



UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

Sistemi intelligenti per la predizione dell'approvazione di un prestito bancario

Gruppo di lavoro:

- Gabriele Marrano, mat: 735972, g.marrano2@studenti.uniba.it

Repository Github:

- <https://github.com/gas300/Progetto-ICON-Gabriele-Marrano>

AA 2022-2023

Introduzione

Questo progetto è volto a testimoniare le competenze acquisite nel corso di Ingegneria della conoscenza, in particolare in merito a ricerca/rappresentazione, ragionamento, apprendimento.

Il progetto verte sull'approvazione di richieste di prestito monetario ad un istituto bancario, ogni richiesta è rappresentata attraverso l'uso di varie features, nonché una variabile target che indicherà se il prestito risulta approvato o meno.

Quindi si procederà con una prima preelaborazione del dataset con conseguente analisi e commento di osservazioni per poi utilizzare questo come input ad alcuni modelli di apprendimento classici e probabilistici. Per ogni modello verranno eseguiti 3 casi:

1. Con l'utilizzo del dataset originale
2. Con l'utilizzo del dataset undersampled
3. Con l'utilizzo del dataset oversampled

Per ogni caso verrà valutate performance ed overfitting e dove necessario si cercherà di migliorare le performance o ridurre l'overfitting.

In conclusione dei capitoli basati sull'apprendimento vengono fatte delle osservazioni ed una discussione dei risultati che porterà, nel capitolo successivo, all'utilizzo di regole alternative nella Knowledge Base e del confronto tra queste e quelle stabilite a priori; quindi una valutazione delle performance della KB per la predizioni dei prestiti approvati e dei test per verificare la correttezza logica di questa.

Si fa notare che i valori monetari presenti nel dataset potrebbero sembrare eccessivi in quanto fanno riferimento alla rupia indiana il quale cambio attuale è 1 EUR = 89 INR.

Sommario

Creazione e preelaborazione del dataset	4
Creazione del dataset	4
Preelaborazione del dataset	4
Analisi del dataset	5
Valutazione correlazioni tra le features	6
Apprendimento supervisionato	7
Decisione progettuale	8
KNN con dataset originale	8
Valutazione performance e presenza overfitting	8
KNN con dataset undersampled	10
Valutazione performance e presenza overfitting	10
KNN con dataset oversampled	12
Valutazione performance e presenza overfitting	12
Random forest con dataset originale	14
Valutazione performance e presenza overfitting	14
Random forest con dataset undersampled	17
Valutazione performance e presenza overfitting	17
Random forest con dataset oversampled	19

Valutazione performance e presenza overfitting	19
Multilayer Perceptron con dataset originale	21
Valutazione performance e presenza overfitting	21
Multilayer Perceptron con dataset undersampled	23
Valutazione performance e presenza overfitting	23
Multilayer Perceptron con dataset oversampled	25
Valutazione performance e presenza overfitting	25
Apprendimento probabilistico	27
Decisione progettuale	27
Naive Bayes	27
Valutazione performance e presenza overfitting	27
Osservazioni	29
Discussione risultati	32
Rappresentazione della conoscenza tramite Knowledge Base	33
Decisione progettuale	33
Creazione della knowledge base	33
Creazione ed uso delle query per l'applicazione delle regole	34
Calcolo delle predizioni dei prestiti approvati e delle relative performance	35
Discussione risultati e conclusione	35
Possibili sviluppi futuri	35

Creazione e preelaborazione del dataset

Creazione del dataset

Si è iniziato con la creazione del dataset mediante caricamento di un file con estensione .csv trovato su kaggle al seguente [link](#). Il dataset inizialmente si presenta così:

no_of_dependents	education	self_employed	income_annum	loan_amount	loan_term	cibil_score	residential_assets_value	commercial_assets_value	luxury_assets_value	bank_asset_value	loan_status	
0	2	Graduate	No	₹9,600,000.00	₹29,900,000.00	12	778	₹2,400,000.00	₹17,600,000.00	₹22,700,000.00	₹8,000,000.00	Approved
1	0	Not Graduate	Yes	₹4,100,000.00	₹12,200,000.00	8	417	₹2,700,000.00	₹2,200,000.00	₹8,800,000.00	₹3,300,000.00	Rejected
2	3	Graduate	No	₹9,100,000.00	₹29,700,000.00	20	506	₹7,100,000.00	₹4,500,000.00	₹33,300,000.00	₹12,800,000.00	Rejected
3	3	Graduate	No	₹8,200,000.00	₹30,700,000.00	8	467	₹18,200,000.00	₹3,300,000.00	₹23,300,000.00	₹7,900,000.00	Rejected
4	5	Not Graduate	Yes	₹9,800,000.00	₹24,200,000.00	20	382	₹12,400,000.00	₹8,200,000.00	₹29,400,000.00	₹5,000,000.00	Rejected

Spiegazione delle features:

- *no_of_dependents*: Numero di persone a carico del richiedente
- *education*: Livello di istruzione del richiedente
- *self_employed*: Condizione occupazionale del richiedente
- *income_annum*: Reddito annuale del richiedente
- *loan_amount*: Ammontare del prestito
- *loan_term*: Durata del prestito in anni
- *cibil_score*: Punteggio rappresentante la solvibilità creditizia dei richiedenti di prestiti o mutui. E' calcolato dall'agenzia di informazione di credito Credit Information Bureau India Limited (CIBIL)
- *residential_assets_value*: Valore dei beni residenziali
- *commercial_assets_value*: Valore dei beni commerciali
- *luxury_assets_value*: Valore dei beni di lusso
- *bank_asset_value*: Valore dei beni depositati in banca
- *loan_status*: Stato di approvazione del prestito

Preelaborazione del dataset

Seppur decisamente facilmente leggibile e perfettamente formattata, la tabella posta qui sopra rende difficoltose tutte le operazioni matematiche che seguiranno, perciò si è deciso di mostrare la tabella per dare la possibilità, al lettore/utilizzatore di questo applicativo, di una più semplice comprensione dei valori tabellari e della struttura del dataset.

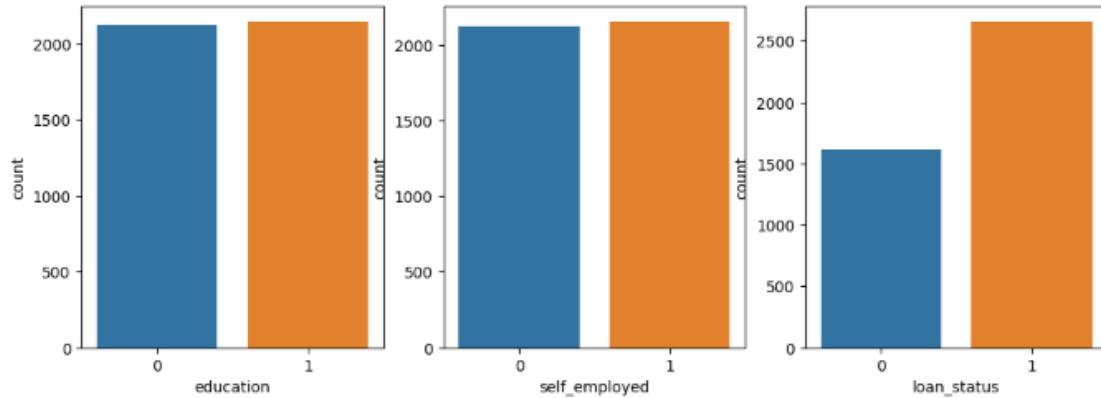
Per il proseguo del caso di studio invece si utilizzerà la tabella ricavata nel seguente passaggio, che mappa i valori stringa rappresentati situazione binarie in appunto valori numeri di 0 e 1 per rispettivamente l'assenza e la presenza del valore della feature in questione.

no_of_dependents	education	self_employed	income_annum	loan_amount	loan_term	cibil_score	residential_assets_value	commercial_assets_value	luxury_assets_value	bank_asset_value	loan_status	
0	2	1	0	9600000.0	29900000.0	12	778	2400000.0	17600000.0	22700000.0	8000000.0	1
1	0	0	1	4100000.0	12200000.0	8	417	2700000.0	2200000.0	8800000.0	3300000.0	0
2	3	1	0	9100000.0	29700000.0	20	506	7100000.0	4500000.0	33300000.0	12800000.0	0
3	3	1	0	8200000.0	30700000.0	8	467	18200000.0	3300000.0	23300000.0	7900000.0	0
4	5	0	1	9800000.0	24200000.0	20	382	12400000.0	8200000.0	29400000.0	5000000.0	0

Effettuato anche un breve controllo sulla mancanza di valori si passa alla fase di analisi e valutazione delle correlazioni tra le features.

