



Predicting Student Dropout

Visualizzazione del dataset e training di un
modello predittivo sul successo accademico

32460A - CRISTIAN CASTELLANO
20631A - ALESSANDRA CAVALLI

Dataset

Il dataset contiene dati provenienti da vari corsi di laurea del Politecnico di Portalegre (IPP), tra gli anni accademici 2008/09 e 2018/19.

Ogni **istanza** corrisponde ad uno **studente**.

È stato preso come riferimento il seguente paper:

M.V.Martins, D. Tolledo, J. Machado, L. M.T.

Baptista, V.Realinho. (2021) "Early prediction of student's performance in higher education: a case study"

Dataset e paper sono visualizzabili scansionando il QR code.

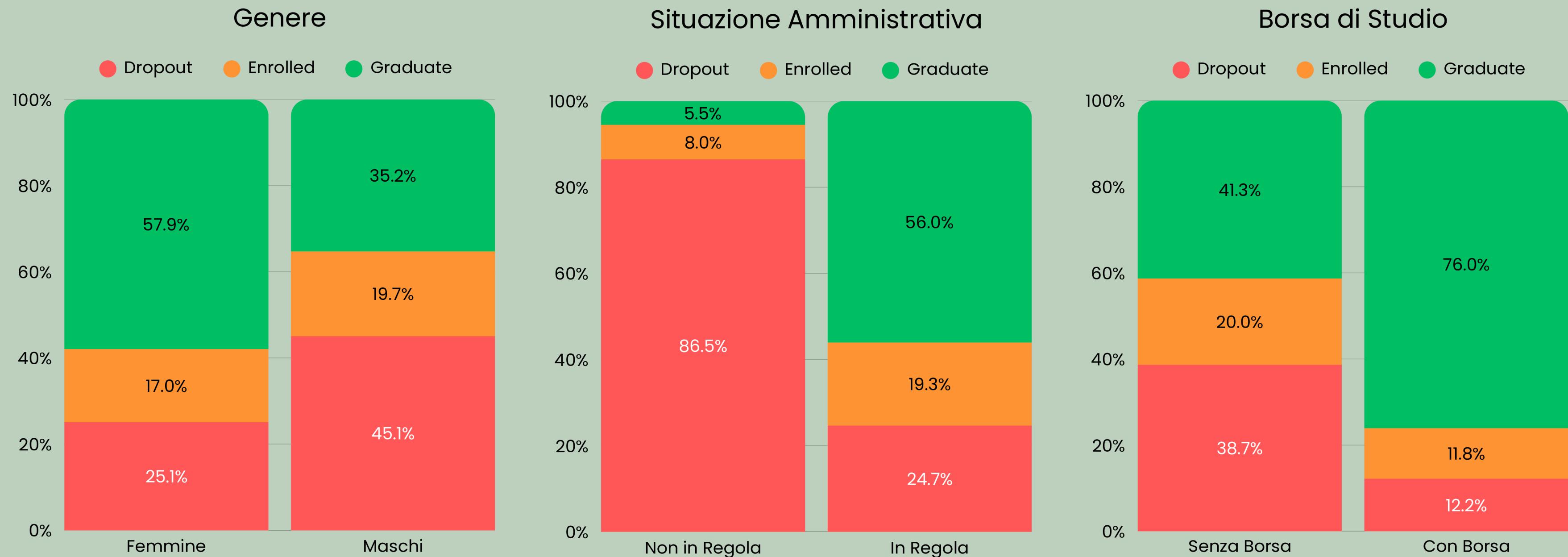


Le istanze sono descritte mediante **36 features** raggruppabili in 8 categorie:

- fattori demografici
- percorso accademico precedente
- informazioni sul corso di laurea
- background familiare
- situazione economica
- indicatori economici
- performance del primo semestre
- performance del secondo semestre

Gli studenti vengono quindi classificati tramite **3 classi target**: successo, successo relativo e insuccesso.

