2019, evidenziando un miglioramento costante anno dopo anno.

A partire dal 2020, si osserva un'anomalia: la percentuale di accuratezza raggiunge il 100%. Questo risultato, seppur a prima vista positivo, è, in realtà, un indicatore di un problema con il dataset utilizzato. Le cause di tale picco sono da imputarsi al fatto che gli unici studenti che hanno concluso il loro percorso di studi in questo periodo sono quelli che hanno usufruito di abbreviazioni di corso. Questi casi, essendo facilmente prevedibili (laurea quasi certa), "sporcano" il dataset utilizzato per l'algoritmo di training, influenzando negativamente la sua accuratezza. Per rimediare a tale problema sono stati eliminati i dati relativi ai periodi più recenti.

CAPITOLO 4

Progettazione degli algoritmi di predizione

Questo capitolo esplora le diverse tecniche di classificazione utilizzate per assegnare ai dati delle categorie specifiche. Verrà introdotto Databricks, la tecnologia usata in questo progetto per effettuare l'attività di previsione. Si passerà, poi, all'illustrazione dei principali modelli usati da Databricks per effettuare la classificazione e verranno definite le metriche utili a valutare i modelli predittivi.

4.1 La classificazione

Per effettuare le previsioni sugli abbandoni sono state impiegate tecniche di machine learning. Queste tecniche sono già state utilizzate in altri studi dello stesso tipo e vengono utilizzate con successo per prevedere gli abbandoni anche in contesti diversi da quello universitario, come ad esempio nei settori di:

- Telecomunicazioni: per prevedere la disdetta di abbonamenti telefonici o internet.
- E-commerce: per prevedere la probabilità che un cliente non completi un acquisto.
- Finanza: per prevedere il rischio di insolvenza di un cliente.

Il machine learning, Aized Amin Soofi [2017], è un campo interdisciplinare che si basa su concetti di informatica, statistica, scienze cognitive, ingegneria, teoria dell'ottimizzazione e molte altre discipline matematiche e scientifiche. Il suo obiettivo è quello di sviluppare algoritmi che siano in grado di apprendere dai dati e di migliorare le proprie prestazioni nel tempo, senza essere esplicitamente programmati. Esistono, principalmente, due grandi categorie di machine learning:

- Machine learning supervisionato: l'algoritmo viene addestrato su un set di dati con esempi già etichettati, imparando a classificare nuovi dati in base alle etichette presenti nel set di addestramento.
- Machine learning non supervisionato: in questo caso, l'algoritmo viene addestrato su un set di dati senza etichette impara, quindi, a identificare schemi e relazioni nei dati senza essere guidato da esempi predefiniti.

Le tecniche di apprendimento supervisionato possono essere ulteriormente divise in due categorie:

- Classificazione: l'obiettivo è di assegnare un'etichetta rappresentante una classe a un nuovo dato in base a un set di dati di addestramento con etichette già note.
- Regressione: l'obiettivo è di prevedere un valore numerico continuo per un nuovo dato in base a un set di dati di addestramento con valori numerici già noti.

Effettuare delle previsioni sugli abbandoni rientra nella classificazione; l'algoritmo, infatti, deve prevedere l'appartenenza ad una classe di uno studente, cioè deve prevedere se fa parte di coloro che abbandonano o che riescono a laurearsi. Esistono diversi modelli per effettuare la classificazione; questi sono sintetizzati in Figura 4.1.

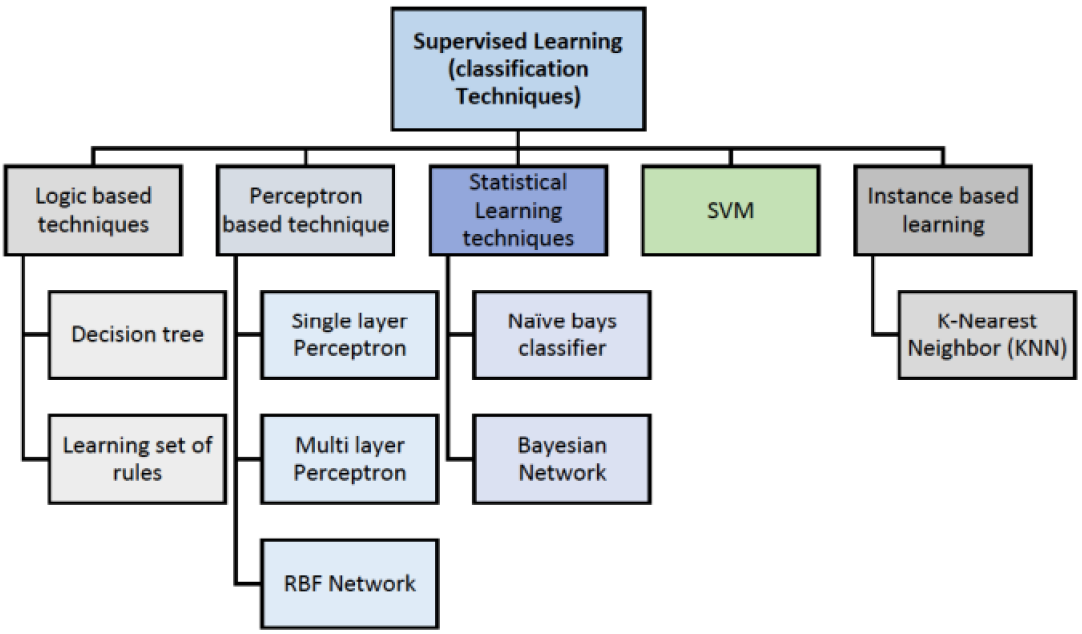


Figura 4.1: Tecniche di classificazione

Le principali tecniche di classificazione sono:

- Decision tree: descrive una struttura ad albero dove i nodi foglia rappresentano le classi, le ramificazioni l'insieme delle proprietà che portano a quelle classi; di conseguenza, ogni nodo interno risulta essere una macro-classe costituita dall'unione delle classi associate ai suoi nodi figli.
- Bayesian network: è un modello grafico probabilistico che rappresenta un insieme di variabili stocastiche con le loro dipendenze condizionali attraverso l'uso di un grafo aciclico diretto (DAG). Per esempio, una rete Bayesiana potrebbe rappresentare la relazione probabilistica esistente tra i sintomi e le malattie; dati i sintomi, la rete può essere usata per calcolare la probabilità della presenza di diverse malattie.
- K-Nearest Neighbor: è un algoritmo utilizzato nel riconoscimento di pattern per la classificazione di oggetti, basandosi sulle caratteristiche degli oggetti vicini a quello considerato. L'input è costituito dai k esempi di addestramento più vicini nello spazio delle funzionalità. Un oggetto è classificato da un voto di pluralità dei suoi vicini, con l'oggetto assegnato alla classe più comune tra i suoi k vicini più vicini, dove k è un numero intero positivo tipicamente piccolo. Se k = 1 l'oggetto viene semplicemente assegnato alla classe di quel singolo vicino più prossimo.

- Support Vector Machines: è una rappresentazione degli esempi come punti nello spazio mappati in modo tale che gli esempi, appartenenti alle due diverse categorie, siano chiaramente separati da uno spazio il più possibile ampio. I nuovi esempi sono, quindi, mappati nello stesso spazio e la previsione della categoria alla quale appartengono viene fatta sulla base del lato nel quale ricadono.

4.2 Databricks

Per implementare la parte predittiva è stato usato Azure Databricks, una piattaforma unificata di analisi aperta per la creazione, la distribuzione, la condivisione e la gestione di dati, analisi e soluzioni di intelligenza artificiale di livello aziendale su larga scala. Questa piattaforma usan Apache Spark, è la tecnologia che alimenta i cluster di calcolo e sql warehouse. Azure Databricks è, quindi, ottimizzata per Apache Spark offrendo, così, una piattaforma efficiente e semplice.

Azure Databricks supporta diversi linguaggi tra cui: Python, Scala, R, Java e SQL, oltre ai framework e le librerie di data science, come per esempio TensorFlow, PyTorch e scikit-learn. L'area di lavoro di Azure Databricks offre un'interfaccia unificata e strumenti per la maggior parte delle attività di dati, tra cui:

- pianificazione e gestione dell'elaborazione dati, in particolare ETL;
- generazione di dashboard e visualizzazioni;
- gestione di sicurezza, governance, disponibilità elevata e ripristino di emergenza;
- individuazione, annotazione ed esplorazione dei dati;
- modellazione, rilevamento e gestione di modelli di Machine Learning;
- soluzioni di intelligenza artificiale generative.

A differenza di molte aziende di dati aziendali, Azure Databricks non forza la migrazione dei dati in sistemi di archiviazione proprietari per l'uso della piattaforma. Si configura, invece, un'area di lavoro configurando integrazioni sicure tra la piattaforma Azure Databricks e l'account cloud, e quindi, i cluster di calcolo sono distribuiti usando le risorse cloud nell'account per elaborare e archiviare i dati.

In questo progetto Azure Databricks è stato usato per effettuare delle piccole trasformazione al dataset utili per il training delle reti, e per l'addestramento dei modelli che verranno poi usati per effettuare le previsioni. Nello specifico, per effettuare una migliore analisi dei diversi modelli predittivi e per ottenere dei risultati ottimali è stata usata una componente di Databricks chiamata AutoML. Databricks AutoML effettua le seguenti operazioni:

- Prepara il set di dati per il training del modello, ad esempio, esegue il rilevamento sbilanciato dei dati per i problemi di classificazione prima del training del modello.
- Esegue l'iterazione per eseguire il training e ottimizzare più modelli, in cui ogni modello viene costruito da componenti open source e può essere facilmente modificato e integrato nelle pipeline di Machine Learning.
- Distribuisce automaticamente le versioni di valutazione dell'ottimizzazione degli iperparametri tra i nodi di lavoro di un cluster.
- Valuta i modelli basati su algoritmi dei pacchetti scikit-learn, xgboost, LightGBM, Prophet e ARIMA .

- Visualizza i risultati e fornisce un notebook Python con il codice sorgente per ogni esecuzione di valutazione in modo da poter esaminare, riprodurre e modificare il codice.
- Calcola le statistiche di riepilogo sul set di dati e salva queste informazioni in un notebook che è possibile esaminare in un secondo momento.

4.3 Modelli

I modelli di AutoML sono stati usati per prevedere il successo degli studenti, in particolare la probabilità che uno studente si ritiri o si laurei. Per effettuare la classificazione AutoMl utilizza i seguenti modelli:

- decision tree;
- random forest;
- logistic regression;
- XGBoost;
- LightGBM.

Nelle successive sezioni i modelli verranno illustrati nel dettaglio.

4.3.1 Decision tree

Il decision tree, Charbuty e Abdulazeez [2021], è uno degli algoritmi più potenti ed utilizzati nella classificazione. Detti anche alberi decisionali, sono un modello sequenziale che unisce in modo efficiente e coeso una serie di test di base; in ogni test, un attributo numerico viene confrontato con un valore di soglia. Le regole concettuali degli alberi decisionali sono molto più facili da costruire rispetto ai pesi numerici delle reti neurali che si basano su connessioni tra nodi. I decision tree vengono utilizzati principalmente per raggruppare i dati e sono dei modelli di classificazione molto usati nel data mining.

Ogni albero è composto da nodi e rami, ogni nodo rappresenta una caratteristica di una categoria da classificare, e ogni sottoinsieme definisce un valore che quel nodo può assumere. Grazie alla loro semplicità di analisi e alla precisione su diversi tipi di dati, gli alberi decisionali, hanno trovato applicazione in molti settori. In Figura 4.2 è mostrato un esempio di albero decisionale.

Nel dettaglio il tipo di albero decisionale usato da `sklearn` è il CART (Classification And Regression Tree), Bittencourt e Clarke [2003], prevede l'identificazione e la costruzione di un albero decisionale binario, questo significa che per ogni nodo dell'albero vengono poste delle domande a cui è possibile avere solo due risposte, ad esempio "sì" o "no", "alto" o "basso", utilizzando un campione di dati di addestramento per i quali è nota la classificazione corretta. Il numero di elementi nei due sottogruppi definiti ad ogni suddivisione binaria, corrispondenti ai due rami che emergono da ciascun nodo intermedio, diventa via via più piccolo; di conseguenza, è necessario un campione di addestramento sufficientemente ampio per ottenere dei buoni risultati.