Analisi del dataset

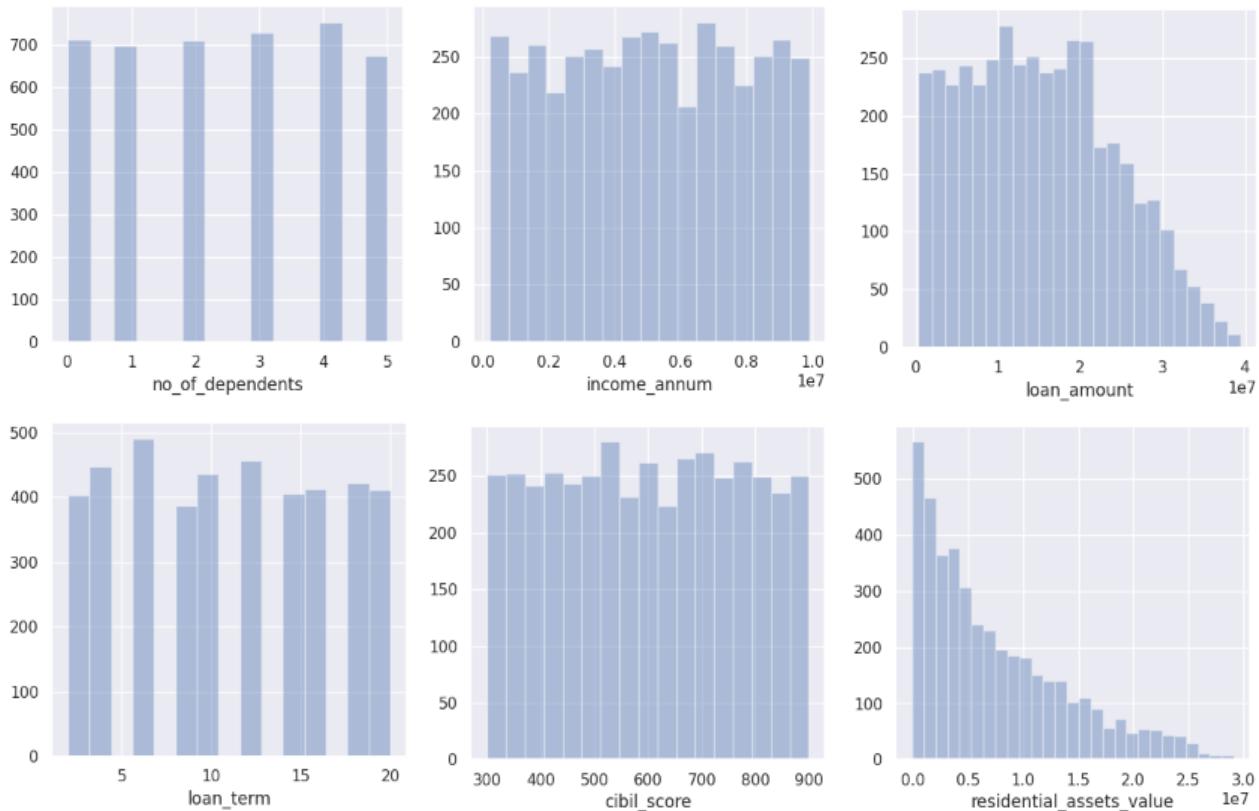
In questo paragrafo si calcolano la frequenza dei valori di tutte le features del dataset partendo da quelle binarie

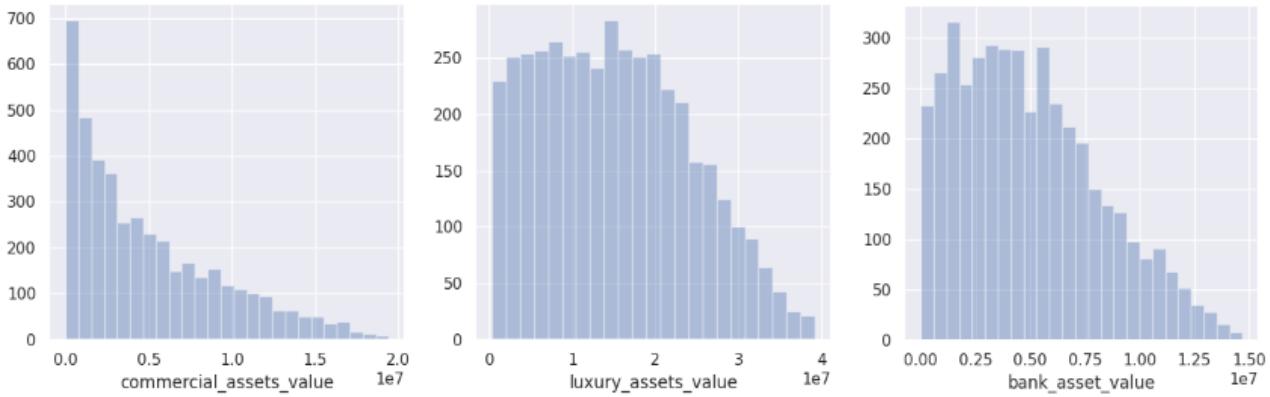


E' possibile notare come il dataframe in questione, seppur non perfettamente bilanciato sulla variabile target, non presenti gravi segni di sbilanciamento.

In seguito verrà valutato l'utilizzo dell'undersampling e dell'oversampling per migliorare le performance dei modelli di ML.

Si prosegue con le features continue





Altre osservazioni che possono essere fatte sono le seguenti:

- Il dataframe presenta una varietà di soggetti ottimamente bilanciata per quanto riguarda il reddito annuale del richiedente.
- Nonostante il punto precedente è possibile notare una decisa differenza nella gestione del denaro, in particolare:
 - La richiesta di prestito presenta valori molto simili a quella del valore dei beni di lusso, questo potrebbe significare che la tendenza dei soggetti nel dataframe sia quella di comprare beni di lusso attraverso la richiesta di un mutuo
 - E' presente un numero di soggetti decisamente elevato che non possiede o possiede in scarse quantità beni commerciali o residenziali che appunto sono una forma importante di reddito generato da investimenti
- Una fetta molto consistente dei richiedenti di un prestito detiene un deposito di beni bancari (denaro liquido, partecipazioni in fondi comuni di risparmio/investimento, azioni, titoli di stato ecc..) piuttosto ridotto. Ciò può significare quanto segue:
 - Un'ampia circolazione di contante
 - Gran parte del reddito viene utilizzato per spese ordinarie

Valutazione correlazioni tra le features

Il seguente schema mostra la correlazione media presente tra i valori delle features



Osservazioni:

- Il reddito annuale presenta una forte correlazione diretta con l'ammontare del prestito richiesto e i valori del bene di lusso. Inoltre non trascurabile la buona correlazione diretta anche con il valore di asset commerciali e residenziali, nonché il valore dei beni bancari
- La richiesta della quantità di debito rispecchia molto le correlazioni presenti nel punto precedente
- Il valore dei beni residenziali, di lusso e commerciali presentano una non leggera correlazione tra essi
- L' **approvazione del prestito** presenta un'importante correlazione con il punteggio calcolato dal **CIBIL**

Apprendimento supervisionato

L'apprendimento supervisionato è uno dei principali approcci nell'ambito del machine learning, che consiste nell'addestrare un modello utilizzando un insieme di dati di addestramento etichettato.

Alcune considerazioni :

1. Il dataframe risulta essere **sbilanciato**. In particolare lo stato dei prestiti approvati è di circa il 62% maggiore rispetto a quelli non approvati. **Seppure non è uno sbilanciamento estremo** si è deciso di verificare se l'applicazione di tecniche di **oversampling** e **undersampling** potesse portare migliori performance.
2. Per ottimizzare le performance si farà utilizzo di una **Grid Search**. La Grid Search è una tecnica utilizzata per ottimizzare i parametri di un modello di machine learning. L'idea principale è di esplorare tutte le combinazioni possibili di valori dei parametri specificati in una griglia predefinita e selezionare la combinazione che offre le migliori prestazioni del modello.
3. Al fine di ottenere una stima più affidabile delle prestazioni del modello rispetto a una singola divisione in set di addestramento e test si è deciso di usare la tecnica di **k-fold cross-validation**. L'obiettivo della k-fold cross-validation è quello di suddividere il dataset in k subset (folds) di dimensioni approssimativamente uguali. Il modello viene addestrato e valutato k volte, utilizzando k-1 folds come set di addestramento e l'ultimo fold come set di test.
4. Si è deciso di **ottimizzare la precision** in quanto predire un errore di debito approvato quando in realtà non è così, non è più o meno aggravante del caso opposto.

Questo capitolo quindi si evolverà come segue:

1. Per ogni modello si eseguiranno i 3 seguenti casi:
 - 1.1. Creazione modello --> applicazione della grid search per ottimizzazione parametri --> K-fold cross validation per addestramento e test --> Calcolo performance
 - 1.2. Creazione modello --> applicazione undersampling al df --> applicazione della grid search per ottimizzazione parametri --> K-fold cross validation per addestramento e test --> Calcolo performance
 - 1.3. Creazione modello --> applicazione oversampling al df --> applicazione della grid search per ottimizzazione parametri --> K-fold cross validation per addestramento e test --> Calcolo performance

2. Discussione delle performance e conclusioni

I modelli utilizzati sono: Knn, Random Forest e MLP.

Nei seguenti paragrafi si ometteranno le spiegazioni del funzionamento teorico dei modelli in quanto esula dall'obiettivo di questa documentazione.