Alcune visualizzazioni

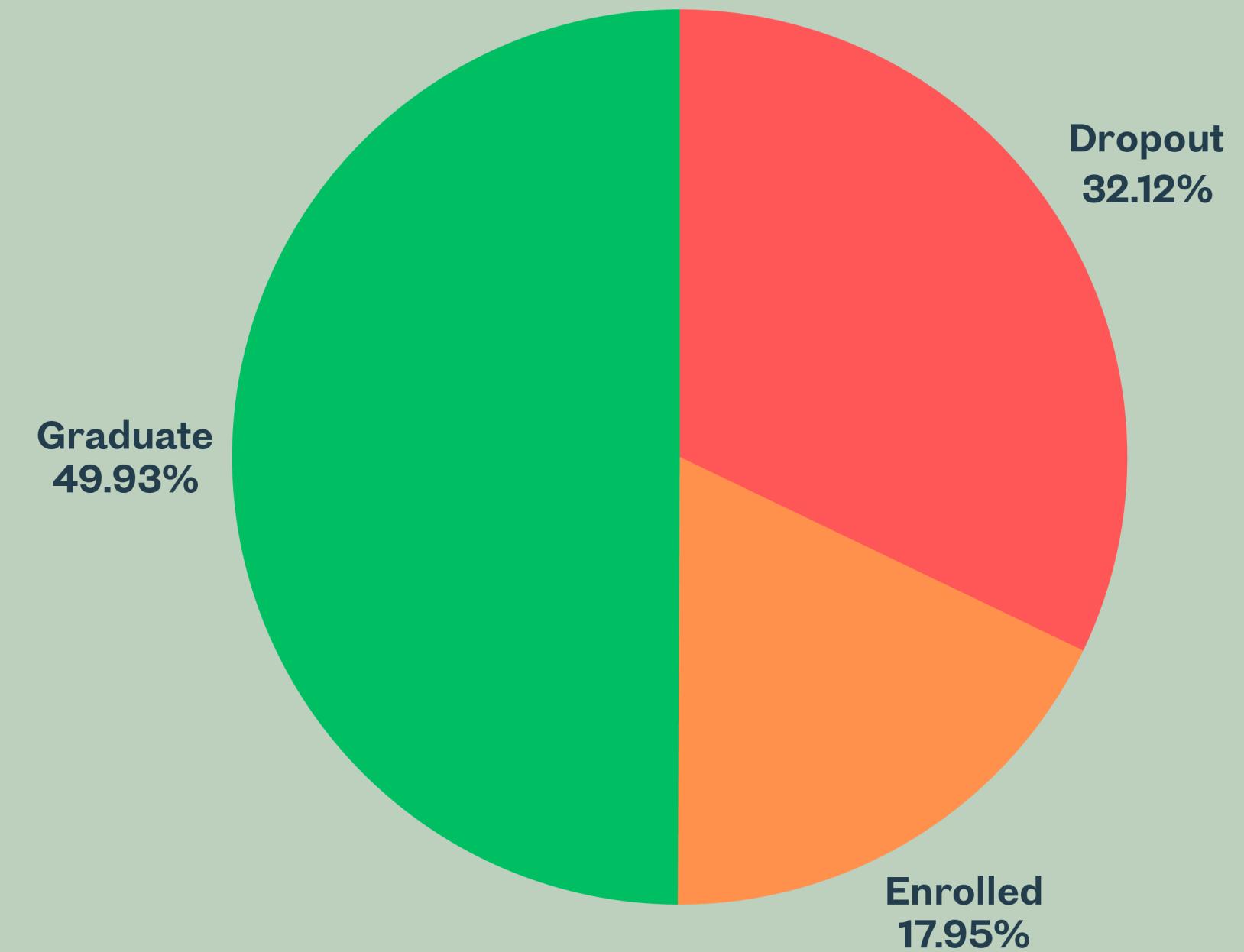


Criticità del Dataset

Ogni record è stato classificato come **Successo** (lo studente ha ottenuto la laurea in tempo), **Successo Relativo** (lo studente ha impiegato fino a 3 anni extra per ottenere la laurea) o **Insuccesso** (lo studente ha impiegato più di tre anni extra per ottenere la laurea o non l'ha ottenuta affatto).

La **distribuzione** dei record tra le tre categorie è **sbilanciata**, con due classi di minoranza.

Le classi che miriamo maggiormente a identificare correttamente sono quelle di minoranza, bisogna quindi superare il problema con **strategie di campionamento**.



Per approfondire



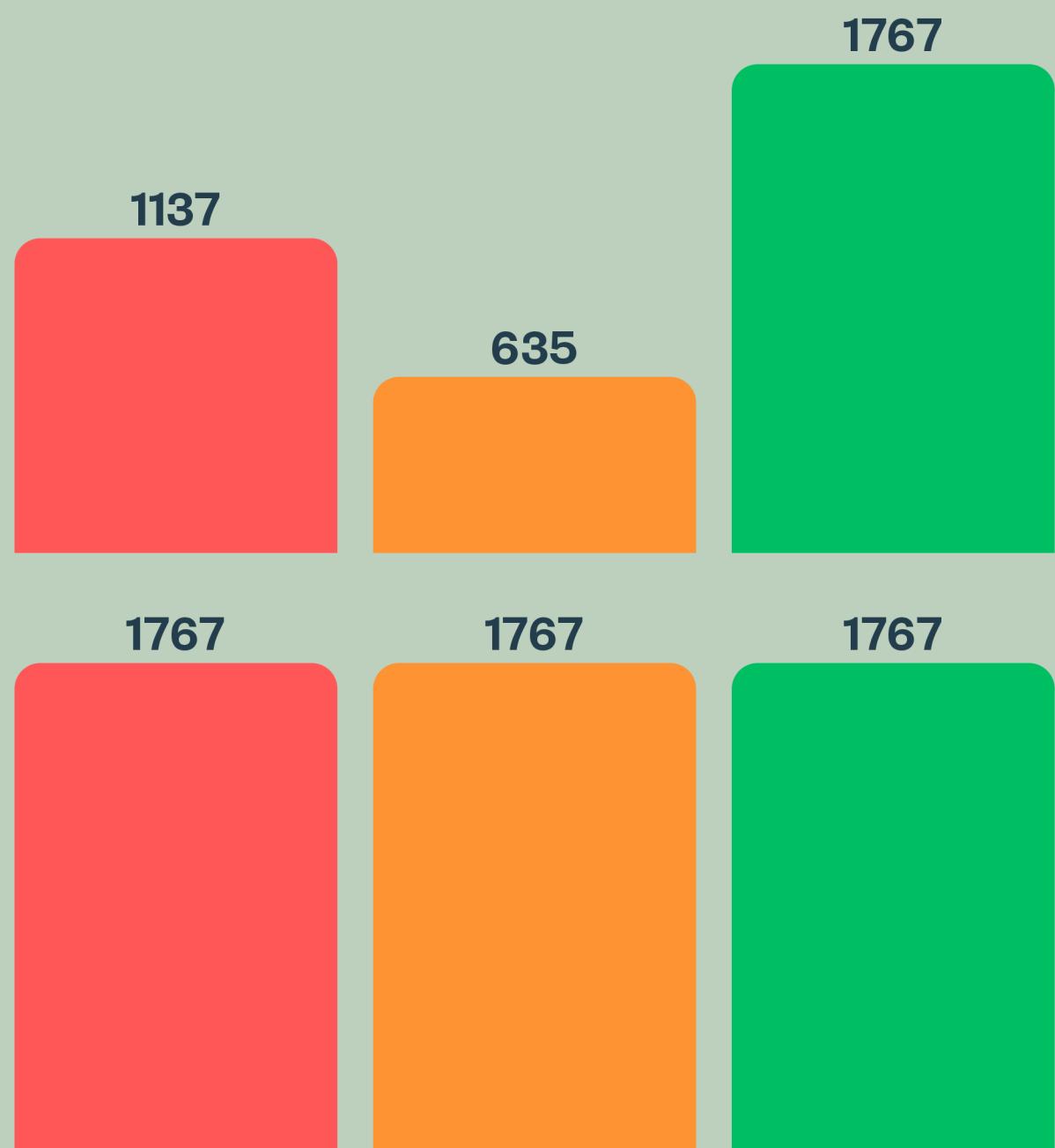
Campionamento Dati (SMOTE)

