Capitolo 1

Risultati Sperimentali

1.1 Tecnologie utilizzate

Per tutti gli esperimenti è stato utilizzato il linguaggio Python 3.7.13 con l'utilizzo del tool *sckit-learn* che offre una serie di modelli di Machine Learning. Per quanto riguarda il bilanciamento dei dati è stata utilizzato il tool *imbalanced-learn*. Per l'esecuzione degli esperimenti è stato utilizzato invece Google Colaboratory nella sua versione gratuita.

1.2 Metriche di valutazione

Definiamo innanzitutto la nostra matrice di confusione che sarà così fatta:

Confusion matrix

	Positive	**Negative**
Positive	TP	FP
Negative	FN	TN

Definiamo a questo punto:

- **TP:** i valori dropout effettivamente riconosciuti come dropout dal modello
- **FP:** i valori non dropout riconosciuti come dropout dal modello
- **FN:** i valori dropout riconosciuti come non dropout dal modello
- **TN:** i valori non dropout effettivamente riconosciuti come non dropout dal modello

Le metriche di valutazione utilizzate sono l'*accuracy, la precision, la recall, l'F1, la macro Avg e la weighted Avg.*

Accuracy

L'accuracy è il rapporto tra le predizioni corrette sul numero di predizioni totali.

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

Precisione

La precisione è il rapporto tra i valori positivi previsti correttamente e i valori positivi previsti. La precisione evidenzia le previsioni positive corrette tra tutte le previsioni positive. Una precisione alta implica un basso tasso di falsi positivi.

$$Precision = \frac{TP}{TP + FP}$$

Recall

La recall è il rapporto tra i valori positivi correttamente previsti e i valori positivi effettivi. Recall mette in evidenza la sensibilità dell'algoritmo cioè tra tutti i positivi effettivi quanti sono stati effettivamente rilevati. Un livello di recall elevato significa che un algoritmo restituisce la maggior parte dei risultati rilevanti.

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

F1-score

Il punteggio F1 é la media ponderata di Precisione e Recall. È una metrica molto più sofisticata dell'accuratezza perché tiene conto sia dei falsi positivi che dei falsi negativi. L'accuratezza è adatta solo quando sia i falsi positivi che i falsi negativi hanno un costo simile.

$$\text{F1-Score} = \frac{2 * (Recall * Precision)}{Recall + Precision}$$

Macro AVG

Per la precision, la recall e l'F1-Score abbiamo calcolato la media.

$$Score_{MacroAVG} = \frac{\text{Score}_{class_1} + \text{Score}_{class_0}}{2}$$

Dove $Support_{Class_i}$, con $i \in 0, 1$ è il numero di occorrenze effettive della classe i nel set di dati specificato.

Weighted AVG

Per la precision, la recall e l'F1-Score abbiamo calcolato la media pesata.

$$Score_{WeightedAVG} = \frac{\text{Score}_{class_1} * Support_{Class_1} + \text{Score}_{class_0} * Support_{Class_0}}{Support_{Class_1} + Support_{Class_0}}$$

Dove $Support_{Class_i}$, con $i \in 0, 1$ è il numero di occorrenze effettive della classe i nel set di dati specificato.

1.3 Studio di Ablazione

Presenteremo ora tutti i vari preprocessamenti dei dati che abbiamo eseguito e i risultati dei modelli relativi

1.4 Preprocessing Base

Alcune colonne avevano dei valori contenenti un ordine implicito che potevano influenzare negativamente il modello. Abbiamo rimpiazzato i codici relativi alle segueti features con i nomi delle features, così da poter utilizzare la funzione *get_dummies()* ed ottenere un *one-hot encoding*:

- Diploma_scuola_superiore
- area_geografica_scuolasuperiore
- area_geografica_residenza
- Ambito

```
features["Diploma_scuola_superiore"] = features["Diploma_scuola_superiore
    "].map({1: "Classico", 2: "Scientifico", 3: "Linguistico", 4: "
    Magistrale", 5: "Artistico", 6: "Tecnico", 7: "Professionale", 8: "
    Altro_italiano", 9: "Estero", 99: "Non_disponibile"})
features["area_geografica_scuolasuperiore"] = features["
    area_geografica_scuolasuperiore"].map({1: "Emilia_romagna", 2: "Nord",
    3: "Centro", 4: "Sud_isole", 5: "Estero", 99: "Non_disponibile"})
features["area_geografica_residenza"] = features["
    area_geografica_residenza"].map({1: "Emilia_romagna", 2: "Nord", 3: "
    Centro", 4: "Sud_isole", 5: "Estero", 99: "Non_disponibile"})
features["Ambito"] = features["Ambito"].map({1: "Economia", 2: "Farmacia",
     3: "Giurisprudenza", 4: "Ingegneria", 5: "Lingue", 6: "Medicina", 7: "
    Veterinaria", 8: "Psicologia", 9: "Scienze", 10: "
    Scienze_agroalimanetari", 11: "Scienze_educazione_formazione", 12: "
    Scienze_motorie", 13: "Scienze_politiche", 14: "Scienze_statistiche",
    15: "Sociologia", 16: "Studi_umanistici"})
```

Abbiamo sostituito la data di nascita creando una nuova colonna con l'età dello studente, tramite una differenza tra la data di nascita e l'anno della coorte associato a quello studente.
Abbiamo ritenuto importante l'età ai fini dell'addestramento principalmente per la natura del problema, poichè potremmo supporre che uno studente che inizia l'università ad un'età maggiore sia più propenso ad abbandonare se incontra delle difficoltà. Abbiamo sostituito i valori **NaN** della feature *voto_scuola_superiore* con la media. Inoltre sono state eliminate le tuple contente almeno un NaN e droppate le colonne non utili ai fini dell'addestramento:

- ID_Studente: poichè si tratta semplicemente di un valore crescente che indica il numero della tupla. Come vedremo in futuro utilizzarlo può creare un bias nel modello.
- Classe_ISEE: poichè c'è la colonna Merito_ISEE contentente già il valore dell'ISEE
- DataNascita: poichè verrà sostituita con l'età dello studente al primo anno di corso
- Coorte: poichè si tratta di un valore appartenente all'insieme 2016,2017,2018 inutile ai fini dell'addestramento.

1.4.1 Divisione del dataset

Abbiamo diviso il dataset in traing e test relativamente 80% e 20% poichè tramite varie ricerche in letteratura è uno dei rapporti più utilizzati per il training di un modello. La scelta di effettuare una divisione in percentuali 80-20 è stata trovata in letteratura in un articolo pubblicato da Gholamy et Al. [**?**].

1.5 Momento dell'iscrizione

Per quanto riguarda il modello predittivo degli studenti al momento dell'iscrizione abbiamo eliminato dal dataset la colonna OFA_Superati poichè al momento dell'iscrizione di uno studente non sappiamo ancora se questo supererà i propri OFA o meno.

Evaluating algorithms

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0	0.91	0.99	0.95	0.91	0.99	0.95	0.91	1.00	0.95	4426
1	0.23	0.02	0.04	0.35	0.03	0.06	0.00	0.00	0.00	431
Accuracy	0.9065266625488985			0.9112620959439983			0.91476219			4857
Macro AVG	0.57	0.51	0.50	0.63	0.51	0.51	0.46	0.50	0.48	4857
Weighted AVG	0.85	0.91	0.87	0.86	0.91	0.87	0.83	0.91	0.87	4857

Tabella 1.1: P = Precision, R = Recall e F1 = F1-score

Come si vede i risultati in accuracy sono ottimi, arrivando ad un 90% circa in tutti i modelli. Se si va ad analizzare peró i risultati in precision, recall e f1-score degli 1, quindi degli studenti che droppano, si può notare come queste siano vicine allo zero, mostrando come il risultato alto in accuracy sia legato al corretto riconoscimento degli studenti che non abbandonano gli studi. Per ovviare a questo problema, da questo momento in poi bilanceremo il dataset 50-50 tra studenti che abbandonano e non.