La costruzione dell'albero decisionale inizia da un nodo radice `t` derivato dalla variabile all'interno dello spazio delle feature che minimizza la misura dell'impurezza dei due nodi figli. La misura dell'impurezza nel nodo t, indicata con $i(t)$, è mostrata nella Formula 4.3.1.

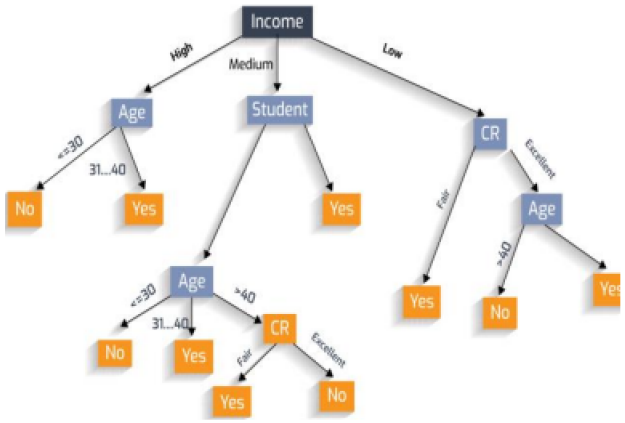


Figura 4.2: Esempio di albero decisionale

$$i(t) = -\sum_{j=1}^{k} p(w_j|t) \log p(w_j|t) \tag{4.3.1}$$

Dove $p(w_j|t)$ rappresenta la proporzione di pattern x_i assegnati alla classe w_j nel nodo t. Ogni nodo non terminale viene successivamente diviso in altri due nodi, t_L e t_R, in modo tale che p_L e p_R rappresentino le proporzioni di entità passate, rispettivamente, ai nuovi nodi t_L e t_R. La suddivisione ottimale è quella che massimizza la differenza espressa nella Formula 4.3.2:

$$\Delta_i(s,t) = i(t) - p_L i(t) - p_R i(t) \tag{4.3.2}$$

L'albero decisionale cresce attraverso successive suddivisioni fino a quando non si raggiunge uno stadio in cui la misura di impurità non diminuisce in modo significativo se si implementa un'ulteriore suddivisione s. Quando si raggiunge questo stadio, il nodo t non viene ulteriormente suddiviso e diventa automaticamente un nodo terminale. La classe w_j associata al nodo terminale t è quella che massimizza la probabilità condizionale $p(w_j|t)$.

4.3.2 Random forest

I random forest, Speiser *et al.* [2019] , sono una tecnica di machine learning molto usata per lo sviluppo di modelli predittivi. Sono insiemi di alberi di classificazione introdotti per la prima volta da Breiman nel 2001. Questi alberi sono modelli semplici che utilizzano suddivisioni binarie su variabili predittrici per determinare le predizioni dell'output.

Nei random forest, vengono costruiti molti alberi di classificazione e regressione utilizzando sottoinsiemi di dati di addestramento e sottoinsiemi casuali di variabili predittrici selezionati casualmente per modellare gli outcome. I risultati di ciascun albero vengono aggregati per fornire una previsione per ogni osservazione, pertanto, i random forest spesso offrono una maggiore accuratezza rispetto a un singolo modello ad albero decisionale, pur mantenendo alcune delle qualità vantaggiose dei modelli ad albero, ad esempio la capacità di interpretare le relazioni tra variabili predittrici e outcome. I random forest si collocano tra i modelli con la più alta accuratezza di predizione rispetto ad altri modelli in ambito di classificazione.

Uno dei principali vantaggi del loro utilizzo per lo sviluppo di modelli predittivi è la capacità di gestire dataset con un numero elevato di variabili predittrici. Tuttavia, nella pratica, è sempre preferibile minimizzare il numero di predittori necessari per ottenere previsioni sull'outcome, al fine di migliorare l'efficienza.

4.3.3 Logistic regression

Nella libreria `scikit-learn` per il machine learning, il modello logistic regression è implementato nella classe `LogisticRegression`. Nonostante il nome, secondo la terminologia di `scikit-learn/ML`, viene implementato come un modello lineare per la classificazione, piuttosto che per la regressione. In questo modello, le probabilità associate ai possibili esiti di un singolo evento vengono modellate utilizzando una funzione logistica.

Il logistic regression è un algoritmo fondamentale di machine learning usato per predire l'esito binario a partire da vari attributi indipendenti di un dataset. Viene utilizzato per calcolare la probabilità che un evento binario specifico si verifichi o meno, assumendo solo valori 0 o 1. Ad esempio, può essere usata dalle società di vendita per prevedere se un cliente acquisterà un prodotto oppure no. Esso è un'estensione del modello di regressione lineare, dalla Figura 4.3 si può osservare che la funzione logistica comprime il grafico del modello lineare tra 0 e 1.

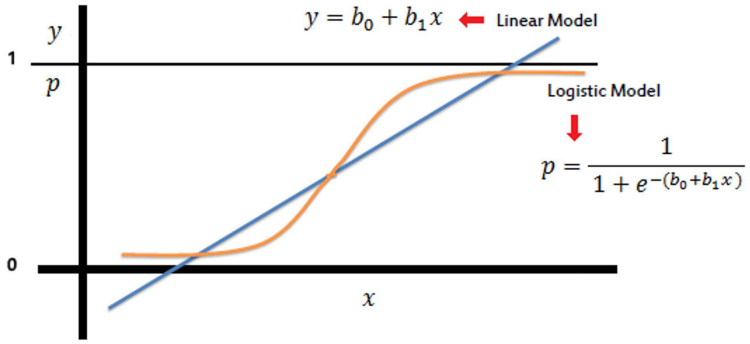


Figura 4.3: Grafico della curva logistic regression

4.3.4 XGBoost

XGBoost (XGB), Chen e Guestrin [2016], è un potente algoritmo di machine learning basato su Gradient Boosting con struttura ad albero. Può essere utilizzato sia per modelli di regressione che di classificazione. Esso è un metodo di apprendimento automatico di tipo ensemble. Supporta tre tipi di Gradient Boosting: Gradient Boosting, Gradient Boosting stocastico e Gradient Boosting regolarizzato ed è generalmente più veloce degli altri algoritmi apparteneti alla stessa categoria.

Questo modello include una funzione di pruning automatico che interrompe la crescita dell'albero decisionale al raggiungimento di determinati limiti, contribuendo a ridurre l'overfitting del modello sul dataset. XGB raggiunge un livello di accuratezza superiore rispetto ad altri algoritmi di boosting senza compromettere l'utilizzo di memoria e grazie ai suoi numerosi vantaggi, è uno degli algoritmi di Gradient Boosting più utilizzati. Gli step seguiti dall'XGBoost sono:

- Partire con un modello di previsione semplice: si parte addestrando un modello iniziale, solitamente un albero decisionale poco profondo, questo primo modello fornisce una previsione iniziale per l'output desiderato.
- Identificare gli errori: si calcola la differenza, chiamata gradiente, tra le previsioni del modello iniziale e i valori effettivi, target, dei dati.

- Addestra un nuovo modello debole: si addestra un nuovo modello debole, solitamente un altro albero decisionale, per correggere gli errori del modello precedente. Questo nuovo modello si concentra, nello specifico, sulle istanze dove il modello iniziale ha commesso errori.
- Combina i modelli: si combinano le previsioni del modello iniziale e del nuovo modello debole, la combinazione può avvenire tramite una media ponderata, dove un peso maggiore viene assegnato al nuovo modello che ha corretto gli errori in modo più efficace.
- Iterazione: si ripetono il secondo, il terzo e il quarto step finché non si raggiunge una performance di predizione soddisfacente.

In sintesi, l'XGBoost costruisce progressivamente un modello di predizione robusto aggiungendo nuovi modelli deboli che mirano a correggere gli errori dei modelli precedenti. Questo processo iterativo porta a un modello finale in grado di catturare relazioni complesse nei dati.

4.3.5 LightGBM

L'algoritmo LightGBM (LGBM), Aziz [2022], si basa su due concetti: GBDT (Gradient Boosting Decision Tree) e GOSS (Gradient-based One-Sided Sampling), ossia algoritmi di machine learning ampiamente utilizzati per la previsione in diversi ambiti. L'architettura di questo modello si basa su alberi decisionali che possono crescere verticalmente o orizzontalmente. A differenza di altri algoritmi che crescono solo orizzontalmente, LightGBM può espandersi verticalmente dalle foglie con perdite significative, cioè quelle che hanno un errore di predizione elevato, sono le foglie che non classificano correttamente i dati in modo accurato. Questo permette di ottenere una migliore approssimazione della funzione obiettivo e di migliorare l'accuratezza delle previsioni. La Figura 4.4 illustra la crescita verticale di un albero di decisione in LGBM, infatti, la foglia sinistra si espande verticalmente dalla foglia con perdita significativa, anziché spostarsi verso la foglia più a destra.

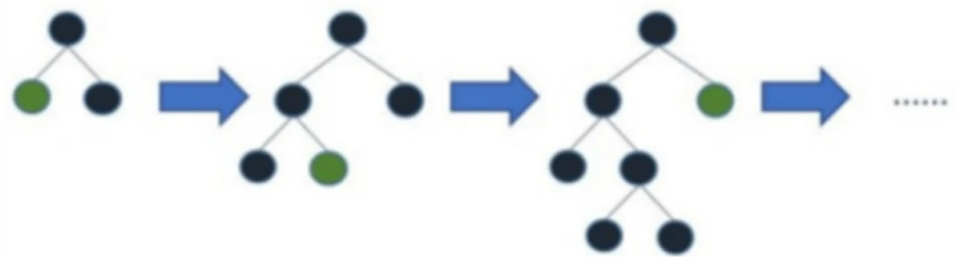


Figura 4.4: Espansione albero ricerca LGBM

In sintesi, LGBM rappresenta un algoritmo potente e versatile per l'analisi di dataset di grandi dimensioni. La sua velocità, accuratezza, efficienza in memoria e capacità di prevenire l'overfitting lo rendono una scelta eccellente per una varietà di compiti di analisi dei dati che vanno dalla classificazione alla regressione. Tuttavia, è importante notare che LGBM potrebbe non essere la scelta ottimale per i dataset di piccole dimensioni; in questi casi, l'algoritmo potrebbe soffrire di overfitting e generare dei modelli poco affidabili.

4.4 Le metriche

Nella valutazione dei modelli di Machine Learning ci sono diverse metriche che forniscono informazioni utili sulle loro prestazioni.

In seguito sono indicate le metriche usate in questo progetto per la valutazione dei modelli:

- Accuracy: un modo semplice per misurare le prestazioni dei modelli di classificazione è quello di contare la proporzione di previsioni che sono corrette. La formula per il calcolo dell'accuracy è riportato nella Formula 4.4.1.

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{True Negative} + \text{False Positive} + \text{False Negative}} \tag{4.4.1}$$

- Recall: misura la capacità del modello di predire un risultato positivo.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{4.4.2}$$

- Precision: si riferisce alla proporzione di casi positivi previsti che sono effettivamente positivi, misura la correttezza delle previsioni positive del modello.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{4.4.3}$$

- F1: è una metrica di valutazione comunemente usata nei problemi di classificazione binaria, è considerata una misura bilanciata che tiene conto sia della precision, correttezza, che della recall, completezza, del modello. Cerca di trovare un equilibrio tra identificare correttamente i veri positivi, evitando falsi positivi, e individuare tutti i casi positivi effettivi, evitando falsi negativi.

$$\text{F1} = 2 * \frac{\text{Precison} * \text{Recall}}{\text{Precison} + \text{Recall}} \tag{4.4.4}$$

- Matrice di confusione: è una tabella che mostra il numero di previsioni corrette ed errate classificate per tipo di risposta. Gli elementi sulla diagonale della matrice indicano il numero di previsioni corrette, mentre gli elementi fuori dalla diagonale denotano il numero di previsioni errate.

CAPITOLO 5

Implementazione degli algoritmi di predizione

In questo capitolo vengono esaminati gli ultimi passaggi per la creazione del dataset che sarà utilizzato per l'addestramento dei modelli e, vengono esplorate le ragioni che hanno portato alla decisione di addestrare modelli distinti per ciascuna tipologia di corso di laurea. Successivamente, vengono presentati i modelli che hanno ottenuto le migliori prestazioni per ogni tipo di corso, evidenziando le sfide legate all'overfitting riscontrate soprattutto nei corsi di laurea magistrale e a ciclo unico. Infine, vengono illustrate le strategie adottate per affrontare efficacemente questo problema.