L'algoritmo SMOTE (*Synthetic Minority Over-sampling Technique*) funziona trovando campioni vicini dalla classe di minoranza nello spazio delle feature e sintetizzando un nuovo campione nello spazio tra i k vicini.

Questo algoritmo va applicato soltanto al **training set**, che corrisponde all'80% del dataset.

```
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline as ImbPipeline
pipeline = ImbPipeline([
    ('smote', SMOTE(k_neighbors=5, random_state=random_state)),
    ('classifier', Classifier(
        ...
    ))
])
```

● Dropout ● Enrolled ● Graduate



Scelta dei modelli

I modelli più performanti per questo tipo di dataset sono **Random Forest** e **Extreme Gradient Boosting**.

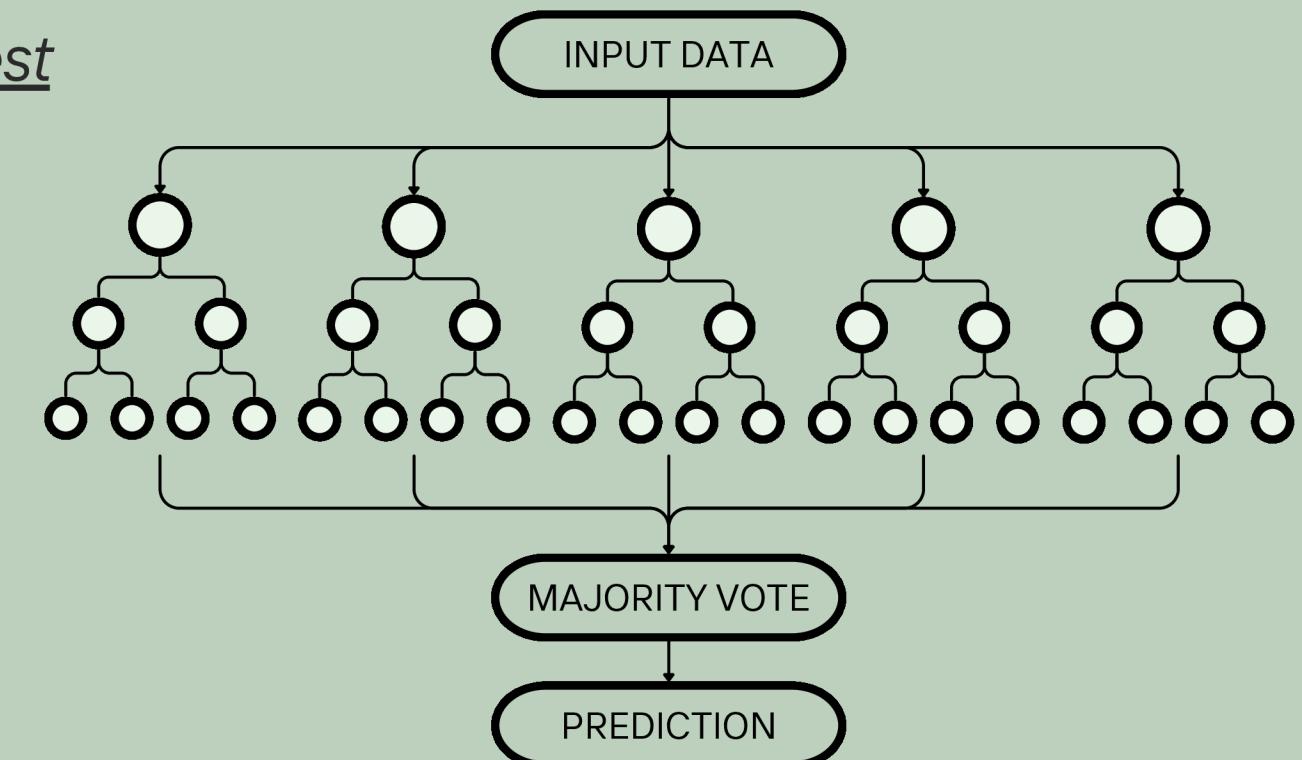
Abbiamo deciso di provarli entrambi per valutare quale fosse il migliore nel nostro caso.

In entrambi i casi si parla di **ensemble learning**, vengono quindi combinati molteplici modelli predittivi per ottenere prestazioni migliori.