Decisione progettuale

Nella fase di valutazione delle performance e presenza overfitting si sceglie di variare il parametro per modello nel modo seguente:

- **KNN** → “**n_neighbors**”, in quanto l’unico parametro del modello.
- **RF** → “**max_depth**”, in quanto si è ritenuto di maggior peso per la presenza di overfitting rispetto al “**n_estimators**”.
- **MLP** → “**hidden_layer_sizes**”, in quanto definiva la struttura della rete neurale.

KNN con dataset originale

Valutazione performance e presenza overfitting

A seguito dell’addestramento del KNN attraverso l’utilizzo della grid search e della cross validation i risultati sono stati i seguenti:

```
Parametri ottimali: {'n_neighbors': 5, 'weights': 'uniform'}

% ----- KNN test ----- %

Average Recall: 0.7421010072350688
Average Precision: 0.6332411317252247
Average Accuracy: 0.5720316434123869
Average F1-score: 0.6832201504511843

% ----- KNN train ----- %

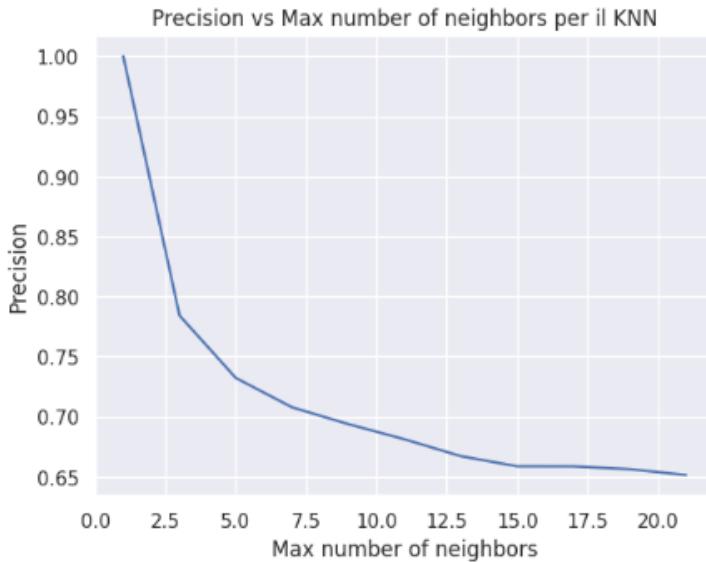
Average Recall: 0.8502760526311185
Average Precision: 0.7348502728913402
Average Accuracy: 0.7159625057044434
Average F1-score: 0.788352084681089

% ----- %

+---+---+---+---+---+
| | Modello | Recall | Precision | Accuracy | F1-score |
+---+---+---+---+---+
| 0 | KNN df originale | 0.742101 | 0.633241 | 0.572032 | 0.68322 |
+---+---+---+---+---+
```

E’ possibile notare una **grande differenza tra gli score** delle performance generati dalla valutazione del modello **sui dati di train** e quelli generati **sui dati di test**. Questo è un segno di **overfitting** perciò nei passi seguenti si approfondirà e cercherà di risolvere questa questione.

Si procede quindi con il calcolo e la conseguente stampa della curva **precision vs n_neighbors**, per capire come influisce la variazione del numero dei vicini sullo score della precision.



Dal grafico è possibile osservare che al diminuire del numero di vicini aumenta lo score della precision in modo parabolico. Questo grafico mostra come un **KNN con K=1 andrebbe facilmente in overfitting** in quanto l'adattamento ai dati di train sarebbe estremo. Nel nostro caso invece la grid search ha scelto 5 come valore dell'iperparametro "n_neighbors", portando sicuramente a diminuire il grado di overfitting ma non abbastanza. Si procede perciò con un **aumento del numero di vicini** che dovrebbe portare ad una **riduzione dell'overfitting**, quindi **performance probabilmente minori** ma sicuramente **più veritieri**. Si sceglie quindi come numero di vicini 15 e si riaddestra il modello.

```
Parametri attuali: n_neighbors = 15, weights = uniform
% ----- KNN test -----
Average Recall: 0.8497715988083417
Average Precision: 0.6259247591519297
Average Accuracy: 0.5905438093039107
Average F1-score: 0.7208112334083522

% ----- KNN train -----
Average Recall: 0.887591928588553
Average Precision: 0.6621228762361053
Average Accuracy: 0.6482653751088907
Average F1-score: 0.7584530055658758

% ----- Grid Search -----
Performance del modello con parametri della Grid Search
+---+-----+-----+-----+-----+
| | Modello | Recall | Precision | Accuracy | F1-score |
+---+-----+-----+-----+-----+
| 0 | KNN df originale | 0.742101 | 0.633241 | 0.572032 | 0.68322 |
+---+-----+-----+-----+-----+-----+
```



```
Performance del modello con parametri aggiornati
+---+-----+-----+-----+-----+
| | Modello | Recall | Precision | Accuracy | F1-score |
+---+-----+-----+-----+-----+
| 0 | KNN df originale | 0.849772 | 0.625925 | 0.590544 | 0.720811 |
+---+-----+-----+-----+-----+-----+
```

Come ipotizzato, è possibile notare una **minor differenza tra gli score** delle performance generati dalla valutazione del modello **sui dati di train** e quelli generati **sui dati di test**, rispetto al caso precedente.

Questo indica una **netta riduzione di adattamento del modello ai dati**. Da notare che **non può esserci un'assenza di differenza tra questi valori** in quanto è logico che testare un modello sui dati su cui esso stesso è stato addestrato porterà sempre a performance migliori, l'obiettivo è appunto cercare di minimizzare questo gap.

Va inoltre notato che in linea generale **le performance calcolate sui dati di test sono addirittura migliorate rispetto al caso precedente**, questo ad ulteriore testimonianza di un minore adattamento ai dati di addestramento.

Possiamo quindi concludere che **il modello attualmente non presenta overfitting o questo è in una forma leggera**.

KNN con dataset undersampled

Valutazione performance e presenza overfitting

A seguito dell'addestramento del KNN attraverso l'utilizzo della grid search, del dataset undersampled, e della cross validation i risultati sono stati i seguenti:

```
Parametri ottimali: {'n_neighbors': 3, 'weights': 'uniform'}

% ----- KNN test ----- %

Average Recall: 0.5387393604784909
Average Precision: 0.5248673536154032
Average Accuracy: 0.5250889371767012
Average F1-score: 0.5312598767565275

% ----- KNN train ----- %

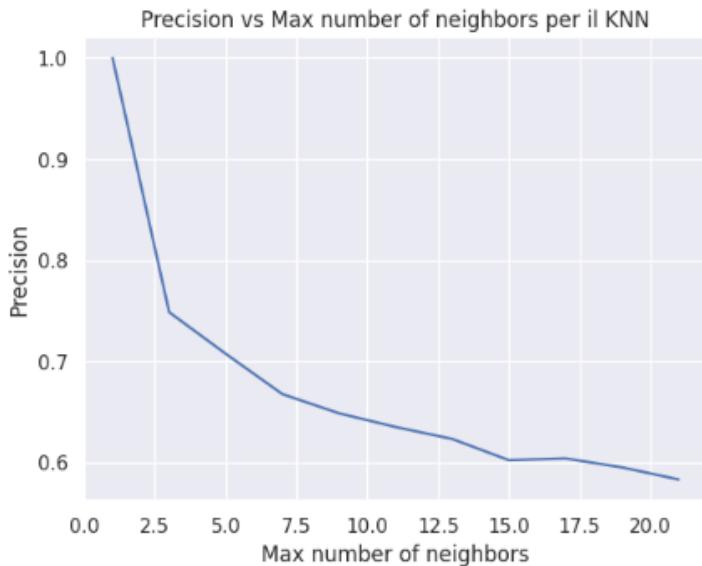
Average Recall: 0.7752984547561955
Average Precision: 0.7515843351998946
Average Accuracy: 0.7594542052536134
Average F1-score: 0.7632025359715132

% ----- %

+-----+-----+-----+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+-----+-----+-----+
| KNN df undersampled | 0.538739 | 0.524867 | 0.525089 | 0.53126 |
+-----+-----+-----+-----+
```

E' possibile notare un **grande differenza tra gli score** delle performance generati dalla valutazione del modello **sui dati di train** e quelli generati **sui dati di test**. Questo è un segno di **overfitting** perciò nei passi seguenti si approfondirà e cercherà di risolvere questa questione.