5.1 Preparazione del dataset di training

Prima di utilizzare il dataset per l'addestramento dei modelli, sono state apportate alcune modifiche per migliorare la qualità e l'efficacia di tale processo. Come descritto nel Capitolo 3, sono stati eliminati tutti i periodi successivi al primo semestre (S1), in quanto non pertinenti all'obiettivo di predizione, sono stati rimossi gli attributi che non forniscono informazioni utili per le predizioni, come il codice fiscale, il nome, il cognome e gli id dei corsi di laurea. Sono state eliminate le variabili che presentavano un'alta correlazione tra loro; infatti, in tali casi, la presenza di entrambe le variabili nel dataset risulta ridondante e si può mantenere solo una delle due per l'addestramento delle reti senza sacrificare l'accuratezza del modello.

In questa fase è stata adottata una strategia di apprendimento differenziata, basata sulla creazione di dataset distinti per effettuare il training dei modelli; in particolare, i dataset utilizzati sono stati i seguenti:

- *dataset per corsi triennali*: focalizzato sulle peculiarità e sui comportamenti tipici di questo percorso formativo;
- *dataset per corsi magistrali*: progettato per cogliere le specificità e le sfide della formazione di secondo livello;
- *dataset per corsi a ciclo unico*: calibrato sulle esigenze e il profilo di questo tipo di corso di laurea.

La scelta di diversificare i dataset di training per i modelli è stata motivata da un'evidenza empirica, ossia dal fatto che il comportamento degli studenti varia considerevolmente in base al tipo di corso di laurea frequentato. I motivi di ciò sono i seguenti:

- Gli studenti triennali affrontano un primo approccio al mondo universitario, con un carico di lavoro e un livello di complessità generalmente inferiore rispetto ai successivi livelli di formazione.

- Gli studenti magistrali si concentrano su un ambito di studio più specifico e approfondito, con una maggiore autonomia e responsabilità nel proprio percorso formativo.
- Gli studenti a ciclo unico si impegnano in un percorso di studi di lunga durata, caratterizzato da una forte interdisciplinarità e da una specifica preparazione professionalizzante.

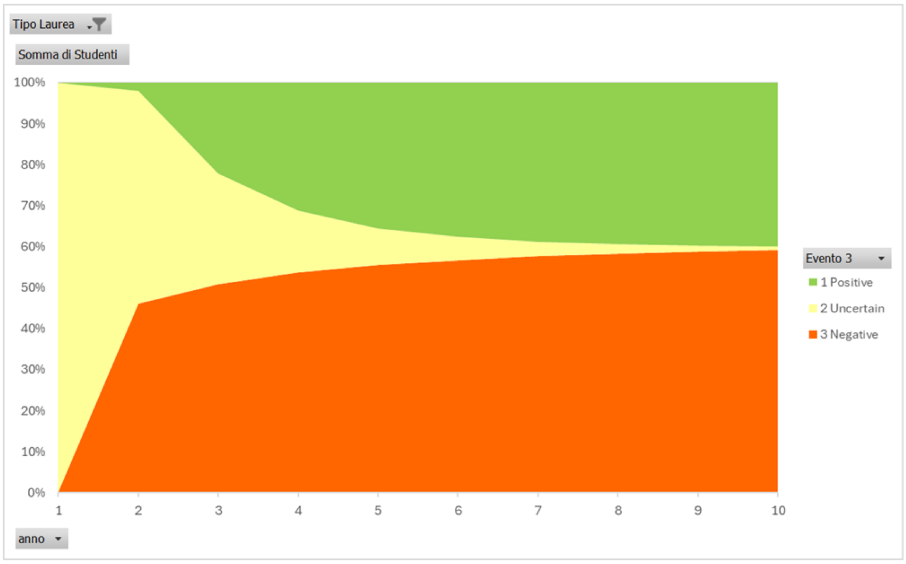


Figura 5.1: Abbandono universitario per lauree triennali

La Figura 5.1 illustra l'andamento degli abbandoni degli studenti triennali; sull'asse delle ascisse sono riportati i diversi anni di iscrizione degli studenti, mentre l'asse delle ordinate indica la percentuale di studenti; la curva rossa rappresenta gli studenti che abbandonano il corso di studi, mentre quella verde rappresenta gli studenti che si laureano. È evidente come il numero di studenti che si ritira sia nettamente superiore a quello di coloro che riescono a laurearsi.

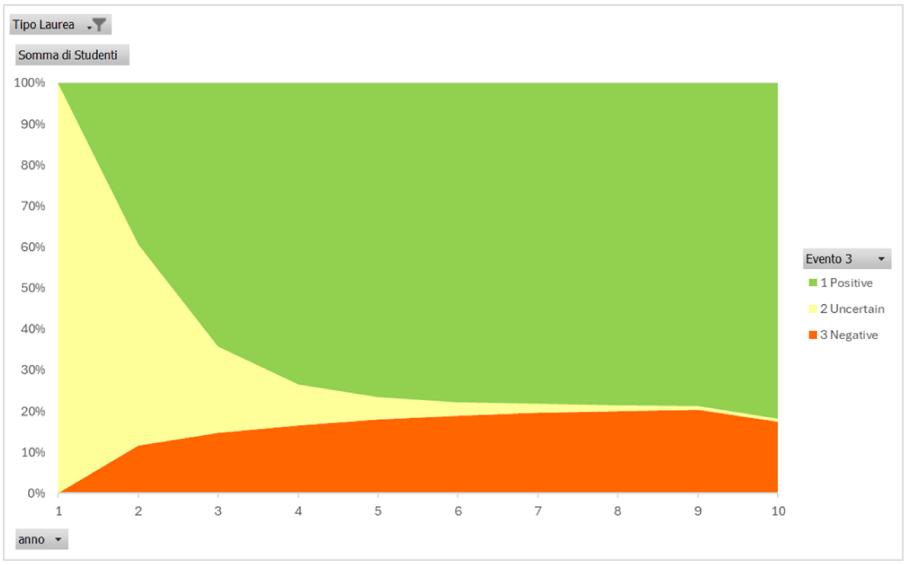
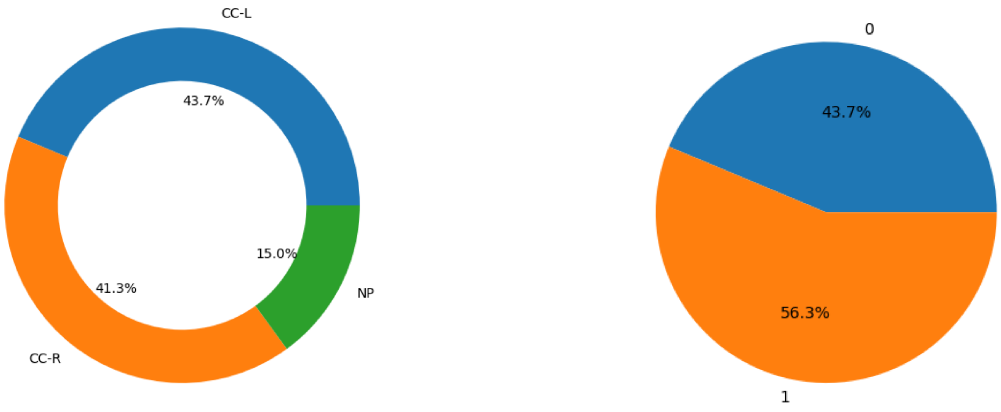


Figura 5.2: Abbandono universitario per lauree magistrali

La Figura 5.2 illustra, invece, l'andamento dei ritirati e dei laureati per le lauree magistrali; in contrasto con quanto osservato per le lauree triennali, si può notare un andamento quasi opposto per questo tipo di corsi; infatti, la percentuale di laureati è di gran lunga superiore a quella dei ritirati. Queste sono le ragioni che hanno portato a diversificare i modelli in base al tipo di corso di laurea portando a diversi benefici come:

- *Migliore accuratezza*: i modelli specifici per ogni tipologia di corso di laurea possono apprendere e adattarsi alle caratteristiche e alle esigenze specifiche degli studenti, ottenendo una maggiore precisione nelle predizioni.
- *Maggiore efficienza*: la focalizzazione su un contesto specifico permette di ottimizzare il processo di apprendimento e di ridurre il rischio di sovraccarico del modello con informazioni non pertinenti.
- *Risultati più significativi*: la diversificazione dei modelli consente di ottenere una rappresentazione più realistica e accurata del comportamento degli studenti, fornendo informazioni più utili per la progettazione di interventi mirati e per l'ottimizzazione dei processi di apprendimento.

Al termine di questa fase si hanno tre dataset, uno per ogni tipo di corso di laurea. Per ogni dataset è stata studiata la distribuzione degli eventi d'uscita degli studenti. Dalla Figura 5.3(a) si ha che il 43,7% degli studenti iscritti ad una laurea triennale riesce a laurearsi, il 15% non effettua più alcun tipo di evento, sono considerati come ritirati, e il 41,3% si ritira. Per effettuare il training dei modelli questi valori sono stati convertiti in valori numerici; nel dettaglio è stato assegnato il valore 0 agli studenti che rinunciano agli studi (questa categoria include gli eventi CC-R e NP) mentre con 1 sono stati identificati gli studenti che si laureano. Come si può notare dalla Figura 5.3(b) si ha una distribuzione abbastanza equa delle due classi.

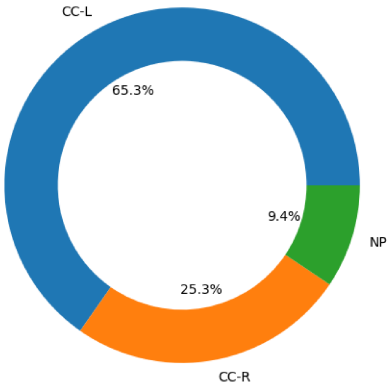


(a) Distribuzione degli eventi d'uscita per lauree triennali

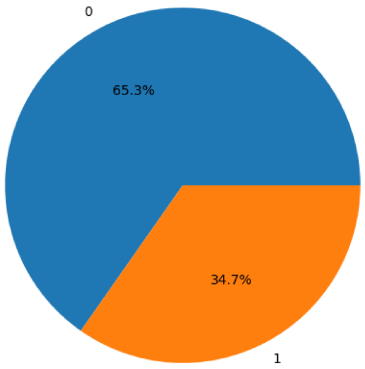
(b) Distribuzione delle classi 0 e 1 per i corsi triennali

Figura 5.3: Distribuzione degli eventi lauree triennali

Lo stesso studio è stato effettuato per i corsi di laurea a ciclo unico; in questo caso, dalla Figura 5.4(a), si può dedurre una netta maggioranza di studenti che riescono a concludere il proprio corso di studi con esito positivo, pari al 65,3%, una piccola parte si ritira, pari al 25,3%, infine, solo una parte minore non effettua più nessun evento. Questo ha un incidenza sulla distribuzione delle due classi con una maggioranza della classe 0, studenti che si laureano, e una minoranza della classe 1.



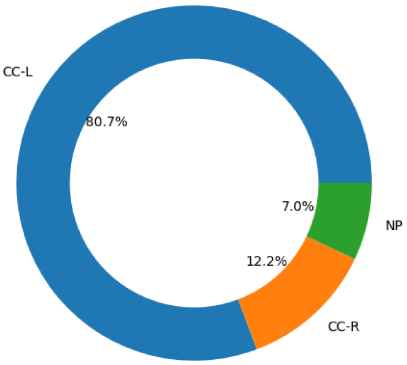
(a) Distribuzione degli eventi d'uscita per lauree ciclo unico



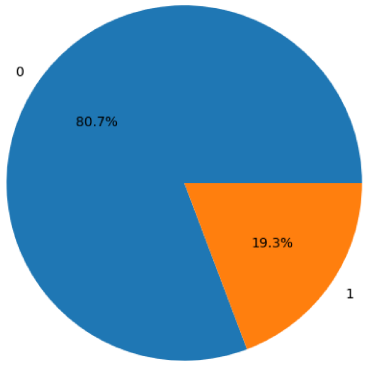
(b) Distribuzione delle classi 0 e 1 per i corsi a ciclo unico

Figura 5.4: Distribuzione degli eventi per le lauree a ciclo unico

Infine, per i corsi di laurea magistrale il divario è ancora più evidente; dalle Figure 5.5(a) e 5.5(b) si evince, infatti, una percentuale di laureati pari all'80,7% e una netta minoranza di studenti che si ritirano; questo grande squilibrio tra le classi porterà a un problema durante la fase di training che verrà discussa nella sezione successiva.