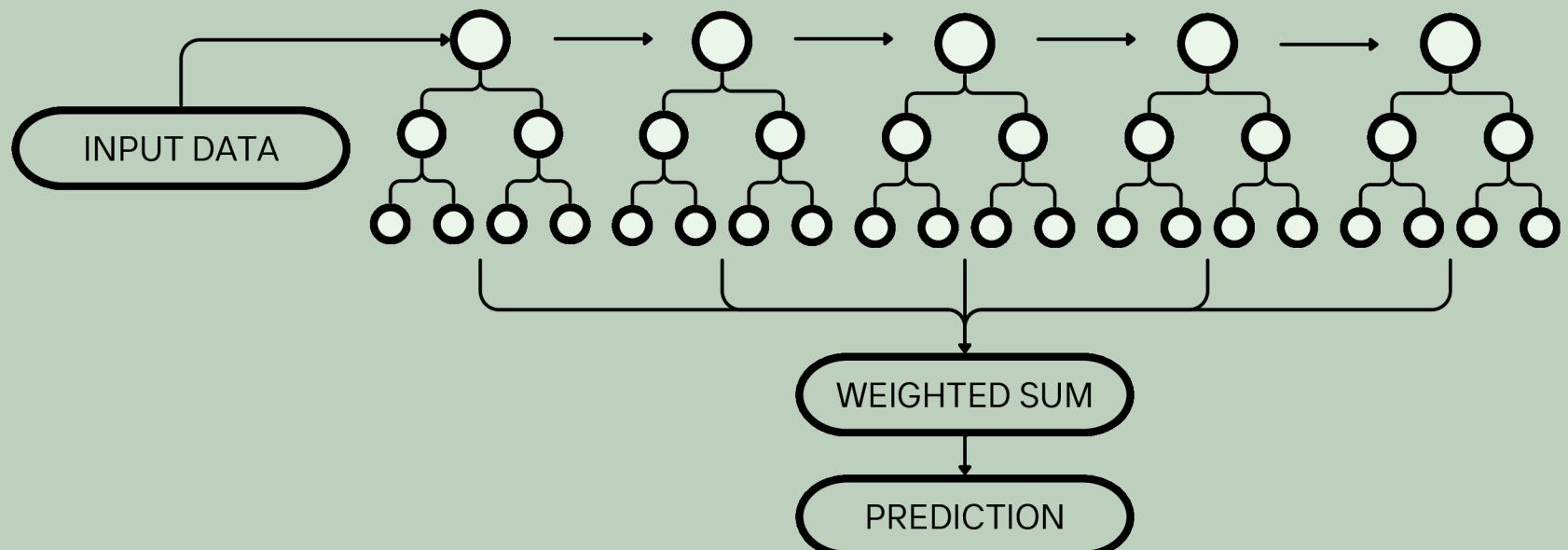
Le tecniche utilizzate sono tuttavia differenti:

- **Bagging** per Random Forest, in cui i modelli utilizzati hanno tutti la stessa importanza. L'output complessivo sarà il valore maggioritario.
- **Boosting** per Extreme Gradient Boosting, in cui i modelli vengono addestrati in sequenza, ognuno correggendo l'errore prodotto dal suo predecessore.

Random Forest



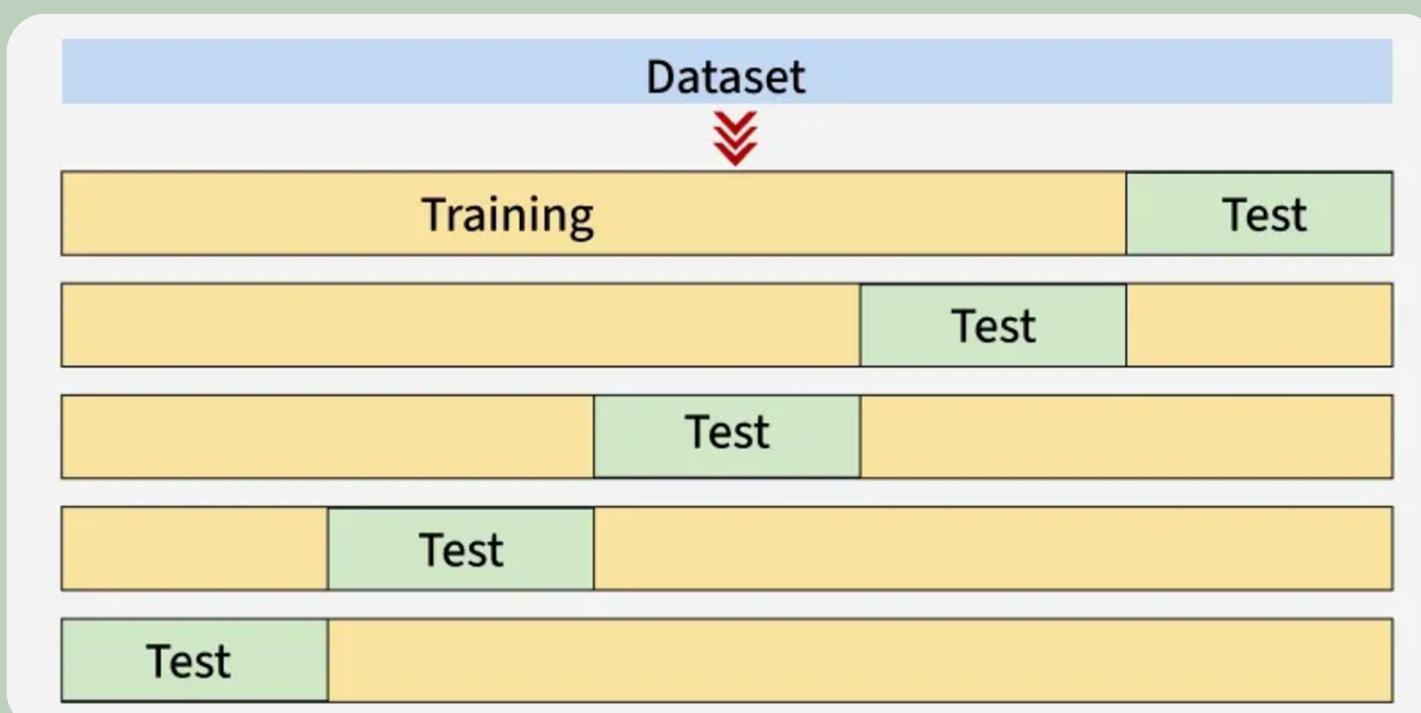
XGBoost



Training e Metriche

5-fold cross validation

Per evitare overfitting, i dati sono stati suddivisi in 5 blocchi. 4 blocchi sono stati utilizzati per il training, mentre il restante per il testing. Il processo è stato ripetuto 5 volte, usando ogni volta un blocco diverso per il testing.