Si procede quindi con il calcolo e la conseguente stampa della curva **precision vs n_neighbors**, per capire come influisce la variazione del numero dei vicini sullo score della precision



Dal grafico è possibile osservare che al diminuire del numero di vicini aumenta lo score della precision in modo parabolico. Questo grafico mostra come un **KNN con K=1 andrebbe facilmente in overfitting** in quanto l'adattamento ai dati di train sarebbe estremo. Nel nostro caso invece la Grid Search ha scelto 3 come valore dell'iperparametro "n_neighbors", portando sicuramente a diminuire il grado di overfitting ma non abbastanza. Si procede perciò con un **aumento del numero di vicini** che dovrebbe portare ad una **riduzione dell'overfitting**, quindi **performance probabilmente minori** ma sicuramente **più veritiere**. Si sceglie quindi come numero di vicini 15 e si riaddestra il modello.

```
Parametri attuali: n_neighbors = 15, weights = uniform

% ----- KNN test -----
Average Recall: 0.5387853692201519
Average Precision: 0.5110695040126156
Average Accuracy: 0.5120762263715555
Average F1-score: 0.5242378904210971

% ----- KNN train -----
Average Recall: 0.640213171119756
Average Precision: 0.6080992943287485
Average Accuracy: 0.613762776514084
Average F1-score: 0.6237084750758872

% ----- %

Performance del modello con parametri della Grid Search
+-----+-----+-----+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+-----+-----+-----+
| KNN df undersampled | 0.538739 | 0.524867 | 0.525089 | 0.53126 |
+-----+-----+-----+-----+

Performance del modello con parametri aggiornati
+-----+-----+-----+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+-----+-----+-----+
| KNN df undersampled | 0.538785 | 0.51107 | 0.512076 | 0.524238 |
+-----+-----+-----+-----+
```

Come ipotizzato, è possibile notare una **minor differenza tra gli score** delle performance generati dalla valutazione del modello **sui dati di train** e quelli generati **sui dati di test**, rispetto al caso precedente.

Questo indica una **netta riduzione di adattamento del modello ai dati**. Da notare che **non può esserci un'assenza di differenza tra questi valori** in quanto è logico che testare un modello sui dati su cui esso stesso è stato addestrato porterà sempre a performance migliori, l'obiettivo è appunto cercare di minimizzare questo gap.

Va inoltre notato che in linea generale **le performance calcolate sui dati di test sono rimaste pressochè invariate rispetto al caso precedente**, questo può significare che **una riduzione del dataset al fine del ribilanciamento della classe minoritaria non ha portato all'ottenimento dei risultati sperati**.

Possiamo quindi concludere che **il modello attualmente presenta una forma leggera di overfitting e restituisce performance peggiori rispetto al modello precedente**.

KNN con dataset oversampled

Valutazione performance e presenza overfitting

A seguito dell'addestramento del KNN attraverso l'utilizzo della grid search, del dataset oversampled, e della cross validation i risultati sono stati i seguenti:

```
Parametri ottimali: {'n_neighbors': 1, 'weights': 'uniform'}

% ----- KNN test ----- %

Average Recall: 0.610693715420627
Average Precision: 0.731559508942694
Average Accuracy: 0.6927817424918228
Average F1-score: 0.6654070400089864

% ----- KNN train ----- %

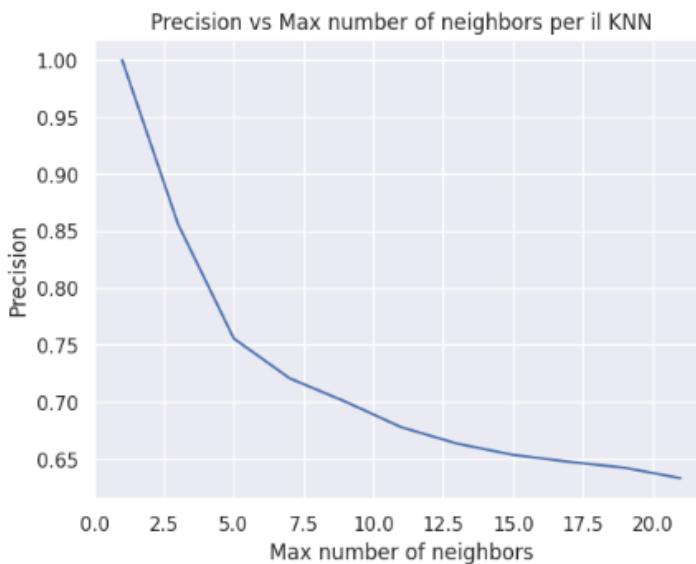
Average Recall: 1.0
Average Precision: 1.0
Average Accuracy: 1.0
Average F1-score: 1.0

% ----- %

+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+
| KNN df oversampled | 0.610694 | 0.73156 | 0.692782 | 0.665407 |
+-----+
```

E' possibile notare un **grande differenza tra gli score** delle performance generati dalla valutazione del modello **sui dati di train** e quelli generati **sui dati di test**. Inoltre **l'ottenimento di performance perfette è un campanello d'allarme**. Questo è un segno di **overfitting** perciò nei passi seguenti si approfondirà e cercherà di risolvere questa questione.

Si procede quindi con il calcolo e la conseguente stampa della curva **precision vs n_neighbors**, per capire come influisce la variazione del numero dei vicini sullo score della precision



Dal grafico è possibile osservare che al diminuire del numero di vicini aumenta lo score della precision in modo parabolico. Questo grafico mostra come un **KNN con K=1 andrebbe facilmente in overfitting** in quanto l'adattamento ai dati di train sarebbe estremo. Nel nostro caso invece la Grid Search ha scelto 1 come valore dell'iperparametro "n_neighbors", rendendo massimo il grado di overfitting. Si procede perciò con un **aumento del numero di vicini** che dovrebbe portare ad una **riduzione dell'overfitting**, quindi **performance probabilmente minori** ma sicuramente **più veritieri**. Si sceglie quindi come numero di vicini 15 e si riaddestra il modello.

```
Parametri attuali: n_neighbors = 15, weights = uniform
% ----- KNN test ----- %
Average Recall: 0.5493275641935026
Average Precision: 0.5744791096768073
Average Accuracy: 0.5705977514407488
Average F1-score: 0.5612086388360702

% ----- KNN train ----- %
Average Recall: 0.6198540727169004
Average Precision: 0.6516708040802128
Average Accuracy: 0.6442645137350688
Average F1-score: 0.6353488506331233

% ----- %

Performance del modello con parametri della Grid Search
+-----+-----+-----+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+-----+-----+-----+
| KNN df oversampled | 0.610694 | 0.73156 | 0.692782 | 0.665407 |
+-----+-----+-----+-----+

Performance del modello con parametri aggiornati
+-----+-----+-----+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+-----+-----+-----+
| KNN df oversampled | 0.549328 | 0.574479 | 0.570598 | 0.561209 |
+-----+-----+-----+-----+
```

Come ipotizzato, e' possibile notare una **minor differenza tra gli score** delle performance generati dalla valutazione del modello **sui dati di train** e quelli generati **sui dati di test**, rispetto al caso precedente.

Questo indica una **netta riduzione di adattamento del modello ai dati**. Da notare che **non può esserci un'assenza di differenza tra questi valori** in quanto è logico che testare un modello sui dati su cui esso stesso è stato addestrato porterà sempre a performance migliori, l'obiettivo è appunto cercare di minimizzare questo gap.

Va inoltre notato che in linea generale **le performance calcolate sui dati di test sono rimaste pressochè invariate rispetto al caso precedente**, questo può significare che un aumento del dataset al fine del ribilanciamento della classe minoritaria non ha portato all'ottenimento dei risultati sperati.

Possiamo quindi concludere che **il modello attualmente presenta una forma leggera di overfitting e restituisce performance peggiori rispetto al modello che utilizza il dataset originale**.

Random forest con dataset originale

Valutazione performance e presenza overfitting

A seguito dell'addestramento del RF attraverso l'utilizzo della grid search e della cross validation i risultati sono stati i seguenti:

```
Parametri ottimali: {'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 50}

% ----- RF test -----
Average Recall: 0.9529365867498936
Average Precision: 0.993016738339826
Average Accuracy: 0.9665000934569165
Average F1-score: 0.9724786432283716

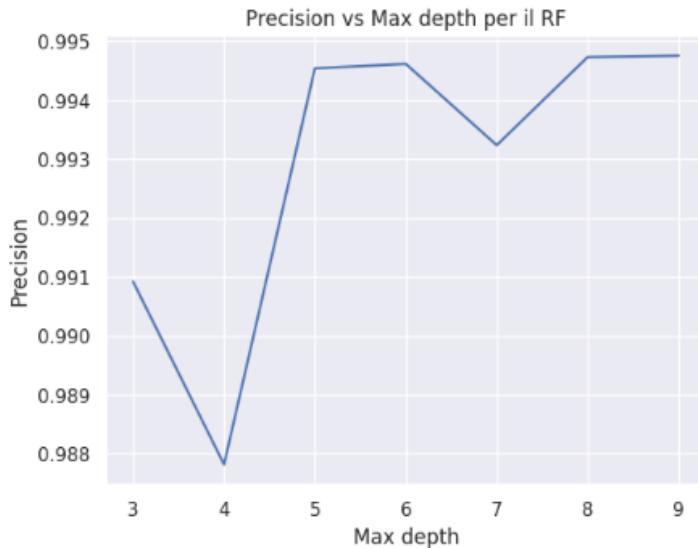
% ----- RF train -----
Average Recall: 0.9624329380224657
Average Precision: 0.9940404138480703
Average Accuracy: 0.9730354398154638
Average F1-score: 0.9779665145377375

% ----- %

+-----+-----+-----+-----+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+-----+-----+-----+-----+
| RF df originale | 0.952937 | 0.993017 | 0.9665 | 0.972479 |
+-----+-----+-----+-----+
```

E' possibile notare una **lieve, e trascurabile, differenza tra gli score** delle performance generati dalla valutazione del modello **sui dati di train** e quelli generati **sui dati di test**. Questo è un segno di assenza di **overfitting** perciò nei passi seguenti si approfondirà tale tesi.