(a) Distribuzione degli eventi d'uscita per lauree magistrali



(b) Distribuzione delle classi 0 e 1 per i corsi magistrali

Figura 5.5: Distribuzione degli eventi per le lauree magistrali

5.2 Fase di training

In questa fase, i modelli descritti nella Sezione 4.3 sono stati addestrati utilizzando AutoML, una funzionalità di Databricks che automatizza il processo di training consentendo di:

- impostare automaticamente i parametri per i diversi modelli;
- selezionare il modello più appropriato in base alla metrica di ottimizzazione scelta.

In questo progetto è stata scelta come metrica di riferimento l'accuracy. AutoML ha, quindi, addestrato i diversi modelli e ha selezionato quello con l'accuracy più elevata. Al

termine dell'addestramento sono stati generati dei notebook con i dettagli del processo di training, inclusi i modelli utilizzati, i parametri ottimizzati e le metriche di performance per ciascun modello. La suddivisione del dataset in parti per l'apprendimento (training), la valutazione (test) e la convalida è stata realizzata automaticamente da AutoML, utilizzando una proporzione del 60% per il training, 20% per il test e 20% per la convalida.

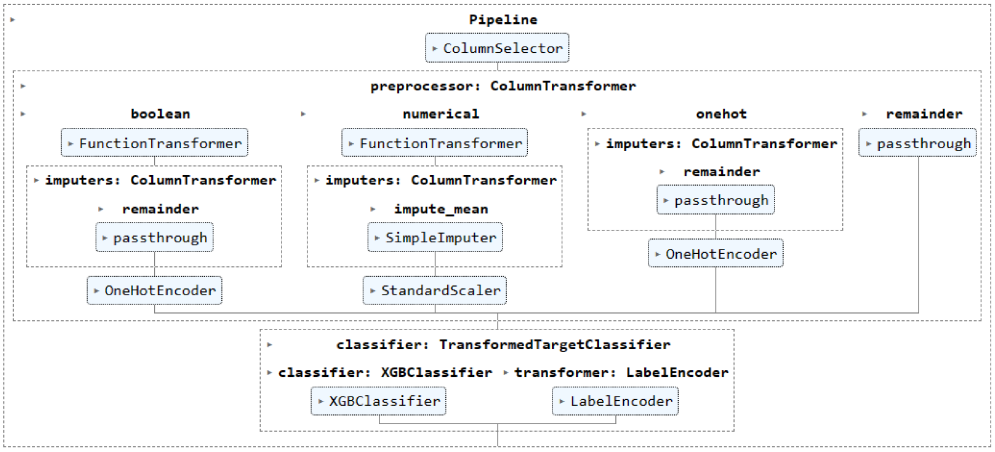


Figura 5.6: Pipeline del modello XGBoost

La Figura 5.6 mostra una pipeline di apprendimento automatico usata da AutoML per effettuare la classificazione. La pipeline inizia con un selettore di colonne, che viene utilizzato per selezionare le colonne dei dati che verranno adattate per addestrare il modello. I dati vengono pre-elaborati da un trasformatore di colonne, che include la rimozione dei valori mancanti, la codifica delle variabili categoriche e la normalizzazione delle feature. I dati pre-elaborati vengono, quindi, utilizzati per addestrare un modello di classificazione. Il tipo di modello di classificazione utilizzato dipende dalla libreria di apprendimento automatico utilizzata; nella Figura 5.6 viene utilizzata la libreria `scikit-learn` e viene utilizzato un modello di classificazione XGBoost. Il modello di classificazione viene, quindi, valutato per assicurarsi che sia accurato; se il modello non è sufficientemente accurato è necessario riaddestrarlo con più dati o con un algoritmo diverso; se è sufficientemente accurato, può essere utilizzato per classificare nuovi dati.

	Accuracy	**Precision**	**Recall**	**F1**
XGBoost	0,76	0,73	0,70	0,72
Logistic regression	0,68	0,65	0,59	0,62
LightGBM	0,75	0,72	0,69	0,70
Decision Tree	0,64	0,57	0,66	0,61
Random forest	0,64	0,62	0,45	0,52

Tabella 5.1: Valori delle metriche per i modelli dei corsi triennali

Dopo aver effettuato il training di tutti i modelli sono state analizzate le loro prestazioni per capire quale sia il migliore da poter utilizzare per effettuare le previsioni. Nella Tabella 5.1 e nella Figura 5.7 sono riportate le metriche analizzate per la scelta del classificatore migliore sui corsi di studi triennali; la classe 0 corrisponde alla classe degli studenti che si laureano mentre la classe 1 è la classe degli studenti che non concludono il corso di studi. Analizzando le matrici di confusione e le metriche dei vari modelli, risalta subito all'occhio che il modello migliore risulta essere XGBoost; questo riesce a prevedere, con dei buoni valori di tutte le metriche, chi abbandona e chi riesce a laurearsi senza avere alcun tipo di problema.

	Accuracy	Precision	Recall	F1
XGBoost	0,78	0,77	0,91	0,84
Logistic regression	0,75	0,75	0,91	0,82
LightGBM	0,76	0,74	0,94	0,83
Decision Tree	0,72	0,72	0,92	0,80
Random forest	0,72	0,71	0,91	0,80

Tabella 5.2: Valori delle metriche per i modelli dei corsi ciclo unico

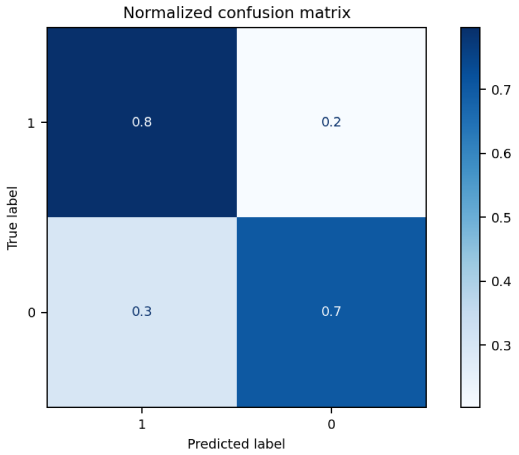
La stessa analisi è stata effettuata per i corsi di laurea a ciclo unico; anche in questo caso dalle matrici di confusione in Figura 5.8, e sopratutto dalla Tabella 5.2, si nota che il miglior modello resta XGBoost. È necessario, comunque, osservare che nonostante le metriche nella Tabella 5.2 sembrano molto buone, addirittura migliori di quelle ottenute dai modelli addestrati per i corsi di laurea triennali, dalla matrice di confusione si evince che, in realtà, tutti i modelli riescono a riconoscere molto bene gli studenti appartenenti alla classe 0, ma hanno alcune difficoltà nel riconoscere gli 1. Questo problema è dovuto a uno sbilanciamento delle classi nel dataset di training visto nella sezione precedente; infatti, per questo corso di studi, ci sono molti laureati e un numero minore di ritirati, cosa che non avveniva per le triennali, dove il dataset risultava essere abbastanza bilanciato; di conseguenza il modello tende ad adattarsi ai dettagli specifici di questa classe, ignorando le caratteristiche più generali che discriminano le classi minoritarie. Poiché la classe minoritaria è sottorappresentata, il modello ha meno dati per imparare a distinguerla correttamente; questo problema prende il nome di overfitting e verrà discusso nella successiva sezione, in cui si illustrerà anche la soluzione implementata.

	Accuracy	Precision	Recall	F1
XGBoost	0,84	0,85	0,96	0,90
Logistic regression	0,82	0,83	0,97	0,90
LightGBM	0,83	0,85	0,96	0,90
Decision Tree	0,81	0,84	0,94	0,89
Random forest	0,81	0,81	1	0,89

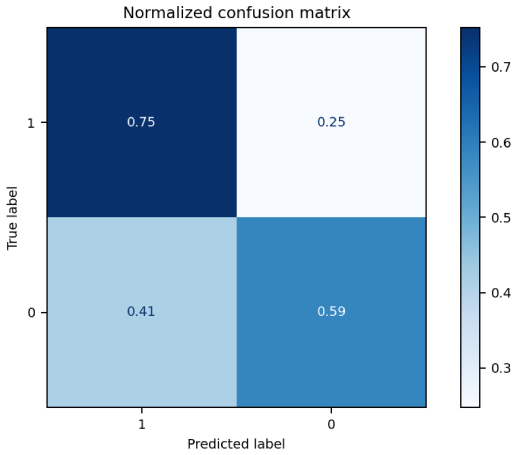
Tabella 5.3: Valori delle metriche per i modelli dei corsi magistrali

Infine, sono stati effettuati gli stessi studi per le lauree magistrali; anche questa volta il modello migliore risulta essere XGBoost; addirittura, guardando la Tabella 5.3, sembra che si ottengano delle metriche migliori rispetto agli altri due modelli visti in precedenza.

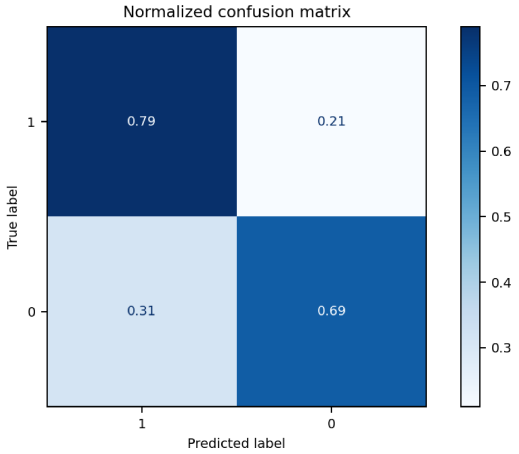
In realtà, guardando le matrici di confusione in Figura 5.9 si osserva che è presente lo stesso problema dei corsi di studi a ciclo unico, addirittura più accentuato. Si avevano, infatti, nel dataset di training un numero di elementi appartenenti alla classe 0 al di sopra di 15.000, mentre il numero di elementi appartenente alla classe 1 era poco al di sopra di 3.000. Si hanno, quindi, delle classi fortemente sbilanciate; questo era stato visto anche nella Figura 5.5; per tali motivi i modelli fanno moltissima fatica a prevedere la classe 1 rispetto alla classe 0. Ancora una volta si ha il problema dell'overfitting; questo fenomeno non viene messo in risalto dalle metriche riportate nella Tabella 5.3, perché ci sono molti casi positivi che vengono previsti bene e pochi casi negativi con un'errata previsione; quindi i casi errati hanno un impatto minore rispetto a quelli esatti; di conseguenza le metriche sembrano essere molto buone. Questo problema viene messo in risalto sopratutto dal modello random forest che mostra una recall pari a uno, in teoria perfetta, perché riesce a prevedere tutti i casi positivi; tuttavia, la corrispettiva matrice di confusione mostra una previsione errata per tutti gli elementi della classe negativa.