Nel nostro caso, l'accuratezza non è la misura più adeguata per le prestazioni del modello, poiché è una metrica complessiva.

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

Dal momento che il dataset è sbilanciato, consideriamo l'F1-score per ogni categoria.

$$F_1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

$$Precision = \frac{TP}{TP + FP} \quad Recall = \frac{TP}{TP + FN}$$

Random Forest

Il modello è stato implementato tramite la libreria scikit-learn

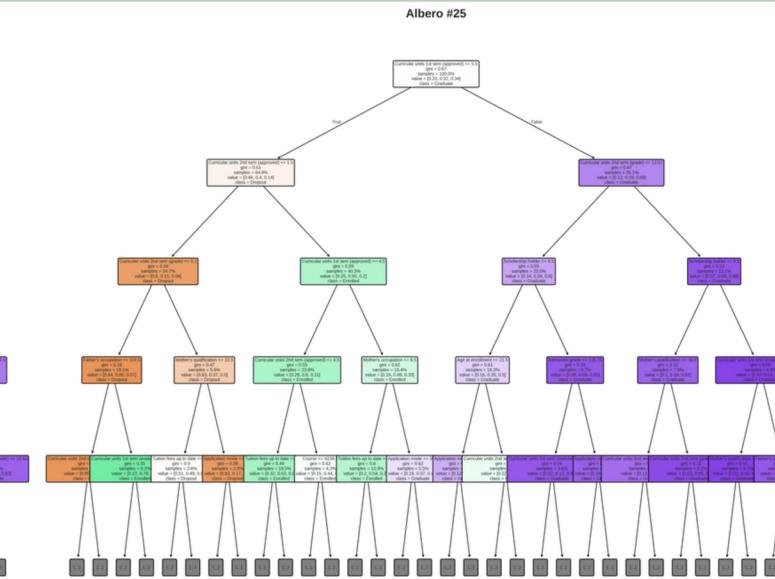
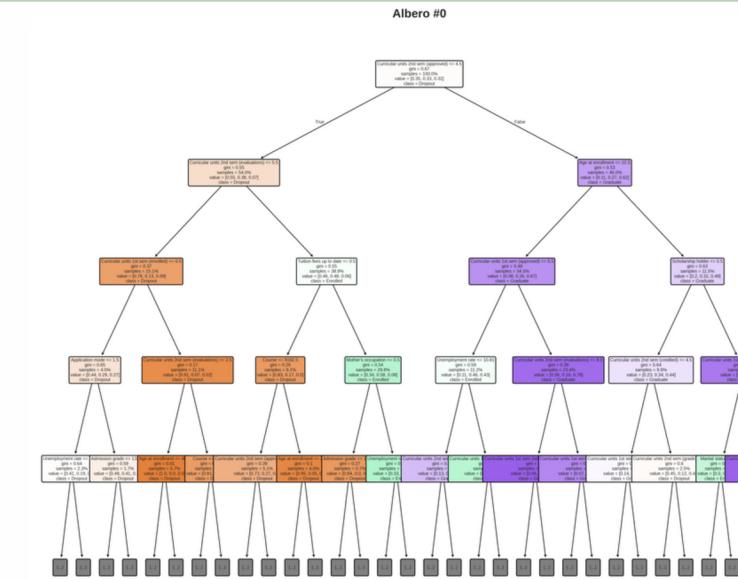
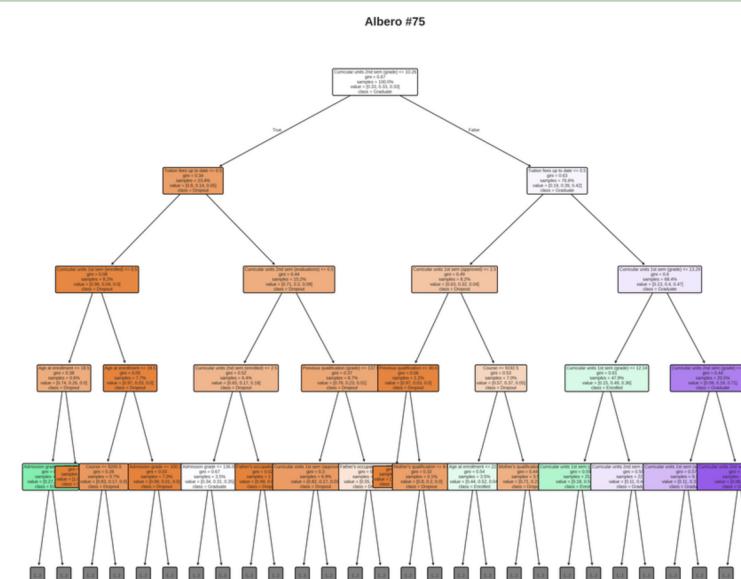
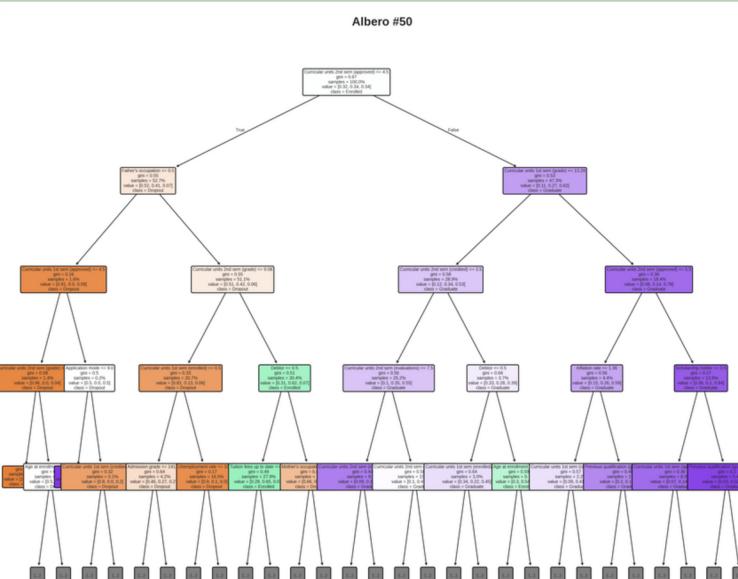
RandomForestClassifier() inizializza il classificatore:

- *n_estimators* indica il numero di alberi nel modello.
- *random_state* inizializza il seed per la generazione pseudocasuale.
- *n_jobs* indica il numero di core della cpu da dedicare

rf_base.fit() costruisce la foresta con il training set

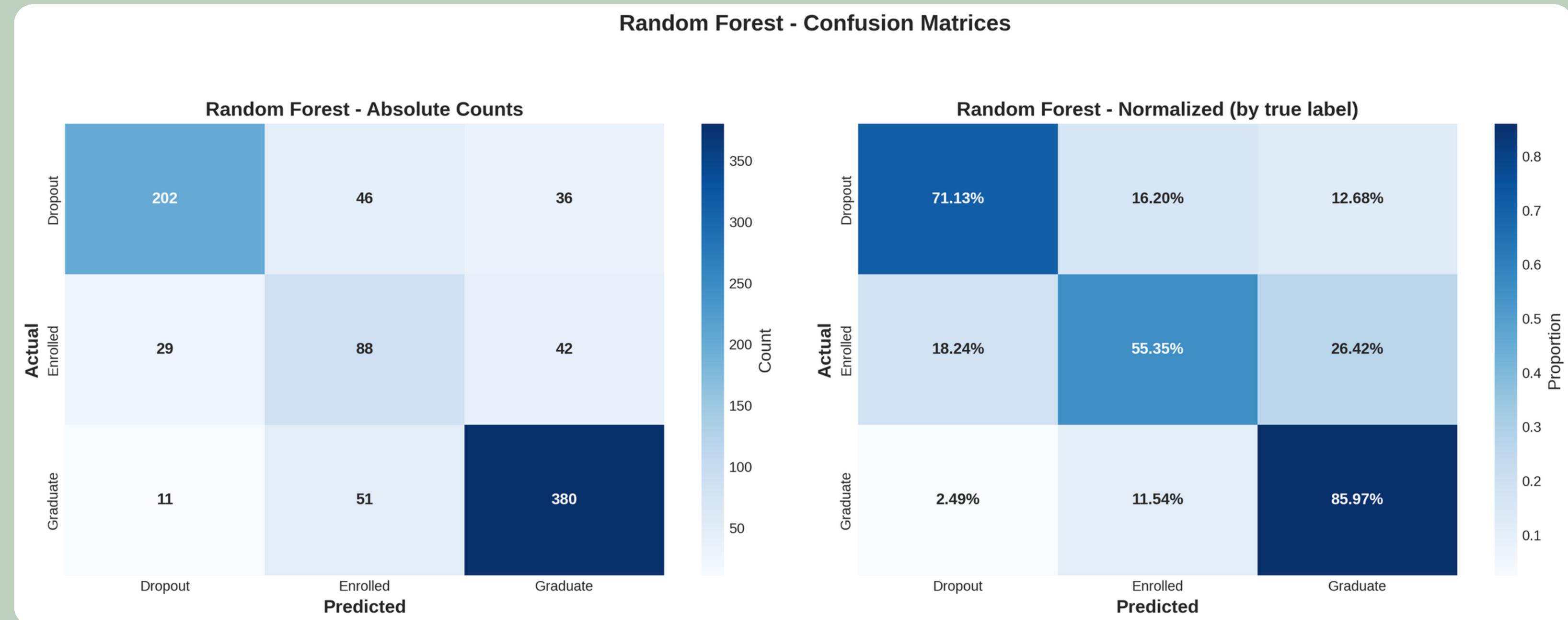
rf_base.predict() prende il test set e restituisce le predizioni

```
from sklearn.ensemble import RandomForestClassifier
rf_pipeline = ImbPipeline([
    ('smote', SMOTE(...)),
    ('classifier', RandomForestClassifier(
        n_estimators=100,
        random_state=random_state,
        n_jobs=-1
    ))
])
rf_pipeline.fit(X_train, y_train)
rf_test_pred = rf_pipeline.predict(X_test)
```



Campione di 4 alberi (profondità limitata per leggibilità)