Si procede quindi con il calcolo e la conseguente stampa della curva **precision vs max depth**, per capire come influisce la variazione della profondità degli alberi sullo score della precision.



Dal grafico è possibile osservare che al variare della profondità degli alberi nel random forest non segue una particolare variazione dello score della precision. Ciò potrebbe indicare che **la situazione potrebbe rimanere pressochè simile a quella attuale anche in un modello più complesso, ovvero una assenza di overfitting.**

Per verificare ciò si aumenta il valore di profondità a 9 e si riaddestra il modello.

```
Parametri attuali : max_depth = 9, min_samples_leaf = 1, min_samples_split = 5, n_estimators = 50,
% ----- RF test -----
Average Recall: 0.9841835721378919
Average Precision: 0.9810779329597817
Average Accuracy: 0.9782129938098537
Average F1-score: 0.9825549046678074

% ----- RF train -----
Average Recall: 0.9991214964065034
Average Precision: 0.9950431154285706
Average Accuracy: 0.9963561729155128
Average F1-score: 0.9970776959881604

% ----- %

Performance del modello con parametri della Grid Search
+-----+-----+-----+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+-----+-----+-----+
| RF df originale | 0.952937 | 0.993017 | 0.9665 | 0.972479 |
+-----+-----+-----+-----+

Performance del modello con parametri aggiornati
+-----+-----+-----+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+-----+-----+-----+
| RF df originale | 0.984184 | 0.981078 | 0.978213 | 0.982555 |
+-----+-----+-----+-----+
```

Come ipotizzato, è possibile notare una **minimo aumento della differenza tra gli score delle performance generati dalla valutazione del modello sui dati di train e quelli generati sui dati di test,** rispetto al caso precedente.

Questo indica un **lieve aumento di adattamento del modello ai dati**. Da notare che **non può esserci un'assenza di differenza tra questi valori** in quanto è logico che testare un modello sui dati su cui esso stesso è stato addestrato porterà sempre a performance migliori, l'obiettivo è appunto cercare di minimizzare questo gap.

Possiamo quindi concludere che **il modello attualmente non presenta overfitting o questo è in una forma così lieve da essere trascurabile**.

Le performance risultano essere leggermente migliorate nel complesso.

Random forest con dataset undersampled

Valutazione performance e presenza overfitting

A seguito dell'addestramento del RF attraverso l'utilizzo della grid search, del dataset undersampled e della cross validation i risultati sono stati i seguenti:

```
Parametri ottimali: {'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 50}

% ----- RF test ----- %

Average Recall: 0.9529365867498936
Average Precision: 0.993016738339826
Average Accuracy: 0.9665000934569165
Average F1-score: 0.9724786432283716

% ----- RF train ----- %

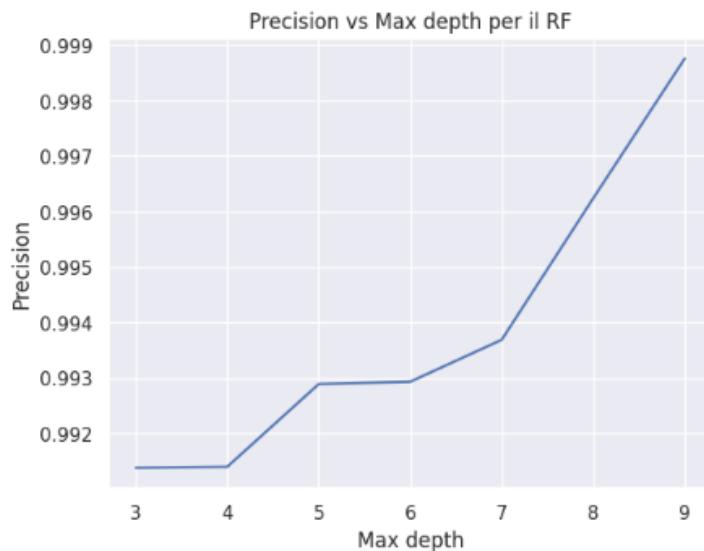
Average Recall: 0.9624329380224657
Average Precision: 0.9940404138480703
Average Accuracy: 0.9730354398154638
Average F1-score: 0.9779665145377375

% ----- %

+-----+-----+-----+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+-----+-----+-----+
| RF df originale | 0.952937 | 0.993017 | 0.9665 | 0.972479 |
+-----+-----+-----+-----+
```

E' possibile notare una **lieve, e trascurabile, differenza tra gli score** delle performance generati dalla valutazione del modello **sui dati di train** e quelli generati **sui dati di test**. Questo è un segno di assenza di **overfitting** perciò nei passi seguenti si approfondirà tale tesi.

Si procede quindi con il calcolo e la conseguente stampa della curva **precision vs max depth**, per capire come influisce la variazione della profondità degli alberi sullo score della precision.



Dal grafico è possibile osservare che al variare della profondità degli alberi nel random forest non segue una particolare variazione dello score della precision. Ciò potrebbe indicare che **la situazione potrebbe rimanere pressochè simile a quella attuale anche in un modello più complesso, ovvero una assenza di overfitting**.

Per verificare ciò aumentiamo il valore di profondità a 9 e si riaddestra il modello.

```
Parametri attuali : max_depth = 9, min_samples_leaf = 1, min_samples_split = 5, n_estimators = 50,  
% ----- RF test ----- %  
Average Recall: 0.979007553441767  
Average Precision: 0.9843850670454044  
Average Accuracy: 0.9773676518662384  
Average F1-score: 0.981650278548319  
% ----- RF train ----- %  
Average Recall: 0.9991083086777502  
Average Precision: 0.9990513548471494  
Average Accuracy: 0.9988633991244928  
Average F1-score: 0.9990795608687504  
% ----- %  
Performance del modello con parametri della Grid Search  
+-----+-----+-----+-----+  
| Modello | Recall | Precision | Accuracy | F1-score |  
+-----+-----+-----+-----+  
| RF df undersampled | 0.952862 | 0.99169 | 0.965899 | 0.971822 |  
+-----+-----+-----+-----+  
Performance del modello con parametri aggiornati  
+-----+-----+-----+-----+  
| Modello | Recall | Precision | Accuracy | F1-score |  
+-----+-----+-----+-----+  
| RF df undersampled | 0.979008 | 0.984385 | 0.977368 | 0.98165 |  
+-----+-----+-----+-----+
```

Come ipotizzato, è possibile notare una **minimo aumento della differenza tra gli score delle performance** generati dalla valutazione del modello **sui dati di train** e quelli generati **sui dati di test**, rispetto al caso precedente.

Questo indica un **lieve aumento di adattamento del modello ai dati**. Da notare che **non può esserci un'assenza di differenza tra questi valori** in quanto è logico che testare un modello sui dati su cui esso stesso è stato addestrato porterà sempre a performance migliori, l'obiettivo è appunto cercare di minimizzare questo gap.

Possiamo quindi concludere che **il modello attualmente non presenta overfitting o questo è in una forma così lieve da essere trascurabile**.

Le performance risultano essere leggermente migliorate nel complesso.

Random forest con dataset oversampled

Valutazione performance e presenza overfitting

A seguito dell'addestramento del RF attraverso l'utilizzo della grid search, del dataset oversampled e della cross validation i risultati sono stati i seguenti:

```
Parametri ottimali: {'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}

% ----- RF test ----- %

Average Recall: 0.94126968364307
Average Precision: 0.9933011280210978
Average Accuracy: 0.9674298741203291
Average F1-score: 0.9664956597899172

% ----- RF train ----- %

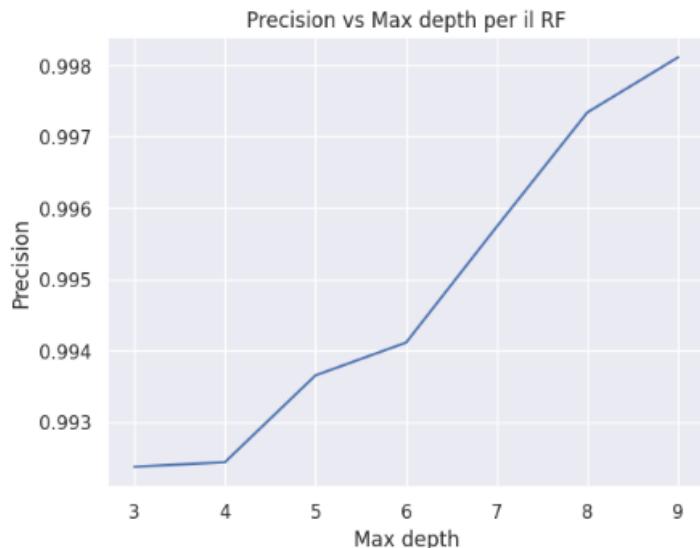
Average Recall: 0.9467035903466451
Average Precision: 0.993764927653366
Average Accuracy: 0.9703815661540318
Average F1-score: 0.9696593998906572

% ----- %

+-----+-----+-----+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+-----+-----+-----+
| RF df oversampled | 0.94127 | 0.993301 | 0.96743 | 0.966496 |
+-----+-----+-----+-----+
```

E' possibile notare una **lieve, e trascurabile, differenza tra gli score** delle performance generati dalla valutazione del modello **sui dati di train** e quelli generati **sui dati di test**. Questo è un segno di assenza di **overfitting** perciò nei passi seguenti si approfondirà tale tesi.