1.6 Momento dell'Iscrizione Bilanciato

Notando come i risultati ottenuti precedemente abbiano dei risultati molto bassi per quanto riguarda la classe dei Dropout, abbiamo cambiato strategia e deciso di andare a bilanciare il dataset con un bilanciamento 50-50. Abbiamo bilanciato il dataset con approccio under-sampling pareggiando il numero di dropout e non dropout. Abbiamo optato per l'under-sampling e non per l'over-sampling poichè crediamo che andare a generare dati sintetici non sia adatto data la natura del problema.
Abbiammo eseguito diverse prove con due tipi di bilanciamento diversi:

- il primo fatto da noi, dove andiamo a prendere i primi n dropout della classe 1 e della classe 0 (Codice 1.1)
- il secondo utilizzando la classe RandomUnderSampler di imblearn (Codice 1.2)

```
#BILANCIAMENTO 1
dropout = pd.DataFrame()
noDropout = pd.DataFrame()

for ind in features.index:
```

```
    studente = pd.DataFrame(features.loc[[ind]])
    if features["Abbandoni"][ind] == 1:
        dropout = pd.concat([studente, dropout], ignore_index=True, axis = 0)
    else:
        noDropout = pd.concat([studente, noDropout], ignore_index=True, axis = 0)

#Balancing: I'm going to get as many non-dropouts as there are dropouts
final = pd.DataFrame()
for ind in dropout.index:
    studente_drop = pd.DataFrame(dropout.loc[[ind]])
    studente_noDrop = pd.DataFrame(noDropout.loc[[ind]])
    final = pd.concat([studente_drop, final], ignore_index=True, axis = 0)
    final = pd.concat([studente_noDrop, final], ignore_index=True, axis = 0)

features = final
```

Listing 1.1: Primo bilanciamento "a mano"

```
#BILANCIAMENTO 2
from collections import Counter
from sklearn.datasets import make_classification
from imblearn.under_sampling import RandomUnderSampler

#Resampling data
rus = RandomUnderSampler(sampling_strategy = "majority", random_state=42)
X_res, y_res = rus.fit_resample(X, y)
```

Listing 1.2: Secondo bilanciamento tramite RandomUnderSampler

Quindi d'ora in poi mostreremo due tabelle dei risultati:

1. Con il primo bilanciamento
2. Con il secondo bilanciamento

1.6.1 Risultati

Evaluating algorithms (Primo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0	0.62	0.73	0.67	0.68	0.63	0.66	0.59	0.60	0.60	437
1	0.64	0.51	0.57	0.63	0.68	0.65	0.56	0.55	0.56	401
Accuracy	0.6252983293556086			0.6539379474940334			0.5763723150357996			838
Macro AVG	0.63	0.62	0.62	0.65	0.65	0.65	0.58	0.58	0.58	838
Weighted AVG	0.63	0.63	0.62	0.66	0.65	0.65	0.58	0.58	0.58	838

Tabella 1.2: P = Precision, R = Recall e F1 = F1-score

Evaluating algorithms (Secondo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0	0.64	0.71	0.68	0.69	0.63	0.66	0.68	0.60	0.64	455
1	0.61	0.52	0.56	0.60	0.67	0.64	0.58	0.66	0.62	383
Accuracy	0.6276849642004774			0.6479713603818615			0.6276849642004774			838
Macro AVG	0.62	0.62	0.62	0.65	0.65	0.65	0.63	0.63	0.63	838
Weighted AVG	0.63	0.63	0.62	0.65	0.65	0.65	0.63	0.63	0.63	838

Tabella 1.3: P = Precision, R = Recall e F1 = F1-score

1.7 Momento dell'iscrizione bilanciato con ISEE

Ci siamo accorti, che la colonna che presentava più valori NaN e che di conseguenza ci faceva perdere più tuple era la colonna dell'ISEE. La colonna Classe_ISEE presentava tra i vari valori possibili, i valori 0 e 1, dove lo 0 rappresenta l'ISEE non disponibile (coorte del 2016 per il quale questo valore non è stato raccolto) e l'1 rappresentava *ISEE non presentato.* Ci siamo interrogati quindi sulla probabile motivazione legata alla non presentazione dell'ISEE e una delle possibili spiegazioni, oltre ad una semplice dimenticanza, è legata al fatto che l'ISEE sia alto a tal punto da non garantire nessun vantaggio economico in termini di tasse universitarie da pagare. Tramite delle ricerche sui portali