Random Forest



Extreme Gradient Boosting

Il modello è stato implementato tramite la libreria xgboost

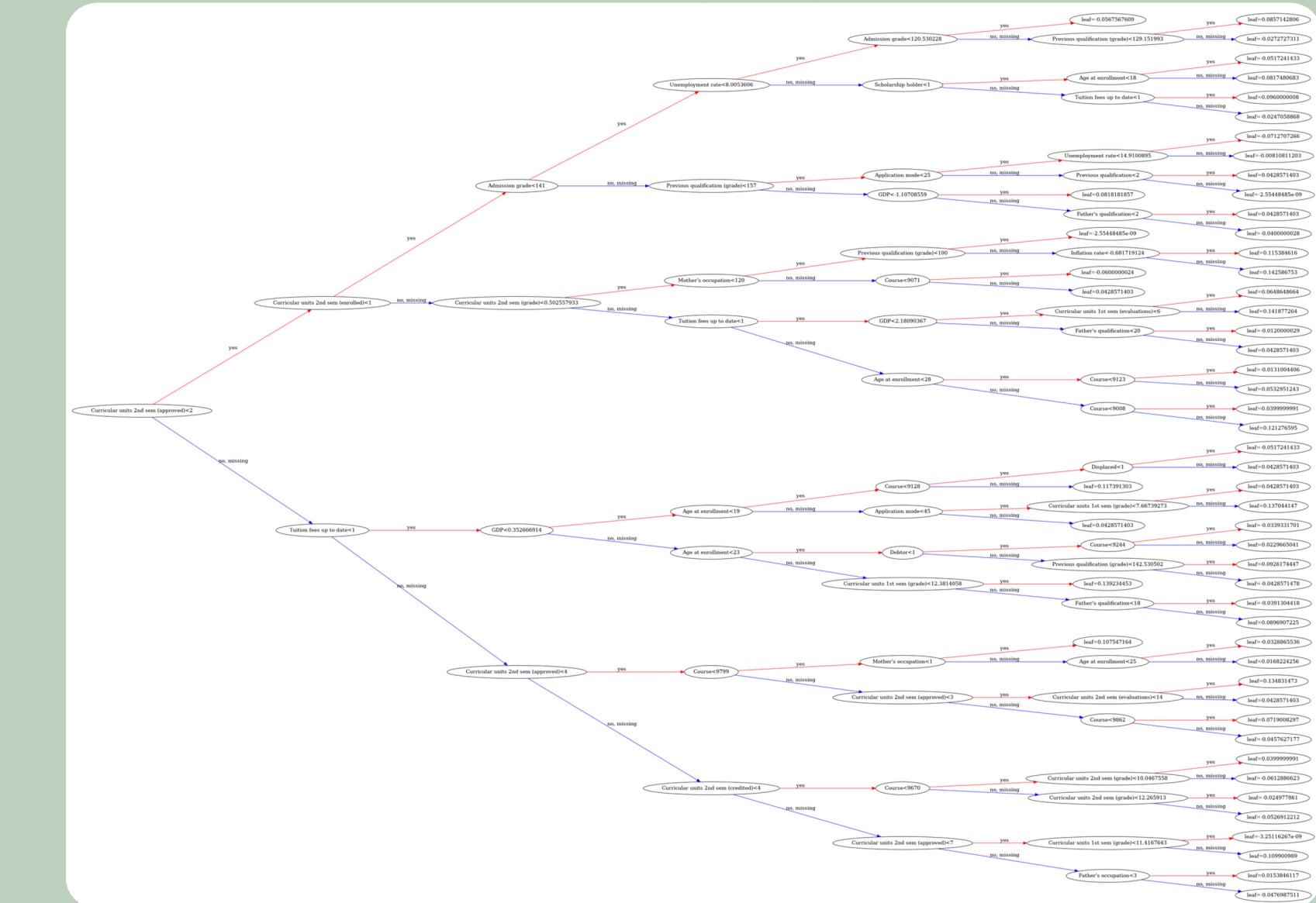
XGBClassifier() inizializza il classificatore:

- *learning_rate*: tasso d'apprendimento del modello
- *max_depth*: determina la profondità massima degli alberi
- *eval_metric*: metrica di valutazione della performance

xgb_base.fit() costruisce il modello con il training set

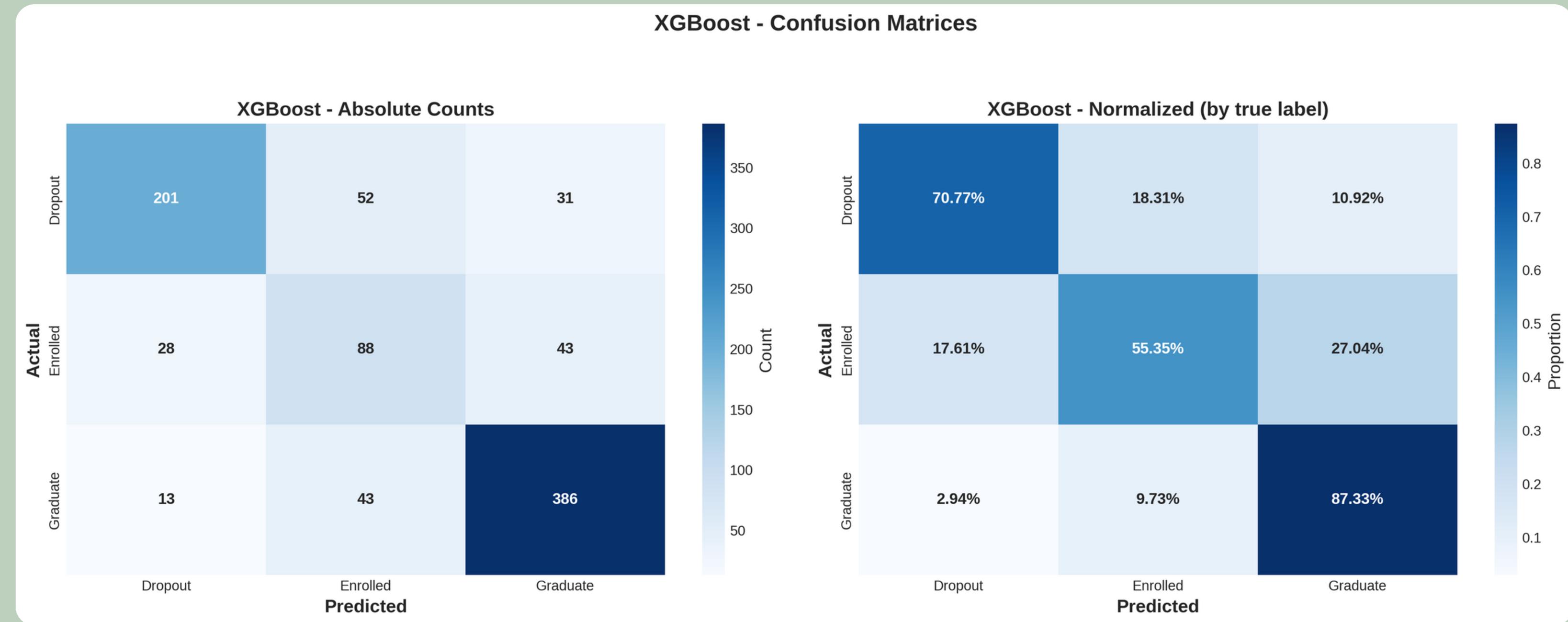
xgb_base.predict() restituisce le predizioni

```
from xgboost import XGBClassifier
xgb_pipeline = ImbPipeline([
    ('smote', SMOTE(...)),
    ('classifier', XGBClassifier(
        n_estimators=100,
        learning_rate=0.1,
        max_depth=6,
        random_state=random_state,
        eval_metric='mlogloss'
    ))
])
xgb_pipeline.fit(X_train, y_train_encoded)
xgb_test_pred_encoded = xgb_pipeline.predict(X_test)
```