Si procede quindi con il calcolo e la conseguente stampa della curva **precision vs max depth**, per capire come influisce la variazione della profondità degli alberi sullo score della precision.



Dal grafico è possibile osservare che al variare della profondità degli alberi nel random forest non segue una particolare variazione dello score della precision. Ciò potrebbe indicare che **la situazione potrebbe rimanere pressochè simile a quella attuale anche in un modello più complesso, ovvero una assenza di overfitting**.

Per verificare ciò aumentiamo il valore di profondità a 9 e si riaddestra il modello.

```
Parametri attuali : max_depth = 9, min_samples_leaf = 1, min_samples_split = 5, n_estimators = 50,  
% ----- RF test ----- %  
  
Average Recall: 0.9638587033621789  
Average Precision: 0.9926595561912801  
Average Accuracy: 0.9783491213910482  
Average F1-score: 0.977997992757515  
  
% ----- RF train ----- %  
  
Average Recall: 0.9946031054389806  
Average Precision: 0.9986980147423201  
Average Accuracy: 0.9966532885138962  
Average F1-score: 0.9966448330203977  
  
% ----- %  
  
Performance del modello con parametri della Grid Search  
+-----+-----+-----+-----+  
| Modello | Recall | Precision | Accuracy | F1-score |  
+-----+-----+-----+-----+  
| RF df oversampled | 0.94127 | 0.993301 | 0.96743 | 0.966496 |  
+-----+-----+-----+-----+  
  
Performance del modello con parametri aggiornati  
+-----+-----+-----+-----+  
| Modello | Recall | Precision | Accuracy | F1-score |  
+-----+-----+-----+-----+  
| RF df oversampled | 0.963859 | 0.99266 | 0.978349 | 0.977998 |  
+-----+-----+-----+-----+
```

Come ipotizzato, e' possibile notare una **minimo aumento della differenza tra gli score** delle performance generati dalla valutazione del modello **sui dati di train** e quelli generati **sui dati di test**, rispetto al caso precedente.

Questo indica un **lieve aumento di adattamento del modello ai dati**. Da notare che **non può esserci un'assenza di differenza tra questi valori** in quanto è logico che testare un modello sui dati su cui esso stesso è stato addestrato porterà sempre a performance migliori, l'obiettivo è appunto cercare di minimizzare questo gap.

Possiamo quindi concludere che **il modello attualmente non presenta overfitting o questo è in una forma così lieve da essere trascurabile**.

Le performance risultano essere leggermente migliorate nel complesso.

Multilayer Perceptron con dataset originale

Valutazione performance e presenza overfitting

A seguito dell'addestramento del MLP attraverso l'utilizzo della grid search e della cross validation i risultati sono stati i seguenti:

```
Parametri ottimali: {'activation': 'relu', 'alpha': 0.0001, 'hidden_layer_sizes': (50, 50)}
```

```
% ----- MLP test ----- %

Average Recall: 0.6006908781387431
Average Precision: 0.7056338159490295
Average Accuracy: 0.5305763543006673
Average F1-score: 0.5190660886200477

% ----- MLP train ----- %

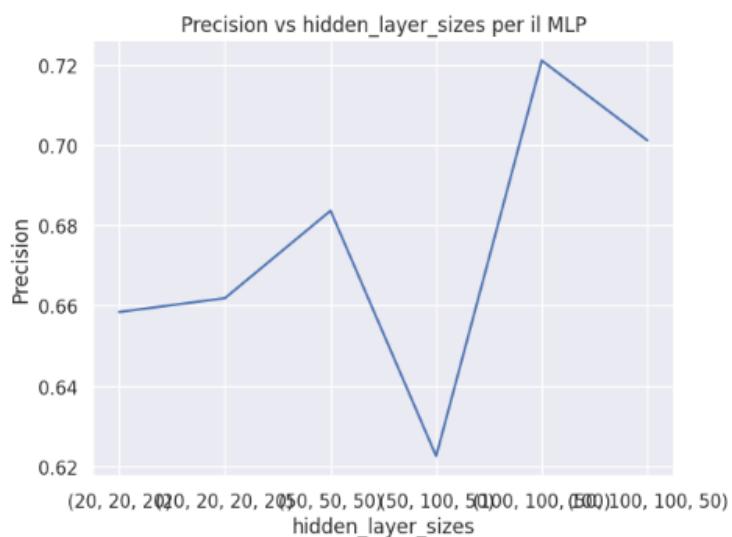
Average Recall: 0.5998393032449089
Average Precision: 0.6562451211892012
Average Accuracy: 0.5314020922455737
Average F1-score: 0.5168478699940511

% ----- %

+-----+-----+-----+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+-----+-----+-----+
| MLP df originale | 0.600691 | 0.705634 | 0.530576 | 0.519066 |
+-----+-----+-----+-----+
```

E' possibile notare una **lieve, e trascurabile, differenza tra gli score** delle performance generati dalla valutazione del modello **sui dati di train** e quelli generati **sui dati di test**. Questo è un segno di assenza di **overfitting** perciò nei passi seguenti si approfondirà tale tesi. Inoltre tutte le performance non sono molto positive, perciò si proverà a rendere il modello più complesso per cercare di aumentare i vari score senza far adattare troppo il modello ai dati.

Si procede quindi con il calcolo e la conseguente stampa della curva **precision vs hidden layer sizes**, per capire come influisce il numero di layer nascosti e di neuroni di questi sullo score della precision.



Dal grafico è possibile osservare che al variare del numero e della complessità dei layer nascosti anche la precision tende a variare. Perciò si prova ad usare uno di questi valori **dell'iperparametro "hidden_layer_sizes"** e riaddestrare il modello aumentandone un pò la complessità.

Per verificare ciò si sceglie di utilizzare 3 layer, entrambi da 50 neuroni e si riaddestra il modello.

```
Parametri attuali: hidden_layer_sizes = (50, 50, 50), activation = relu, alpha = 0.0001
% ----- MLP test -----
Average Recall: 0.9084494254504184
Average Precision: 0.6263330590834796
Average Accuracy: 0.6050598674011282
Average F1-score: 0.7380662524885307

% ----- MLP train -----
Average Recall: 0.917970790044256
Average Precision: 0.6297820130942927
Average Accuracy: 0.61161883874397
Average F1-score: 0.7434968854358901

% ----- %

Performance del modello con parametri della Grid Search
+-----+-----+-----+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+-----+-----+-----+
| MLP df originale | 0.600691 | 0.705634 | 0.530576 | 0.519066 |
+-----+-----+-----+-----+

Performance del modello con parametri aggiornati
+-----+-----+-----+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+-----+-----+-----+
| MLP df originale | 0.908449 | 0.626333 | 0.60506 | 0.738066 |
+-----+-----+-----+-----+
```

E' possibile notare una **assenza di aumento della differenza tra gli score** delle performance generati dalla valutazione del modello **sui dati di train** e quelli generati **sui dati di test**, rispetto al caso precedente.

Questo indica un'assenza **di adattamento del modello ai dati**. Da notare che **non può esserci un'assenza di differenza tra questi valori** in quanto è logico che testare un modello sui dati su cui esso stesso è stato addestrato porterà sempre a performance migliori, l'obiettivo è appunto cercare di minimizzare questo gap.

L'obiettivo di aumentare la complessità per migliorare le performance risulta raggiunto, infatti **le performance risultano essere migliorate nel complesso**.

Possiamo quindi concludere che **il modello attualmente non presenta overfitting o questo è in una forma così lieve da essere trascurabile**.