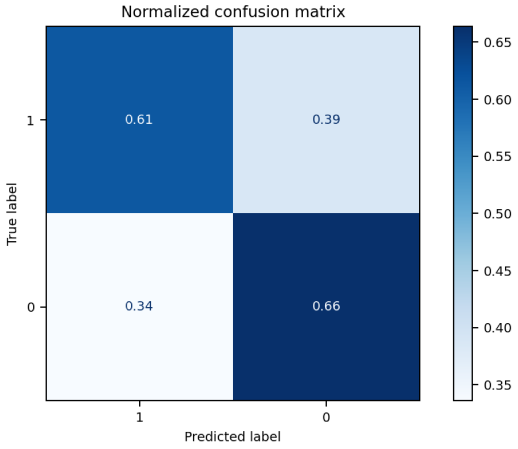
(a) XGBoost



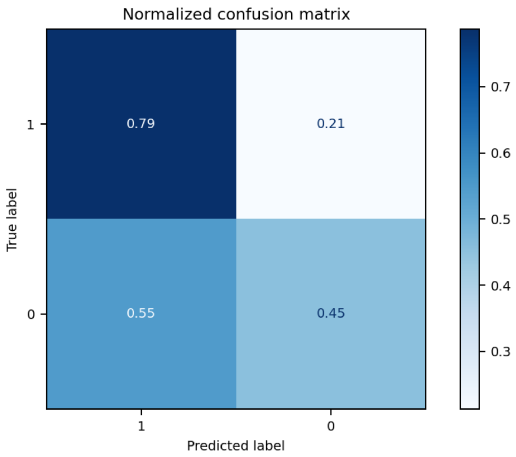
(b) Logistic regression



(c) LightGBM

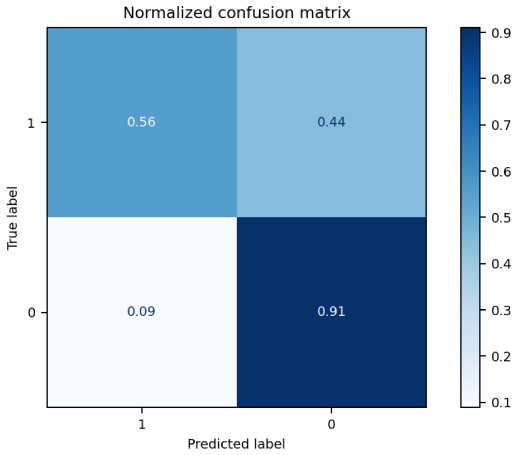


(d) Decision tree

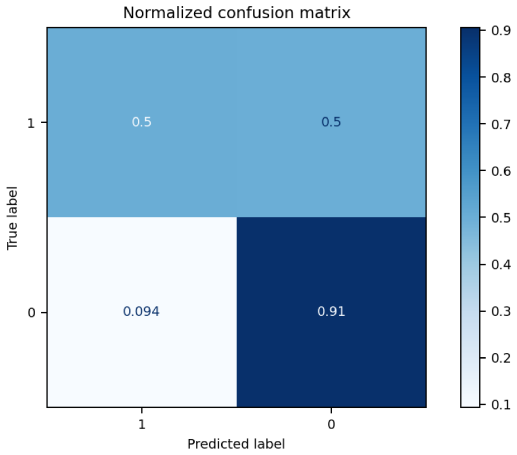


(e) Random forest

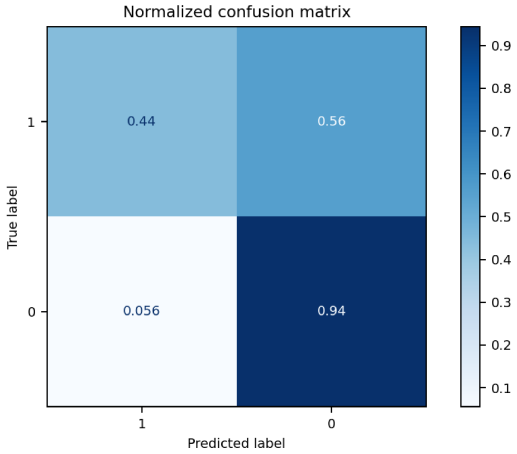
Figura 5.7: Confronto delle matrici di confusione per i corsi triennali



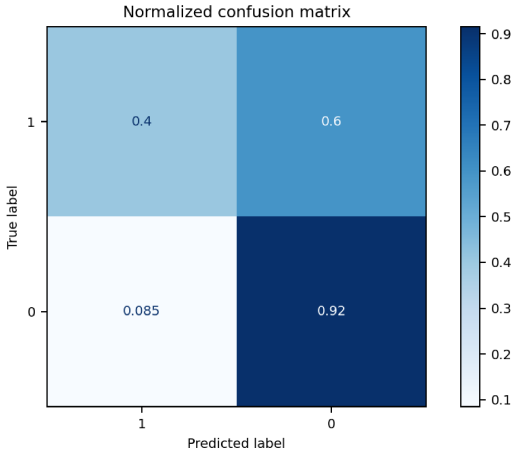
(a) XGBoost



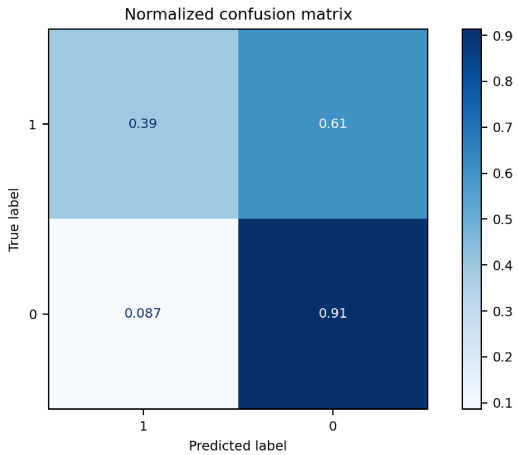
(b) Logistic regression



(c) LightGBM

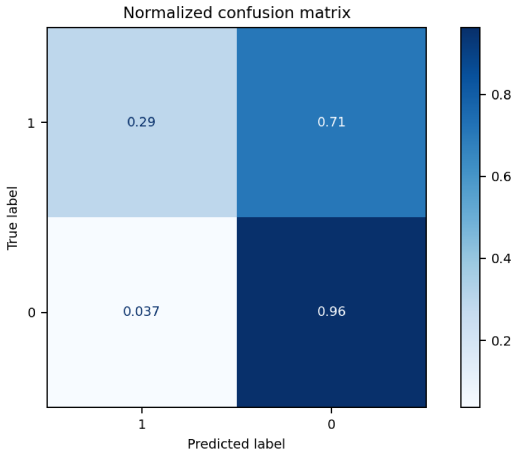


(d) Decision tree

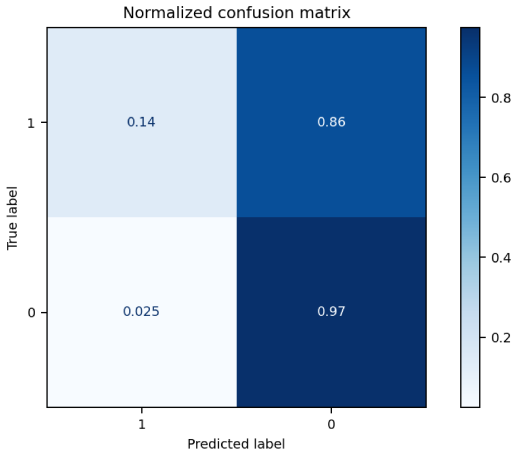


(e) Random forest

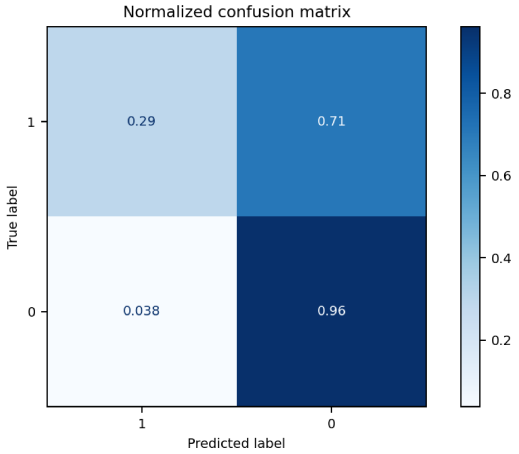
Figura 5.8: Confronto delle matrici di confusione per i corsi a ciclo unico



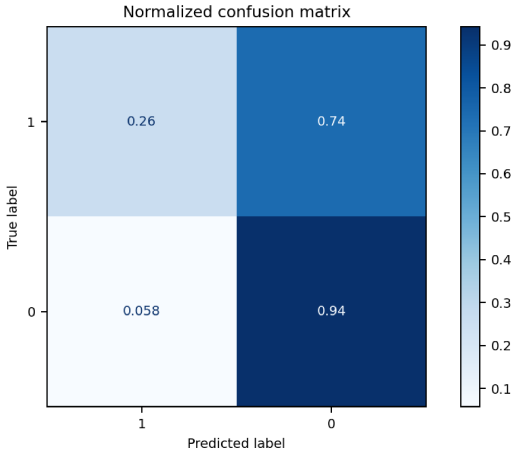
(a) XGBoost



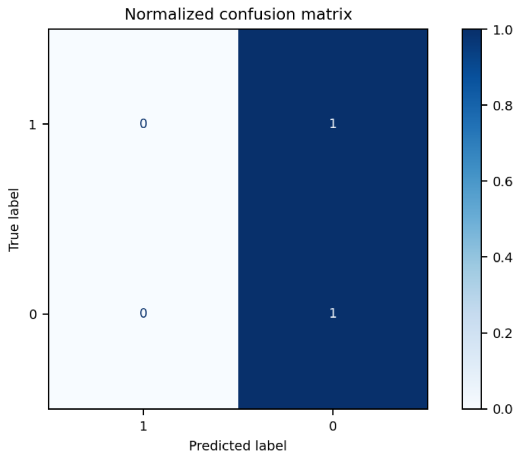
(b) Logistic regression



(c) LightGBM



(d) Decision tree



(e) Random forest

Figura 5.9: Confronto delle matrici di confusione per i corsi magistrali

5.3 Overfitting

Come visto nella Sezione 5.2, il modello usato per i corsi di laurea magistrale e a ciclo unico soffre del problema dell'overfittig; questo è un problema del machine learning supervisionato, Ying [2019], consiste nell'incapacità del modello di generalizzare bene i dati osservati. A causa dell'overfitting, il modello ottiene prestazioni perfette sul set di addestramento, ma scarse sul set di test; ciò avviene perché il modello che "overfitta" ha difficoltà a gestire le informazioni nel set di test che sono diverse da quelle del set di addestramento. Inoltre, i modelli affetti da questo problema tendono a memorizzare tutti i dati, incluso il rumore inevitabile nel set di addestramento, invece di imparare la logica nascosta dietro i dati. Le cause di questo fenomeno possono essere complesse; generalmente, possiamo classificarle in tre categorie:

- *Apprendimento del rumore dal set di addestramento*: quando il set di addestramento è troppo piccolo, contiene dati poco rappresentativi o troppo rumore; questa situazione fa sì che il rumore venga appreso con maggiore probabilità, e successivamente utilizzato come base per le previsioni. Pertanto, un algoritmo ben funzionante dovrebbe essere in grado di distinguere i dati rappresentativi dal rumore.
- *Complessità dell'ipotesi*: il compromesso in termini di complessità, un concetto chiave in statistica e machine learning, è un equilibrio tra varianza e bias, e si riferisce a un bilanciamento tra accuratezza e consistenza. Quando gli algoritmi hanno troppe ipotesi (troppi input), il modello diventa in media più accurato con una minore consistenza; ciò significa che i modelli possono essere drasticamente diversi su dataset differenti.
- *Procedure di confronto multiplo*: durante questi processi, si confrontano sempre più elementi in base ai punteggi di una funzione di valutazione e si seleziona l'elemento con il punteggio massimo. Tuttavia, questo processo potrebbe selezionare elementi che non migliorano, o addirittura riducono, l'accuratezza della classificazione.

I modelli precedenti sono affetti da overfitting a causa delle presenza di classi sbilanciate. La presenza di classi rare, in un dataset sbilanciato, può essere vista come una forma di rumore; questo perché le informazioni sulle classi rare sono spesso incomplete o poco affidabili. Gli algoritmi di machine learning possono imparare questo rumore e utilizzarlo per fare previsioni errate; ciò accade perché gli algoritmi tendono a concentrarsi sulle classi più frequenti, ignorando le classi rare; infatti, nei corsi di laurea a ciclo unico e magistrali si aveva la distribuzione delle classi riportata nella Tabella 5.4 e nelle Figure 5.4 e 5.5. Come si nota, nei corsi a ciclo unico c'era uno sbilanciamento delle classi; i casi positivi risultavano essere quasi il doppio dei negativi, ma si mantenevano comunque nello stesso ordine di grandezza. Nelle magistrali, invece, il numero degli elementi nella classe positiva risulta essere circa quattro volte il numero degli elementi negativi; quindi in questo caso si ha uno sbilanciamento molto elevato, infatti, come si è visto nella Sezione 5.2, anche le matrici di confusione risultano molto più sbilanciate.

	Ciclo Unico	**Magistrale**
0	3405	15262
1	1738	3708

Tabella 5.4: Distribuzione delle classi nel dataset

Uno degli approcci usati per risolvere questo problema è la tecnica del resampling che rende le classi bilanciate, Mohammed *et al.* [2020]. Le tecniche di resampling sono due, ovvero:

- *undersampling*: questa tecnica riduce il numero di elementi della classe maggioritaria;
- *oversampling*: questa tecnica è usata per aumentare il numero di elementi della classe minoritaria, questo può essere fatto producendo nuove istanze o ripetendole.