dell'UniBo abbiamo scoperto come oltre i 70.000€ non ci sia più nessuna agevolazione fiscale ma bisogna pagare l'importo massimo. Quindi, oltre ai procedimenti precedenti sono stati sostituiti i valori **NaN** della colonna *Merito_ISEE* in corrispondenza di una colonna *Classe_ISEE* con valore 1, con la media degli ISEE oltre i 70000 euro, in modo tale da cercare di perdere meno tuple possibili dopo aver droppato quelle con almeno un NaN.

```
rich =  features[features['Merito_ISEE'] > 70000]
rich = rich['Merito_ISEE']
mean_rich = rich.mean()
features = features[features['Classe_ISEE'] != 0]
features['Merito_ISEE'] = features['Merito_ISEE'].fillna(mean_rich)
```

1.7.1 Risultati

Evaluating algorithms (Primo Bilanciamento)

	SVM			**RF**			**LR**			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0				0.68	0.71	0.70	0.66	0.68	0.67	867
1				0.71	0.67	0.69	0.68	0.65	0.66	883
Accuracy			...	0.6931428571428572			0.666285714285			1750
Macro AVG			...	0.69	0.69	0.69	0.67	0.67	0.67	1750
Weighted AVG			...	0.69	0.69	0.69	0.67	0.67	0.67	1750

Tabella 1.4: P = Precision, R = Recall e F1 = F1-score

In questo e tutti i modelli in cui recuperiamo una maggior quantità di dati andando a riempire i valori NaN con la media degli ISEE oltre i 70.000, non siamo stati in grado di utilizzare il modello SVM su questo dataset per mancanza di potenza di calcolo.

Evaluating algorithms (Secondo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0				0.67	0.67	0.67	0.57	0.61	0.59	883
1				0.67	0.66	0.66	0.57	0.52	0.54	867
Accuracy			...	0.664			0.5674285714			1750
Macro AVG			...	0.66	0.66	0.66	0.57	0.57	0.57	1750
Weighted AVG			...	0.66	0.66	0.66	0.57	0.57	0.57	1750

Tabella 1.5: P = Precision, R = Recall e F1 = F1-score

1.8 Fine del primo anno

Siamo andati poi a valutare l'impatto che i cfu ottenuti durante tutto l'anno avessero. Abbiamo quindi per ogni studente aggiunto ai dati precedenti due colonne contenenti una la somma totale dei crediti ottenuti alla fine del primo anno di corso e l'altra la media dei voti ottenuti a fine anno di corso.

Evaluating algorithms (Primo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0	0.90	0.90	0.90	0.91	0.90	0.90	0.93	0.88	0.90	437
1	0.89	0.90	0.89	0.89	0.90	0.90	0.87	0.92	0.90	401
Accuracy	0.8973747016706444			0.9009546539379475			0.8949880668257757			838
Macro AVG	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.90	838
Weighted AVG	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.90	838

Tabella 1.6: P = Precision, R = Recall e F1 = F1-score

Evaluating algorithms (Secondo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0	0.90	0.91	0.90	0.91	0.90	0.91	0.90	0.90	0.90	455
1	0.89	0.88	0.88	0.88	0.89	0.89	0.88	0.88	0.88	383
Accuracy	0.8949880668257757			0.8973747016706444			0.8914081145584726			838
Macro AVG	0.89	0.89	0.89	0.90	0.90	0.90	0.89	0.89	0.89	838
Weighted AVG	0.89	0.89	0.89	0.90	0.90	0.90	0.89	0.89	0.89	838

Tabella 1.7: P = Precision, R = Recall e F1 = F1-score

1.9 Fine del primo anno con ISEE

Abbiamo poi anche in questo caso, recuperato le tuple con ISEE al preprocessing precedente.