Multilayer Perceptron con dataset undersampled

Valutazione performance e presenza overfitting

A seguito dell'addestramento del MLP attraverso l'utilizzo della grid search, dataset undersampled e della cross validation i risultati sono stati i seguenti:

```
Parametri ottimali: {'activation': 'relu', 'alpha': 0.001, 'hidden_layer_sizes': (50,)}

% ----- MLP test ----- %

Average Recall: 0.5268959435626102
Average Precision: 0.5806273777521744
Average Accuracy: 0.5337865123165972
Average F1-score: 0.48601324725725165

% ----- MLP train ----- %

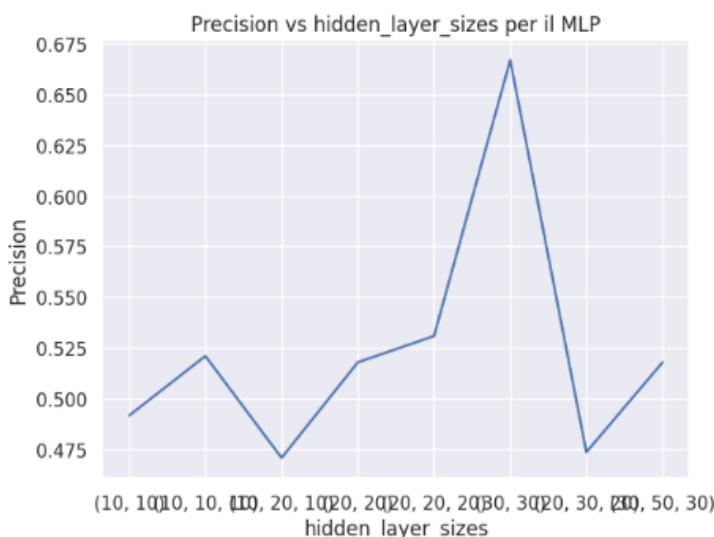
Average Recall: 0.5213909187736016
Average Precision: 0.537901685718544
Average Accuracy: 0.5268312845360883
Average F1-score: 0.4728800533020173

% ----- %

+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+
| MLP df undersampled | 0.526896 | 0.580627 | 0.533787 | 0.486013 |
+-----+
```

E' possibile notare una **leggera differenza tra gli score** delle performance generati dalla valutazione del modello **sui dati di train** e quelli generati **sui dati di test**. Questo è un segno di assenza di **un leggero overfitting** perciò nei passi seguenti si approfondirà tale tesi. Inoltre tutte le performance non sono molto positive, perciò si proverà a rendere il modello più complesso per cercare di aumentare i vari score senza far adattare troppo il modello ai dati.

Si procede quindi con il calcolo e la conseguente stampa della curva **precision vs hidden layer sizes**, per capire come influisce il numero di layer nascosti e di neuroni di questi sullo score della precision.



Dal grafico è possibile osservare che al variare del numero e della complessità dei layer nascosti anche la precision tende a variare. Perciò si prova ad usare uno di questi valori **dell'iperparametro "hidden_layer_sizes"** e **riaddestrare il modello aumentandone un pò la complessità**.

Per verificare ciò si sceglie di utilizzare 2 layer, entrambi da 30 neuroni e si riaddestra il modello.

```
Parametri attuali: hidden_layer_sizes = (30, 30), activation = relu, alpha = 0.0001
% ----- MLP test -----
Average Recall: 0.5727206502568821
Average Precision: 0.5091799530162716
Average Accuracy: 0.5040218833528833
Average F1-score: 0.4291541782037179

% ----- MLP train -----
Average Recall: 0.5714140812928483
Average Precision: 0.5063372286226256
Average Accuracy: 0.5041678054145564
Average F1-score: 0.4311078258480773

% -----
Performance del modello con parametri della Grid Search
+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+
| MLP df undersampled | 0.526896 | 0.580627 | 0.533787 | 0.486013 |
+-----+

Performance del modello con parametri aggiornati
+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+
| MLP df undersampled | 0.572721 | 0.50918 | 0.504022 | 0.429154 |
+-----+
```

E' possibile notare una **assenza di aumento della differenza tra gli score** delle performance generati dalla valutazione del modello **sui dati di train** e quelli generati **sui dati di test**, rispetto al caso precedente.

Questo indica **un'assenza di adattamento del modello ai dati**. Da notare che **non può esserci un'assenza di differenza tra questi valori** in quanto è logico che testare un modello sui dati su cui esso stesso è stato addestrato porterà sempre a performance migliori, l'obiettivo è appunto cercare di minimizzare questo gap.

L'obiettivo di aumentare la complessità per migliorare le performance **NON** risulta raggiunto, **infatti le performance risultano essere circa invariate nel complesso**.

Possiamo quindi concludere che **il modello attualmente non presenta overfitting o questo è in una forma così lieve da essere trascurabile**.

Si potrebbe quindi procedere con nuove valutazioni empiriche sull'aumento della complessità ai fini dell'aumento degli score delle performance, ma per questioni di tempo in questo caso di studio ciò non verrà approfondito.

Multilayer Perceptron con dataset oversampled

Valutazione performance e presenza overfitting

A seguito dell'addestramento del MLP attraverso l'utilizzo della grid search, dataset oversampled e della cross validation i risultati sono stati i seguenti:

```
Parametri ottimali: {'activation': 'relu', 'alpha': 0.001, 'hidden_layer_sizes': (50, 50)}
```

```
% ----- MLP test ----- %

Average Recall: 0.47861682508157183
Average Precision: 0.6230352402782042
Average Accuracy: 0.5096041657816859
Average F1-score: 0.3781044982740147

% ----- MLP train ----- %

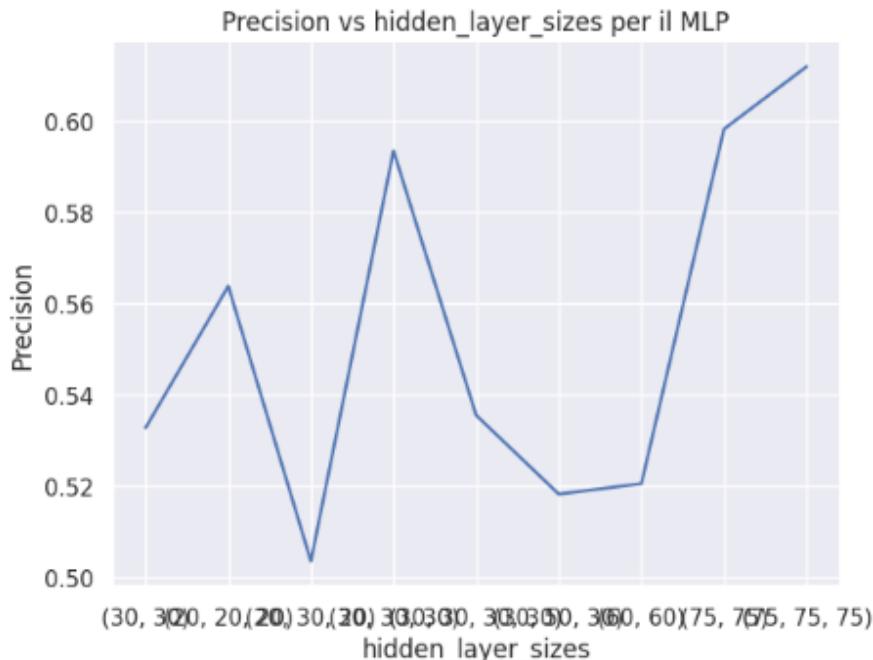
Average Recall: 0.4913293399760959
Average Precision: 0.5961048542809779
Average Accuracy: 0.5162522983672295
Average F1-score: 0.3878958268305256

% ----- %

+-----+-----+-----+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+-----+-----+-----+
| MLP df oversampled | 0.478617 | 0.623035 | 0.509604 | 0.378104 |
+-----+-----+-----+-----+
```

E' possibile notare una **lieve, e trascurabile, differenza tra gli score** delle performance generati dalla valutazione del modello **sui dati di train** e quelli generati **sui dati di test**. Questo è un segno di assenza di **overfitting** perciò nei passi seguenti si approfondirà tale tesi. Ioltre tutte le performance non sono molto positive, perciò si proverà a rendere il modello più complesso per cercare di aumentare i vari score senza far adattare troppo il modello ai dati.

Si procede quindi con il calcolo e la conseguente stampa della curva **precision vs hidden layer sizes**, per capire come influisce il numero di layer nascosti e di neuroni di questi sullo score della precision.



Dal grafico è possibile osservare che al variare del numero e della complessità dei layer nascosti anche la precision tende a variare. Perciò si prova ad usare uno di questi valori **dell'iperparametro "hidden_layer_sizes"** e **riaddestrare il modello aumentandone un pò la complessità**.

Per verificare ciò si sceglie di utilizzare 3 layer, entrambi da 75 neuroni e si riaddestra il modello.

```
Parametri ottimali: hidden_layer_sizes = (75, 75, 75), activation = relu, alpha = 0.0001
%
% ----- MLP test ----- %
Average Recall: 0.478611150517804
Average Precision: 0.5925354856842129
Average Accuracy: 0.518073786160316
Average F1-score: 0.38528912222715594
%
% ----- MLP train ----- %
Average Recall: 0.4810409852847761
Average Precision: 0.5889827551825922
Average Accuracy: 0.5207699584915535
Average F1-score: 0.3833695606082932
%
% ----- %
Performance del modello con parametri della Grid Search
+-----+-----+-----+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+-----+-----+-----+
| MLP df oversampled | 0.478617 | 0.623035 | 0.509604 | 0.378104 |
+-----+-----+-----+-----+
Performance del modello con parametri aggiornati
+-----+-----+-----+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+-----+-----+-----+
| MLP df oversampled | 0.478611 | 0.592535 | 0.518074 | 0.385289 |
+-----+-----+-----+-----+
```

E' possibile notare una **assenza di aumento della differenza tra gli score** delle performance generati dalla valutazione del modello **sui dati di train** e quelli generati **sui dati di test**, rispetto al caso precedente.