Inizialmente, si era optato per l'uso di entrambe le tecniche, si era cioè aumentata la classe minoritaria e diminuito il numero di elementi nella classe maggioritaria; questo approccio, però, non aveva portato a grandi miglioramenti; così, dopo una serie di prove, si è deciso di usare solo l'undersampling, lasciando invariato il numero di elementi nella classe minoritaria.

Per effettuare l'undersampling viene usata la funzione `RandomUnderSampler` della libreria Python `imblearn` che effettua il resampling eliminando gli elementi in modo casuale; questo rientra tra gli algoritmi che selezionano gli elementi a partire da un insieme principale S, generando un dataset S', dove S' è un sottoinsieme di S. La funzione va a ridurre il numero di elementi in base a quanto indicato dall'utente; questo risulta questo essere un metodo veloce e semplice per bilanciare le classi selezionando in modo casuale un sottoinsieme di dati della classe target.

Per il dataset dei corsi a ciclo unico è bastato applicare l'undersampling, impostando il parametro `sampling_strategy` a 0.9; questo significa che si vuole ridurre la classe di maggioranza in modo che rappresenti solo il 90% della classe di minoranza. In altre parole, si vuole creare un dataset bilanciato in cui la classe di minoranza rappresenta il 90% della classe di maggioranza, ottenendo la distribuzione di classi riportata nella Tabella 5.5.

	0	**1**
Prima	3405	1738
Dopo	1931	1738

Tabella 5.5: Distribuzione delle classi nei corsi ciclo unico dopo undersampling

Questo è bastato per ottenere dei risultati di classificazione migliori sopratutto nella matrice di confusione nella Figura 5.10, dove si è passati da predirre bene solo lo 0.52 dei casi negativi a predirne correttamente lo 0.74. Il numero di elementi della classe 0 predetti correttamente è sceso, passando da 0.91 a 0.79; ciò è dovuto al fatto che è diminuito il numero di elementi positivi che il classificatore usa per effettuare il training, ma resta un buon compromesso.

Per i corsi di laurea magistrale non è bastato il resampling a causa dello sbilanciamento delle classi che risultava essere molto più grande. In questo caso, è stato aggiunto alla rete un peso che permesse di dare più importanza alla classe di minoranza. Per la funzione di `RandomUnderSampler` è stato impostato un valore di `sampling_strategy` pari a 0.7, ottenendo la distribuzione delle classi riportata nella Tabella 5.7. Non è stato impostato un parametro più alto altrimenti il dataset sarebbe diventato troppo piccolo e si sarebbero ottenute prestazioni peggiori. Per i pesi il modello XGBoost prevede il parametro `scale_pos_weight` che è stato impostato come il rapporto tra il numero di campioni nella classe di maggioranza e il numero di campioni nella classe di minoranza dopo il resampling. In questo modo si dà più peso alla classe minoritaria durante l'addestramento del modello, aiutando, così, a migliorare le prestazioni della classe di minoranza.

Al termine del nuovo addestramento si è ottenuto un modello che riesce a riconoscere molto meglio la classe negativa, peggiorando, però, sia le prestazioni sul riconoscimento della classe positiva, che passa da 0.96 a 0.73, sia le metriche; questo si può dedurre dalla Tabella 5.8 e dalla Figura 5.11.

	Accuracy	Precision	Recall	F1
Prima	0,78	0,77	0,91	0,84
Dopo	0,76	0,83	0,79	0,81

Tabella 5.6: Valori delle metriche per i corsi a ciclo unico dopo l'undersampling

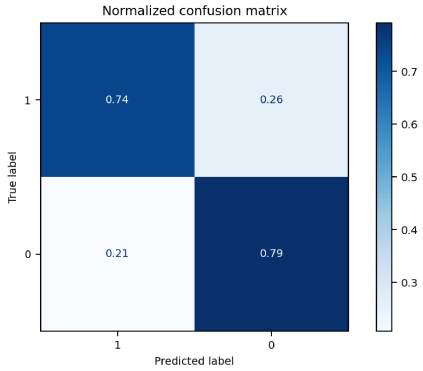


Figura 5.10: Matrice di confusione dei corsi a ciclo unico dopo l'undersampling

	0	1
Prima	15262	3709
Dopo	5298	3709

Tabella 5.7: Distribuzione delle classi nei corsi magistrali dopo l'undersampling

	Accuracy	Precision	Recall	F1
Prima	0,84	0,85	0,96	0,90
Dopo	0,73	0,92	0,73	0,81

Tabella 5.8: Valori delle metriche per i corsi magistrali dopo l'undersampling

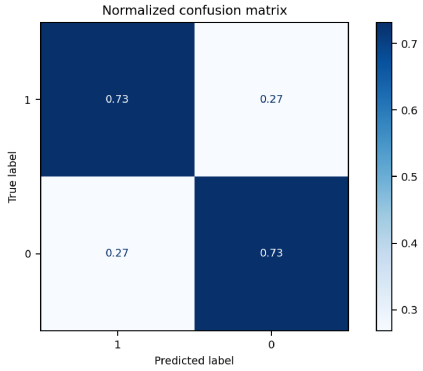


Figura 5.11: Matrice di confusione dei corsi magistrali dopo l'undersampling

CAPITOLO 6

Risultati ottenuti e loro commento

In questo capitolo saranno approfondite le performance dei modelli di classificazione introdotti nel capitolo precedente. Verranno analizzate le loro capacità predittive e le feature che sono state usate per classificare correttamente i dati.

6.1 Risultati

Dopo aver deciso di dividere il dataset in base ai diversi tipi di corsi di laurea (triennali, magistrali e a ciclo unico) è stato eseguito il training dei vari modelli per ciascuna suddivisione del dataset. Tutti i modelli selezionati per l'addestramento sono classificatori ad albero; In particolare, essi sono l'albero decisionale, il random forest, la regressione logistica, XGBoost e LightGBM. Infine, è stato scelto il modello che presentava le migliori metriche di valutazione; nel dettaglio, per tutti i dataset, il modello migliore è risultato essere XGBoost, un potente algoritmo di machine learning basato su Gradient Boosting.

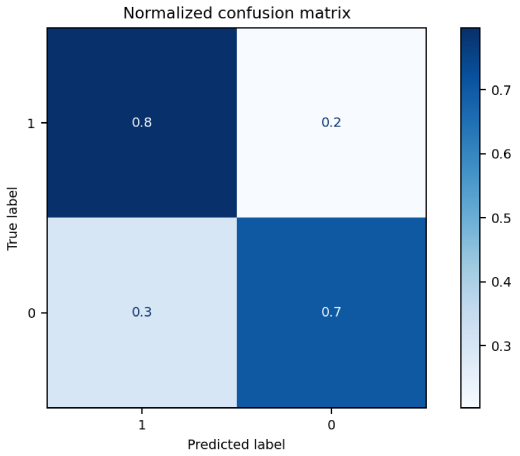
Per il dataset delle lauree triennali, effettuando il training dal modello XGBoost, si sono ottenuti subito dei buoni risultati con delle bune metriche ed una buona matrice di confusione; questo risultato è dovuto al fatto che il dataset aveva una suddivisione delle classi abbastanza bilanciata. In questo caso è bastato salvare il modello XGBoost su cui è stato effettuato il training senza apportare alcuna modifica.

Anche per i dataset dei corsi di laurea a ciclo unico ed i corsi di laurea magistrale, il miglior modello è risultato essere XGBoost. Tuttavia, a differenza del dataset precedente, questi modelli hanno evidenziato una distribuzione sbilanciata delle classi, soprattutto nel caso dei corsi di laurea magistrale. Nonostante le metriche sembrassero promettenti, le matrici di confusione hanno rivelato una classificazione completamente errata. In particolare, entrambi i modelli hanno mostrato difficoltà nel classificare correttamente la classe minoritaria, corrispondente alla classe 1 che rappresenta gli studenti che abbandonano il corso di laurea. Al contrario, la classe 0 è stata classificata con precisione elevata, risultando essere la classe maggioritaria. Questa problematica è comunemente nota come overfitting; per contrastarla, sono state impiegate tecniche di undersampling in entrambi i casi. Questo approccio mira a ridurre gli elementi presenti nella classe maggioritaria, in modo da avvicinare il numero di elementi a quelli della classe minoritaria. Inoltre, nel caso del modello relativo alle lauree magistrali, lo squilibrio tra le classi era così marcato che l'utilizzo dell'undersampling da solo non è stato sufficiente. È risultato, quindi, necessario aggiungere dei pesi al modello, per conferire maggiore importanza alla classe minoritaria e consentire una migliore classificazione.

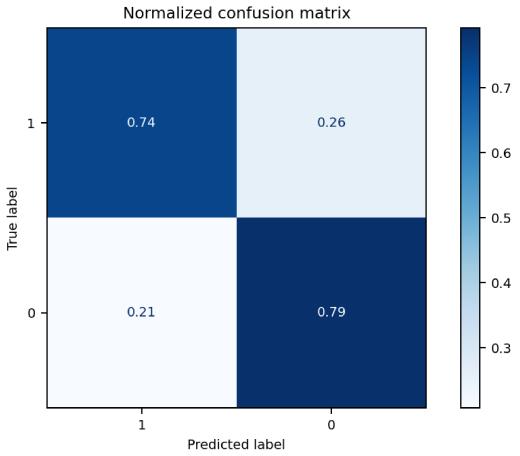
Dopo aver apportato queste correzioni, le matrici di confusione hanno evidenziato un miglioramento, sebbene a scapito delle metriche che hanno registrato un leggero peggioramento. Tuttavia, è stato raggiunto un buon compromesso: il modello è ora in grado di classificare entrambe le classi con successo, come si può notare dalla Tabella 6.1 e dalla Figura 6.1.

Corsi di Laurea	Accuracy	Precision	Recall	F1-Score
Triennale	0,76	0,73	0,70	0,72
Ciclo Unico	0,76	0,83	0,79	0,81
Magistrale	0,73	0,92	0,73	0,81

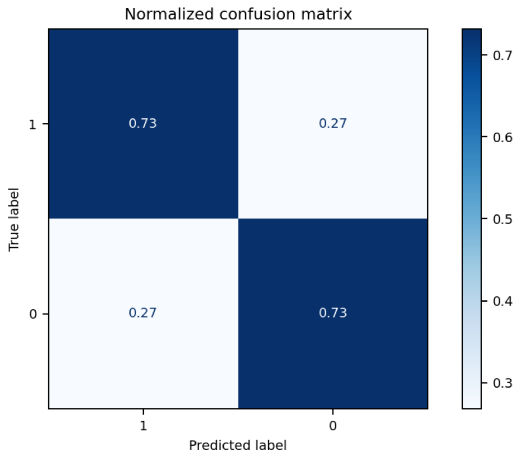
Tabella 6.1: Riepilogo delle metriche per i diversi corsi di laurea



(a) Matrice di confusione dei corsi di laurea triennali



(b) Matrice di confusione dei corsi di laurea a ciclo unico

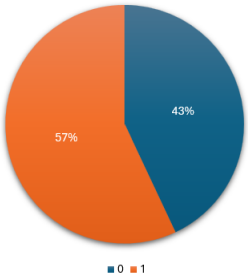


(c) Matrice di confusione dei corsi di laurea magistrale

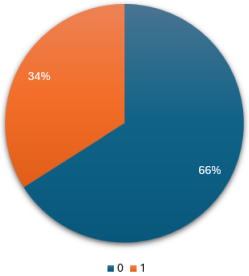
Figura 6.1: Riepilogo delle matrici di confusione per i diversi corsi di laurea

Dopo aver ottenuto un modello per ogni tipologia di corso di laurea, sono stati creati i dataset su cui effettuare le predizioni. Per la creazione di tali dataset sono stati presi in considerazione tutti gli studenti che ancora non hanno concluso il loro percorso di studi, sono stati divisi in base al corso di laurea e poi sono stati dati in input ai corrispondenti modelli. I risultati ottenuti sono molto simili a quelli avuti con i dataset di training; infatti,

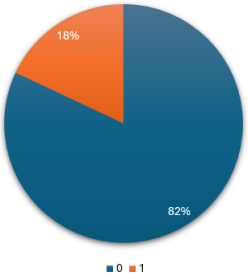
confrontando la Figura 6.2 con le Figure 5.3, 5.4 e 5.5, si nota una distribuzione delle classi del tutto simile, con un maggior numero di elementi della classe 1 per i corsi di laurea triennale, e quindi si avrà un maggior numero di ritiri rispetto a chi riesce a completare con successo il proprio percorso di studi, ed una situazione completamente opposta per gli altri due corsi di laurea in cui gli elementi della classe 0 rappresentano la netta maggioranza, soprattutto nei corsi di laurea magistrale in cui rappresentano più dell'80%.