Evaluating algorithms (Primo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0				0.88	0.92	0.90	0.86	0.89	0.88	867
1				0.92	0.88	0.90	0.89	0.86	0.87	883
Accuracy			...	0.9022857142857142			0.8754285714285714			1750
Macro AVG			...	0.90	0.90	0.90	0.88	0.88	0.88	1750
Weighted AVG			...	0.90	0.90	0.90	0.88	0.88	0.88	1750

Tabella 1.8: P = Precision, R = Recall e F1 = F1-score

Evaluating algorithms (Secondo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0				0.89	0.88	0.89	0.88	0.86	0.87	883
1				0.88	0.89	0.89	0.86	0.88	0.87	867
Accuracy			...	0.8868571428571429			0.8725714285714286			1750
Macro AVG			...	0.89	0.89	0.89	0.87	0.87	0.87	1750
Weighted AVG			...	0.89	0.89	0.89	0.87	0.87	0.87	1750

Tabella 1.9: P = Precision, R = Recall e F1 = F1-score

1.10 Fine del primo semestre

Successivamente abbiamo voluto valutare l'impatto degli esami ottenuti al primo semestre. Siamo andati quindi ad aggiungere le stesse due colonne aggiunte nel preprocessamento precedente, considerando però solo CFU e voti ottenuti fino al 1/03.

Evaluating algorithms (Primo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0	0.77	0.84	0.80	0.79	0.80	0.79	0.79	0.83	0.81	437
1	0.81	0.72	0.76	0.78	0.77	0.77	0.80	0.79	0.79	401
Accuracy	0.7863961813842482			0.7852028639618138			0.7923627684964201			838
Macro AVG	0.79	0.78	0.78	0.78	0.78	0.78	0.79	0.79	0.79	838
Weighted AVG	0.79	0.79	0.79	0.79	0.79	0.79	0.79	0.79	0.79	838

Tabella 1.10: P = Precision, R = Recall e F1 = F1-score

Evaluating algorithms (Secondo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0	0.79	0.83	0.81	0.80	0.79	0.80	0.81	0.81	0.81	455
1	0.78	0.73	0.76	0.76	0.77	0.76	0.77	0.77	0.77	383
Accuracy	0.7852028639618138			0.7828162291169452			0.7911694510739857			838
Macro AVG	0.79	0.78	0.78	0.78	0.78	0.78	0.79	0.79	0.79	838
Weighted AVG	0.79	0.79	0.78	0.78	0.78	0.78	0.79	0.79	0.79	838

Tabella 1.11: P = Precision, R = Recall e F1 = F1-score

1.11 Fine del primo semestre con ISEE

Vediamo a questo punto l'impatto che avremmo aggiungendo anche in questo caso le tuple con l'isee.

Evaluating algorithms (Primo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0				0.79	0.83	0.81	0.75	0.85	0.79	867
1				0.82	0.78	0.80	0.83	0.72	0.77	883
Accuracy			...	0.8062857142857143			0.7828571428571428			1750
Macro AVG			...	0.81	0.81	0.81	0.78	0.78	0.78	1750
Weighted AVG			...	0.81	0.81	0.81	0.78	0.78	0.78	1750

Tabella 1.12: P = Precision, R = Recall e F1 = F1-score

Evaluating algorithms (Secondo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0				0.79	0.80	0.79	0.75	0.82	0.78	883
1				0.79	0.78	0.78	0.80	0.72	0.76	867
Accuracy			...	0.788			0.7725714285714286			1750
Macro AVG			...	0.79	0.79	0.79	0.78	0.77	0.77	1750
Weighted AVG			...	0.79	0.79	0.79	0.77	0.77	0.77	1750

Tabella 1.13: P = Precision, R = Recall e F1 = F1-score

1.12 Extra

Come si può notare la differenza tra i risultati delle due versioni dei bilanciamenti è molto lieve. In alcuni dei nostre primi risultati, avevamo notato come ci fosse una sostanziale differenza tra queste due diverse tecniche di bilanciamento, con il nostro bilanciamento che, nel caso in cui si consideravano tutti gli esami arrivava ad accuratezze del 95% circa. Questo era dettato dal fatto che utilizzavamo nel nostro modello la colonna *IDStudente* che, nel nostro bilanciamento, venendo selezionata in tuple contigue influenzava molto di più il modello rispetto al bilanciamento tramite *RandomUnderSampler*. Questo spiega il perchè droppiamo la colonna IDStudente ed il perchè abbiamo deciso di mostrare comunque i risultati di entrambi i bilanciamenti per quanto simili.