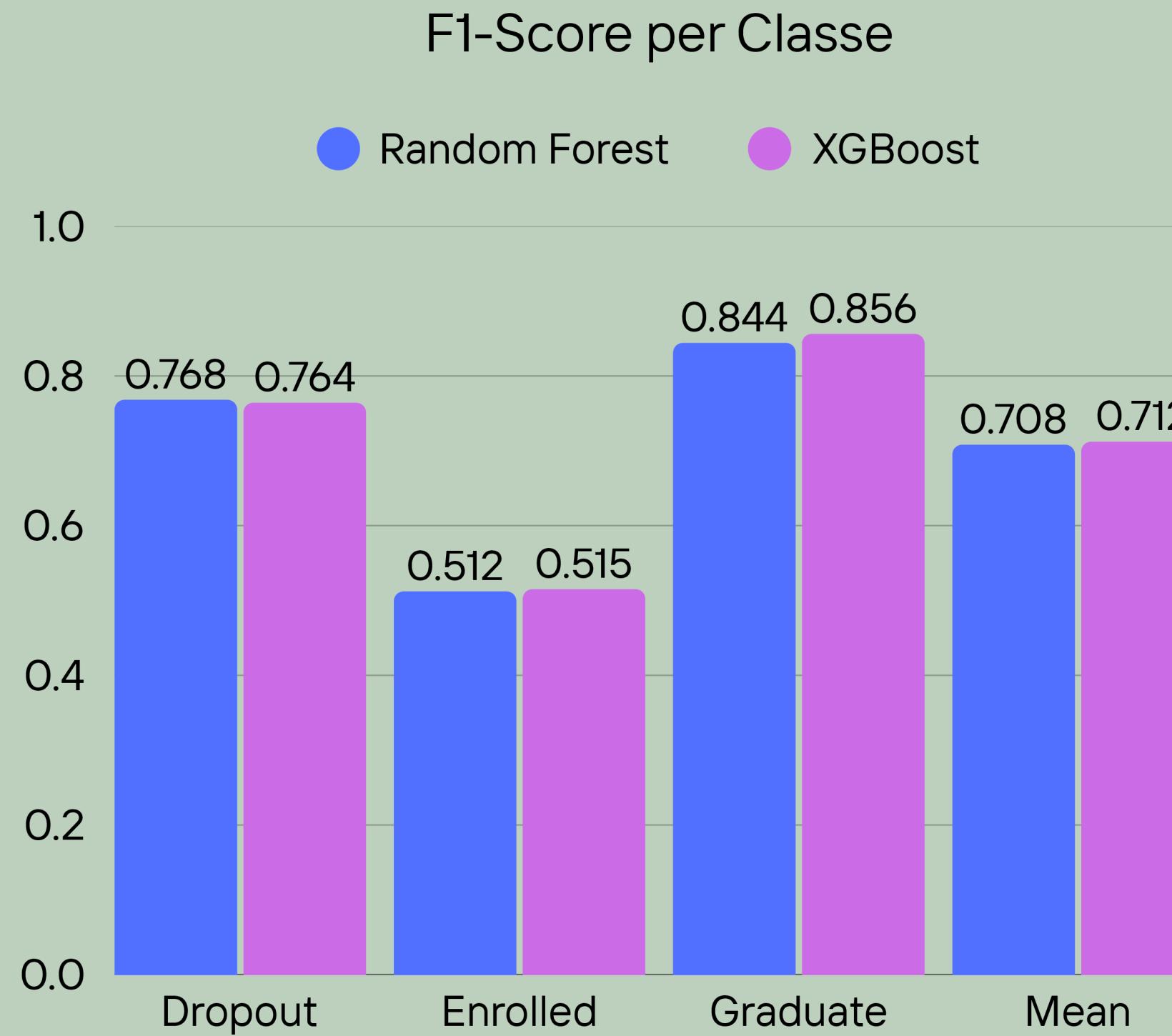


Albero #0 (Dettagliato)

Extreme Gradient Boosting



Confronto tra Modelli



Simulatore Studente

Selezione Modello

Quale modello vuoi usare?

XGBoost (BA: 71.15%) Random Forest (BA: 70.82%) Confronta Entrambi

Performance

Balanced Accuracy
71.15%

F1-Score
71.16%

Dati Studente

▼ Fattori Demografici

Età all'iscrizione	Genere	Fuori sede
20	- + Femmina	No
Stato civile	Nazionalità	
Single	Portuguese	

➤ Percorso Accademico Precedente

➤ Informazioni sul Corso di Laurea

➤ Background Familiare

➤ Situazione Economica

➤ Performance Universitaria - 1° Semestre

➤ Performance Universitaria - 2° Semestre

➤ Indicatori Economici

Simulatore Studente

Confronto Modelli

Random Forest

XGBoost

Graduate

Graduate

Confidenza

68.0%

Confidenza

80.4%

Classe	Probabilità
Dropout	10.0%
Enrolled	22.0%
Graduate	68.0%

Classe	Probabilità
Dropout	16.4%
Enrolled	3.2%
Graduate	80.4%