(a) Distribuzione delle classi per i corsi di laurea triennale



(b) Distribuzione delle classi per i corsi di laurea a ciclo unico



(c) Distribuzione delle classi per i corsi di laurea magistrale

Figura 6.2: Distribuzione delle classi per i diversi corsi di laurea

6.2 Feature importance

Dopo aver eseguito le previsioni, risulta interessante condurre un'analisi per comprendere su quali feature i tre modelli si sono concentrati maggiormente per effettuare la classificazione. Per eseguire questa analisi è stata usata la libreria Python `SHAP` (SHapley Additive exPlanations), questa è una tecnica potente che sfrutta la teoria dei giochi per spiegare come funzionano i modelli di machine learning; essa offre una rappresentazione visiva completa, riassumendo le connessioni tra le feature e l'output del modello.

`SHAP` non è in grado di interpretare modelli che gestiscono dati con valori mancanti (null). Se il dataset contiene valori nulli, sia i dati di background (utilizzati per addestrare il modello) sia i campioni scelti per la spiegazione saranno soggetti a imputazione utilizzando la moda (i valori più frequenti). Questo processo di imputazione può introdurre potenziali distorsioni nei valori `SHAP` calcolati, poiché i campioni imputati potrebbero non riflettere accuratamente la reale distribuzione dei dati.

Con questa libreria si può ottenere un grafico beeswarm progettato per mostrare in modo denso ed informativo come le feature principali di un dataset influenzano l'output del modello. Ogni istanza è rappresentata da un singolo punto su una riga, che a sua volta indica una feature. La posizione sull'asse x del punto è determinata dal valore `SHAP` di quella feature, ed i punti si "accumulano" su ogni riga per mostrare la densità. Il colore viene utilizzato per visualizzare il valore originale di una feature.

I colori principalmente usati sono il rosso e il blu, dove con il rosso si indicano valori elevati per una certa feature, mentre con il colore blu vengono indicati valori bassi di una certa feature.

Analizzando il grafo beeswarm dei corsi di laurea triennale in Figura 6.3, si può dedurre come l'attributo più importante per la classificazione sembra essere quello che indica le tasse pagate dagli studenti, rappresentato dall'attributo "pagato"; questo attributo ha influenzato soprattutto la scelta di abbandono; infatti si osserva una predominanza di punti rossi sull'asse negativo; indicando che valori elevati della feature che rappresenta il contributo pagato dagli studenti tendono fortemente ad essere associati all'abbandono. D'altro canto, si notano molti punti blu sull'asse positivo, indicanti valori bassi di contributo, suggerendo che importi

minori contribuiscano positivamente al percorso accademico degli studenti. Altra feature importante per il modello risultata essere il voto del diploma, voto_dipl; contrariamente alle aspettative comuni, si osserva che valori elevati del voto di maturità non necessariamente portano gli studenti a completare il corso di studi, come indicato dai numerosi punti rossi. Infine, un'altra feature di rilevanza significativa risulta essere l'età dell'iscrizione al corso di studi. Dall'analisi dei punti blu emerge che coloro che si iscrivono immediatamente a una laurea triennale spesso non completano il percorso accademico. È importante, tuttavia, considerare che la maggior parte degli studenti si iscrive alla laurea triennale appena concluso il percorso scolastico; pertanto, il numero maggiore di questi studenti potrebbe contribuire alla maggior probabilità di abbandono riscontrata; a supporto di questa affermazione, si può fare riferimento alla Figura 6.4, che illustra la distribuzione dell'età degli studenti iscritti alla laurea triennale in giorni.

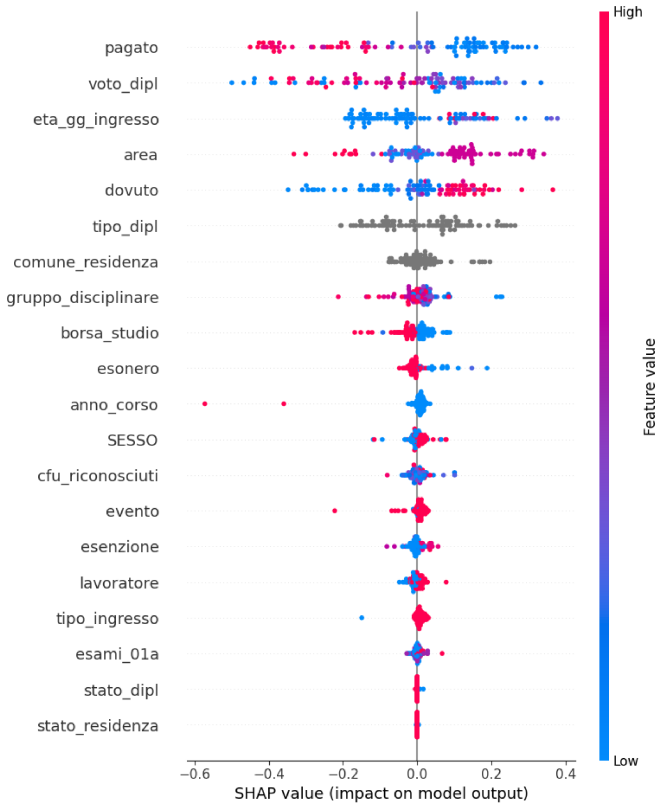


Figura 6.3: Beeswarm per i corsi di laurea triennale

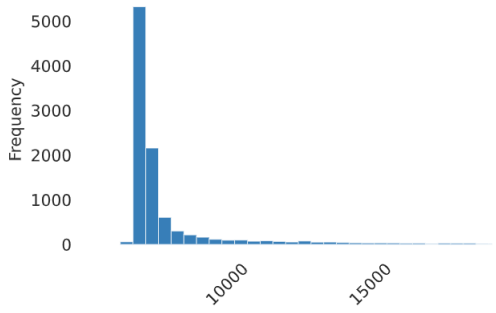


Figura 6.4: Distribuzione degli studenti per età calcolata in giorni per i corsi di laurea triennale

La stessa analisi è stata effettuata sui corsi di laurea a ciclo unico; in questo caso, come si nota dalla Figura 6.5, sono state rilevanti le stesse feature del modello dei corsi triennali, ma con un ordine differente. Per i corsi di laurea a ciclo unico, a influire maggiormente è stato il voto di diploma; anche in questo caso un voto alto non coincide con un'alta probabilità di esito positivo del percorso di studi universitario. La seconda feature è rappresentata dalle tasse pagate dagli studenti; anche per gli studenti dei corsi di laurea a ciclo unico, elevate tasse pagate portano ad un'elevata probabilità di abbandono. Ed infine, tra le feature utilizzate dal modello per la classificazione, l'età di iscrizione dello studente al corso di laurea è ancora presente. In questo contesto, emerge chiaramente che un'età di ingresso inferiore sembra essere correlata ad un esito negativo rispetto a un'età di iscrizione più avanzata, anche per questo corso di studi; però, l'età in cui si hanno maggiori iscrizioni è attorno ai 19/20 anni, cioè subito dopo il termine delle scuole superiori, come illustrato dalla distribuzione in Figura 6.6.

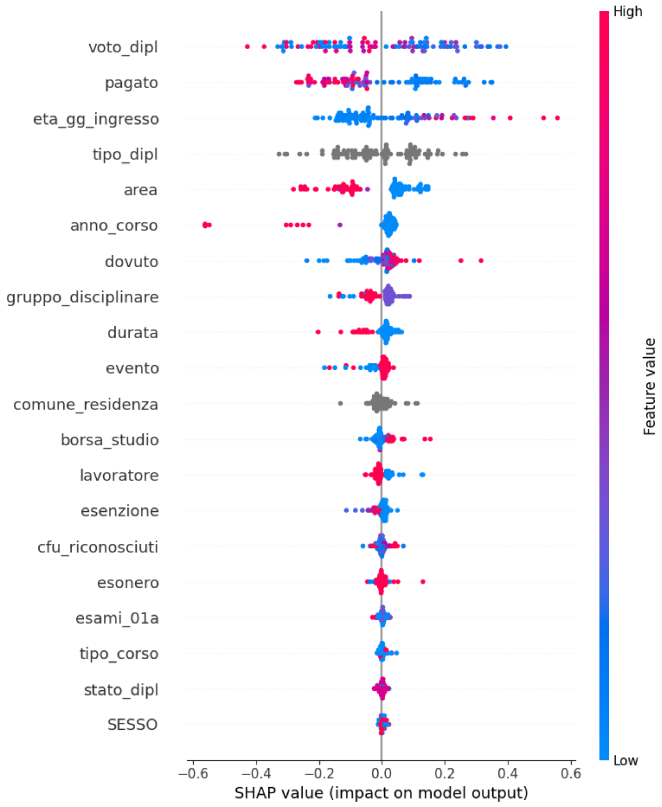


Figura 6.5: Beeswarm per i corsi di laurea a ciclo unico

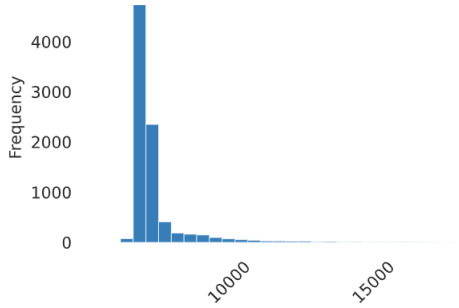


Figura 6.6: Distribuzione degli studenti per età calcolata in giorni per i corsi di laurea a ciclo unico

Infine, sono stati presi in considerazione i corsi di laurea magistrale. In Figura 6.7 si nota che, per il modello di questo tipo di corso, l'attributo più significativo è stato l'età di iscrizione al corso di studi dello studente, poiché per i corsi di laurea magistrale si osserva una distribuzione più ampia delle età di iscrizione, come evidenziato nella Figura 6.8; mentre per i corsi di laurea triennale e a ciclo unico, gli studenti si iscrivono principalmente all'età di 19 anni, per i corsi di laurea magistrale l'età di iscrizione varia considerevolmente. La seconda feature è ancora una volta l'ammontare di tasse pagato, dove un elevato importo comporta una maggior probabilità di abbandono. In questo contesto, al voto di diploma non viene data una particolare enfasi, avendo gli studenti già conseguito una laurea triennale.