Evaluating algorithms (Primo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0	0.91	0.99	0.95	0.92	0.99	0.96	0.85	0.95	0.90	437
1	0.99	0.90	0.94	0.99	0.91	0.95	0.94	0.81	0.87	401
Accuracy	0.9463007159904535			0.9522673031026253			0.8866348448687351			838
Macro AVG	0.95	0.94	0.94	0.96	0.95	0.95	0.89	0.88	0.89	838
Weighted AVG	0.95	0.95	0.95	0.95	0.95	0.95	0.89	0.89	0.89	838

Tabella 1.14: P = Precision, R = Recall e F1 = F1-score

Evaluating algorithms (Secondo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0	0.65	0.60	0.63	0.71	0.63	0.66	0.62	0.56	0.59	455
1	0.57	0.62	0.59	0.61	0.69	0.65	0.53	0.58	0.55	383
Accuracy	0.609785202863918			0.6551312649164678			0.5715990453460621			838
Macro AVG	0.61	0.61	0.61	0.66	0.66	0.65	0.57	0.57	0.57	838
Weighted AVG	0.61	0.61	0.61	0.66	0.66	0.66	0.58	0.57	0.57	838

Tabella 1.15: P = Precision, R = Recall e F1 = F1-score

1.12.1 Stacking Algorithm

Abbiamo applicato lo Stacking Algorithm ai tre momenti con il secondo bilanciamento e senza la sostituzione dei NaN degli ISEE (altrimenti non avremmo potuto inserire SVM), definendolo in questo modo:

```
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, StackingClassifier

def get_stacking():
  # define the base models
  level0 = list()
  level0.append(('rf', RandomForestClassifier(n_estimators=207,
    random_state=42)))
  level0.append(('svm', SVC(C=0.01, kernel='linear')))
  level0.append(('lr', LogisticRegression()))
  # define meta learner model
  level1 = LogisticRegression()
  # define the stacking ensemble
  model = StackingClassifier(estimators=level0, final_estimator=level1, cv
    =5)
  return model
```

Listing 1.3: Stacking definition

Come si può vedere dal codice 1.3 abbiamo applicato al:

- **Livello 0:** Random Forest, SVM e Logistic Regression (i nostri tre modelli)
- **Livello 1:** Logistic Regression (di default)

Momento dell'Iscrizione Bilanciato

Evaluating algorithm (Secondo Bilanciamento)

	Precision	Recall	F1-score	Support
0	0.69	0.66	0.67	455
1	0.62	0.65	0.63	383
Accuracy	0.6551312649164678			838
Macro AVG	0.65	0.65	0.65	838
Weighted AVG	0.66	0.66	0.66	838

Fine del primo anno

Evaluating algorithm (Secondo Bilanciamento)

	Precision	Recall	F1-score	Support
0	0.91	0.89	0.90	455
1	0.88	0.89	0.88	383
Accuracy	0.8926014319809069			838
Macro AVG	0.89	0.89	0.89	838
Weighted AVG	0.89	0.89	0.89	838

Fine del primo semestre

Evaluating algorithm (Secondo Bilanciamento)

	Precision	Recall	F1-score	Support
0	0.80	0.80	0.80	455
1	0.76	0.77	0.76	383
Accuracy	0.7828162291169452			838
Macro AVG	0.78	0.78	0.78	838
Weighted AVG	0.78	0.78	0.78	838

1.13 Studio di comparazione

Il nostro riferimento principale per comparare i risultati è il lavoro di Del Bonifro et Al. [**?**]. Confrontando il modello che ha i risultati migliori, quindi il nostro modello alla fine del primo anno con il modello *Basic + ALR + CC* di [**?**] possiamo notare come otteniamo un'accuracy migliore di circa 2 punti percentuali. Anche per quanto riguarda la Recall otteniamo un miglioramento rispettamente di 0.01 per il modello SVM e di 0.04 per il modello RF.