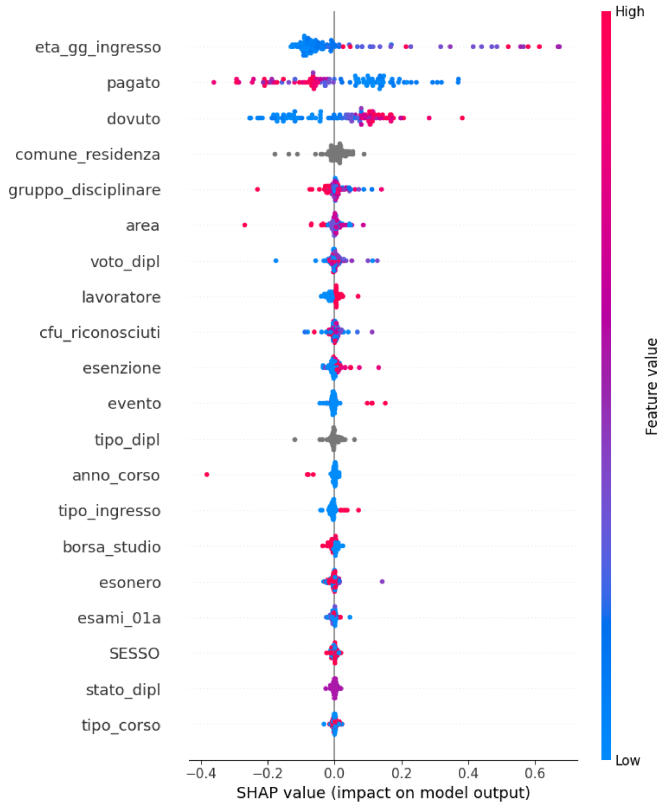


Figura 6.7: Beeswarm per i corsi di laurea magistrale

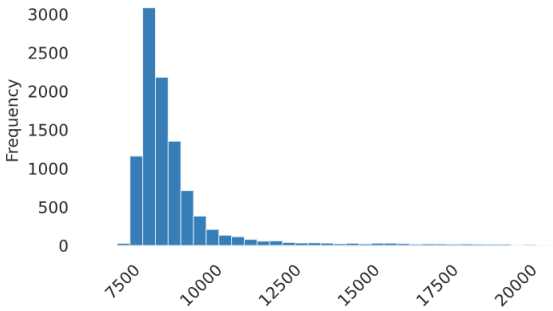


Figura 6.8: Distribuzione degli studenti per età calcolata in giorni per i corsi di laurea magistrale

CAPITOLO 7

Conclusioni

In questa tesi è stato studiato il fenomeno dell'abbandono universitario; tale problema è molto presente in Italia, sopratutto nei corsi di laurea triennale, e porta a una serie di problemi sia dal punto di vista economico sia dal punto di vista del capitale umano. Le perdite economiche si hanno sia per le famiglie degli studenti, che si ritrovano a sostenere i costi di un percorso formativo che non viene completato, sia per lo Stato, che investe risorse economiche per studenti che non faranno parte della forza lavoro qualificata. Nel dettaglio è stato studiato il fenomeno dell'abbandono nell'Università di Pisa. L'obbiettivo è stato quello di raccogliere i dati per creare dei modelli di classificazione che riescano a prevedere quali studenti molto probabilmente abbandoneranno il corso di laurea e quali porteranno a termine il proprio percorso di studi, in modo tale che l'università possa mettere in atto delle politiche correttive per arginare il fenomeno dell'abbandono. Per dare inizio allo studio, è stata intrapresa un'accurata raccolta di dati da diverse fonti; nello specifico in collaborazione con l'Università, è stato ottenuto accesso a un prezioso database contenente informazioni pertinenti al campo di ricerca. Tali dati sono stati integrati con informazioni reperite dal sito web dell'Osservatorio del CINECA; per quest'ultimo è stato usato uno script Python che permette di scaricare i dati periodicamente in modo che questi siano sempre aggiornati.

Dopo aver scaricato i dati, la seconda fase del progetto è stata quella dell'ETL; questa è stata effettuata attraverso Data Factory, un servizio cloud che permette di creare flussi di lavoro per il trasferimento e la trasformazione di informazioni su vasta scala. Nello specifico è stata creata una pipeline principale chiamata `MAIN` all'interno della quale vengono eseguite in serie altre pipeline, tra cui quella per effettuare il caricamento dei dati, una per effettuarne la loro pulizia ed, infine, una per caricare le tabelle delle dimensioni ed una per le tabelle dei fatti. Dopo la fase di ETL è stata effettuata la creazione di metadati, indispensabili per la creazione del dataset finale, che servirà per effettuare le previsioni.

Nella fase successiva dello studio si è passati ad analizzare quali modelli utilizzare per effettuare la classificazione; per questo problema sono stati scelti modelli di classificazione ad albero, e sono state stabilite le metriche da utilizzare per valutare i risultati ottenuti. Per effettuare il training dei modelli di classificazione è stato usato Databricks. Si è deciso di dividere il dataset in tre parti, in base al tipo di corso di laurea; si è ottenuto un dataset per i corsi di laurea triennali, uno per i corsi a ciclo unico ed uno per quelli magistrali. Per tutti, il modello migliore è risultato essere XGBoost, anche se per i corsi di laurea a ciclo unico e magistrale, che risultavano avere una distribuzione delle classi sbilanciata, è stato effettuato l'undersampling.

Infine, si è effettuato uno studio delle feature usate dai modelli per effettuare la classificazione. Da tale analisi sembrerebbe che un attributo fondamentale per tutti i modelli sia stato l'ammontare delle tasse pagate dagli studenti, mentre per i corsi di laurea triennale e a ciclo unico, ha giocato un ruolo fondamentale anche il voto di diploma degli studenti; al contrario di quello che ci si aspettava, un voto alto di diploma spesso non coincide con un'alta probabilità di concludere gli studi. Sulla base dei risultati ottenuti si potrebbero dedurre diverse politiche per cercare di evitare l'abbandono da parte degli studenti del proprio corso di studi, tra cui:

- *Aumentare le borse di studio e gli aiuti finanziari*: sarebbe fondamentale ampliare l'accesso a sostegni economici per studenti meritevoli o provenienti da famiglie con redditi bassi. Questo permetterebbe di ridurre il peso finanziario dell'università e incoraggiare la partecipazione di tutti.
- *Introdurre un sistema di prestiti a tassi d'interesse agevolati*: per coloro che non possono permettersi il pagamento delle tasse universitarie in anticipo, un sistema di prestiti accessibili potrebbe facilitare l'accesso all'istruzione.
- *Offrire sconti o esoneri agli studenti meritevoli*: premiare gli studenti con voti di diploma alti con riduzioni o esenzioni dalle tasse universitarie potrebbe incentivare l'impegno.
- *Potenziare i servizi di orientamento scolastico e universitario*: fornire agli studenti informazioni precise e complete sulle diverse opzioni di studio e sulle opportunità professionali può aiutarli a fare scelte consapevoli e ridurre il rischio di errori.
- *Organizzare iniziative di sensibilizzazione e informazione*: promuovendo l'importanza dell'istruzione universitaria e le sue ricadute positive sulla vita individuale e professionale potrebbe incentivare i giovani a proseguire gli studi.
- *Promuovere l'inclusione e la diversità*: attivare misure di sostegno per studenti con disabilità o provenienti da contesti svantaggiati può garantire pari opportunità a tutti.

Sarebbe nell'interesse comune delle istituzioni, dell'Università e degli studenti tutti, perseguire nella lotta all'abbandono universitario, sfruttando i modelli predittivi elaborati e qui illustrati, oltre a promuovere ed adottare politiche sociali ed economiche che favoriscano gli studenti e li sostengano nelle situazioni più disagiate. Si auspica, quindi, che questo studio sperimentale possa fornire idee e suggerimenti per affrontare un problema così serio e grave del nostro Paese, in un ottica di continuo miglioramento delle politiche sociali universitarie.

Bibliografia

AIZED AMIN SOOFI, A. A. (2017), «Classification Techniques in Machine Learning: Applications and Issues», *Journal of Basic Applied Sciences*. (Cited at page 26)

ANVUR (2023), «Rapporto sul sitema di istruzione superiore e della ricerca», *Agenzia Nazionale di Valutazione del Sistema Universitario e della Ricerca*. (Cited at pages 1 e 4)

AZIZ, B. M. P. S., R.M. (2022), «LGBM: a machine learning approach for Ethereum fraud detection.», p. 3321–3331, URL `https://doi.org/10.1007/s41870-022-00864-6`. (Cited at page 32)

BECKER (2001), «Why Don't Italians Finish University? Explaining Enrollment Behaviour in Italy and Germany, European University Institute», . (Cited at page 6)

BITTENCOURT, H. e CLARKE, R. (2003), «Use of classification and regression trees (CART) to classify remotely-sensed digital images», in «IGARSS 2003. 2003 IEEE International Geoscience and Remote Sensing Symposium. Proceedings (IEEE Cat. No.03CH37477)», vol. 6, p. 3751–3753 vol.6. (Cited at page 29)

BURGALASSI, C. M., BIASI (2016), «Il fenomeno dell'abbandono universitario precoce. Uno studio di caso sui corsi di laurea del Dipartimento di Scienze della Formazione dell'Università "Roma Tre"», *Giornale Italiano della Ricerca Educativa*. (Cited at page 6)

CHARBUTY, B. e ABDULAZEEZ, A. (2021), «Classification Based on Decision Tree Algorithm for Machine Learning», *Journal of Applied Science and Technology Trends. 2, 01 (Mar. 2021)*. (Cited at page 29)

CHEN, T. e GUESTRIN, C. (2016), «XGBoost: A Scalable Tree Boosting System», in «Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining», KDD '16, p. 785–794, Association for Computing Machinery, New York, NY, USA, URL `https://doi.org/10.1145/2939672.2939785`. (Cited at page 31)

FILIPPO BELLOC, A. M. e PETRELLA, L. (2010), «University drop-out: an Italian experience», *Springer*. (Cited at page 5)

LARA GITTOA, L. M., LEO FULVIO MINERVINI (2016), «University dropouts in Italy: Are supply side characteristics part of the problem?», *Economic Analysis and Policy*. (Cited at page 6)

MARCO DELOGU, D. P. G. R., RAFFAELE LAGRAVINESE (2023), «Predicting dropout from higher education: Evidence from Italy», *Economic Modelling*. (Cited at page 3)

MOHAMMED, R., RAWASHDEH, J. e ABDULLAH, M. (2020), «Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results», in «2020 11th International Conference on Information and Communication Systems (ICICS)», p. 243–248. (Cited at page 43)

SPEISER, J. L., MILLER, M. E., TOOZE, J. e IP, E. (2019), «A comparison of random forest variable selection methods for classification prediction modeling», *Expert Systems with Applications*, vol. 134, p. 93–101, URL https://www.sciencedirect.com/science/article/pii/S0957417419303574. (Cited at page 30)

YING, X. (2019), «An Overview of Overfitting and its Solutions», *Journal of Physics: Conference Series*, vol. 1168. (Cited at page 43)

Siti web consultati

- learn.microsoft.com, data factory – https://learn.microsoft.com/it-it/azure/data-factory
- bigdata4innovation.it – https://www.bigdata4innovation.it/big-data/databricks/
- learn.microsoft.com, Databricks – https://learn.microsoft.com/it-it/azure/databricks/machine-learning/automl/how-automl-works
- scikit-learn.org – https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
- imbalanced-learn.org – https://imbalanced-learn.org/dev/references
- SHARP – https://github.com/shap/shap
- sharp.readtheadocs – https://shap.readthedocs.io/en/latest/example_notebooks/api_examples/plots/beeswarm.html
- Wikipedia – www.wikipedia.org

Ringraziamenti

Per prima cosa, vorrei ringraziare il mio relatore il Professore Domenico Ursino per la sua pazienza e disponibilità, riuscendo sempre a dedicare del tempo per rispondere alle mie domande e fornirmi un supporto prezioso.

Non posso non ringraziare le due persone che hanno avuto più influenza nel mio percorso educativo: i miei genitori. Mamma e papà, desidero ringraziarvi di cuore per il vostro costante sostegno, sia dal punto di vista economico che emotivo. Grazie a voi, ho trovato la forza necessaria per superare i momenti più difficili lungo questo cammino. Senza il vostro amore e il vostro appoggio incondizionato, non avrei mai potuto raggiungere questo importante traguardo. Siete stati la mia roccia e la mia fonte di ispirazione, e per questo vi sarò eternamente grata.

Un grande grazie va alla mia fantastica sorellina Chiara! Sei stata la mia compagna di avventure e di risate in ogni momento. Grazie per tutte le volte che hai tirato fuori le tue perle di saggezza e hai creato le tue magiche ricette culinarie per darmi una svolta quando ne avevo bisogno. Senza di te, il mio percorso sarebbe stato molto meno divertente e saporito!

Un ringraziamento speciale va a David che mi ha accolto nella grande famiglia di Advant, mi ha aiutato nell'intero lavoro di progetto di tesi e sostenuto in ogni momento di difficoltà lavorativa e non, continuando ancora adesso ad essere un punto di riferimento fondamentale nel mio percorso. Ringrazio Angelo per avermi aiutato durante i momenti più duri, la tua capacità di darmi consigli pratici e conforto è stata davvero preziosa. Ringrazio anche i miei colleghi Lorenzo e Gianluca, con cui ho iniziato il mio percorso lavorativo, su di loro so di poter sempre contare.

Un ringraziamento va a Maria una persona speciale con cui, nonostante la distanza, siamo sempre riuscite a mantenere vivo il nostro legame e a non perderci di vista, grazie per essere riuscita a restarmi vicina anche quando le circostanze sembravano renderlo difficile.

Un grande grazie va a Michele per il suo sostegno e il suo amore durante il mio percorso accademico. Grazie per essere sempre stato al mio fianco, incoraggiandomi e sostenendomi in ogni fase di questa avventura. La tua presenza mi ha dato la forza e la determinazione necessarie per superare le sfide e raggiungere i miei obiettivi.

Un piccolo ringraziamento alle due pesti Roger e Ariel.