Questo indica un'assenza **di adattamento del modello ai dati**. Da notare che **non può esserci un'assenza di differenza tra questi valori** in quanto è logico che testare un modello sui dati su cui esso stesso è stato addestrato porterà sempre a performance migliori, l'obiettivo è appunto cercare di minimizzare questo gap.

L'obiettivo di aumentare la complessità per migliorare le performance **NON** risulta raggiunto, infatti **le performance risultano essere circa invariate nel complesso**.

Possiamo quindi concludere che **il modello attualmente non presenta overfitting o questo è in una forma così lieve da essere trascurabile**.

Si potrebbe quindi procedere con nuove valutazioni empiriche sull'aumento della complessità ai fini dell'aumento degli score delle performance, ma per questioni di tempo in questo caso di studio ciò non verrà approfondito.

Apprendimento probabilistico

L'**apprendimento probabilistico** è un approccio di apprendimento automatico che si basa sulla teoria delle probabilità per costruire modelli che rappresentano le relazioni tra variabili. Questi modelli descrivono l'incertezza nei dati e consentono di fare previsioni informate considerando la probabilità degli eventi.

In questo capitolo si utilizzerà il modello probabilistico Naive Bayes. Il **Naive Bayes** è un tipo di algoritmo di classificazione probabilistica basato sul teorema di Bayes, che si basa sull'assunzione "naive" (semplice) di indipendenza condizionale tra le caratteristiche. In altre parole, l'algoritmo assume che le caratteristiche siano indipendenti tra loro dato il valore della variabile di classe. Questa è spesso un'assunzione semplificatrice, ma può funzionare sorprendentemente bene in molte situazioni. Perciò si verifica se nel contesto di studio questo può essere un approccio vincente.

Decisione progettuale

Nella fase di valutazione delle performance e presenza overfitting si sceglie di variare il valore del parametro "var_smoothing" perchè è l'unico che avesse un senso cambiare, altresì si sarebbe potuto scegliere un parametro diverso per le probabilità a priori della classe di appartenenza calcolate sulle features ma in realtà è un parametro che viene calcolato in automatico e non ha senso modificare.

Naive Bayes

Come per gli altri modelli visti finora si parte dall'addestramento del modello attraverso l'utilizzo della grid search e della cross validation, si segue con il calcolo delle performance e il test di eventuali migliorie al modello.

Valutazione performance e presenza overfitting

A seguito dell'addestramento del MLP attraverso l'utilizzo della grid search e della cross validation i risultati sono stati i seguenti:

```
Parametri ottimali: {'var_smoothing': 1e-09}

% ----- NB test ----- %

Average Recall: 0.9702596112923819
Average Precision: 0.7302828783379078
Average Accuracy: 0.7584875372453299
Average F1-score: 0.8333143922931445

% ----- NB train ----- %

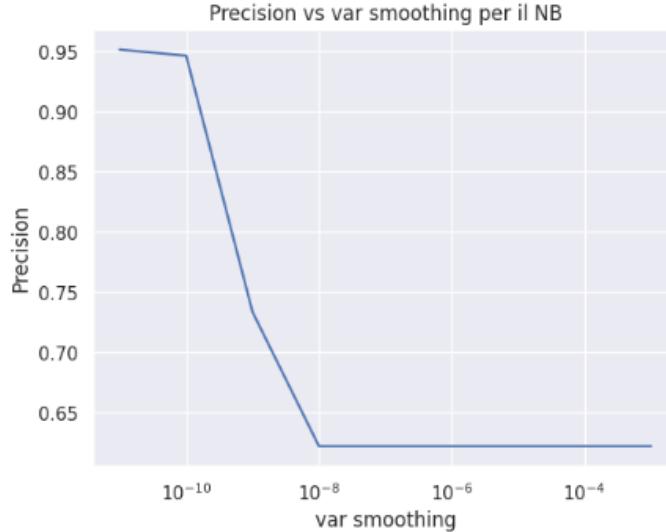
Average Recall: 0.9731007666475924
Average Precision: 0.735813643038181
Average Accuracy: 0.7658831209837772
Average F1-score: 0.8379804415874602

% ----- %

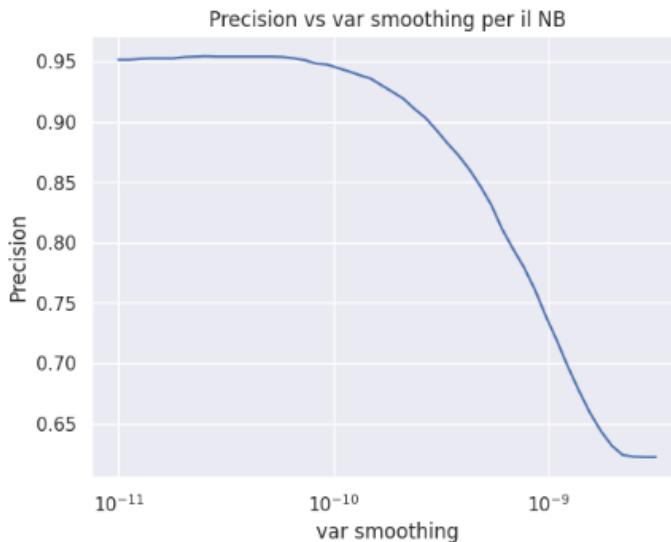
+-----+-----+-----+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+-----+-----+-----+
| NB df originale | 0.97026 | 0.730283 | 0.758488 | 0.833314 |
+-----+-----+-----+-----+
```

E' possibile notare una **lieve, e trascurabile, differenza tra gli score** delle performance generati dalla valutazione del modello **sui dati di train** e quelli generati **sui dati di test**. Questo è un segno di **assenza di overfitting** perciò nei passi seguenti si approfondirà tale tesi.

Si procede quindi con il calcolo e la conseguente stampa della curva **precision vs var_smoothing**, per capire come influisce la variazione di tale parametro sullo score della precision.



Il grafico mostra un aumento netto dello score della precisione tra $1e-11$ ed $1e-8$, perciò si procede con il ricalcolo del grafico concentrandosi soltanto su questa porzione.



Dal grafico è possibile osservare che al diminuire del parametro in questione aumenta il valore della precision. Proviamo quindi a diminuire tale valore per capire come si comporta il modello, si sceglie **var_smoothing = 1e-10**.

```

Parametri aggiornati: var_smoothing = 1e-10

% ----- NB test ----- %

Average Recall: 0.9356192367711731
Average Precision: 0.9434472601083519
Average Accuracy: 0.9248040153489244
Average F1-score: 0.939359912978907

% ----- NB train ----- %

Average Recall: 0.9359521847093966
Average Precision: 0.945969396126328
Average Accuracy: 0.9268889208567994
Average F1-score: 0.9409323265979094

% ----- %

Performance del modello con parametri della Grid Search
+-----+-----+-----+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+-----+-----+-----+
| NB df originale | 0.97026 | 0.730283 | 0.758488 | 0.833314 |
+-----+-----+-----+-----+

Performance del modello con parametri aggiornati
+-----+-----+-----+-----+
| Modello | Recall | Precision | Accuracy | F1-score |
+-----+-----+-----+-----+
| NB df originale | 0.935619 | 0.943447 | 0.924804 | 0.93936 |
+-----+-----+-----+-----+

```

E' possibile notare una **assenza di aumento della differenza tra gli score** delle performance generati dalla valutazione del modello **sui dati di train** e quelli generati **sui dati di test**, rispetto al caso precedente.

Questo indica un di **assenza di aumento di adattamento del modello ai dati**. Da notare che **non può esserci un'assenza di differenza tra questi valori** in quanto è logico che testare un modello sui dati su cui esso stesso è stato addestrato porterà sempre a performance migliori, l'obiettivo è appunto cercare di minimizzare questo gap.

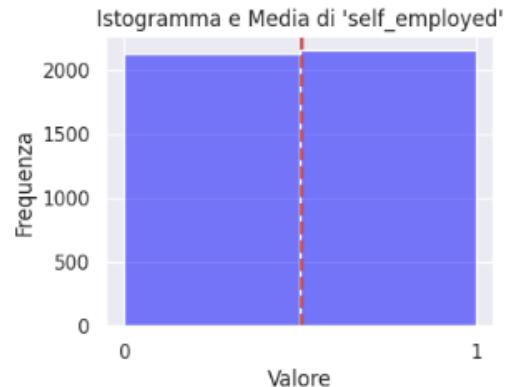
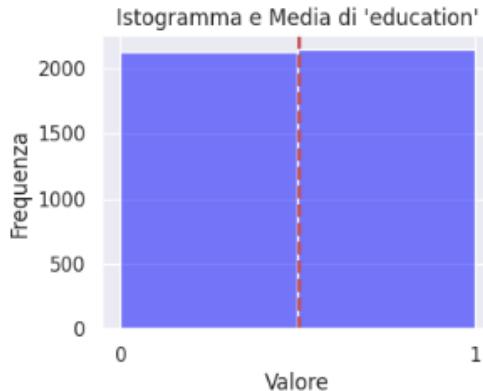
Le performance risultano essere decisamente migliorate nel complesso.

Possiamo quindi concludere che **il modello attualmente non presenta overfitting o questo è in una forma così lieve da essere trascurabile**.

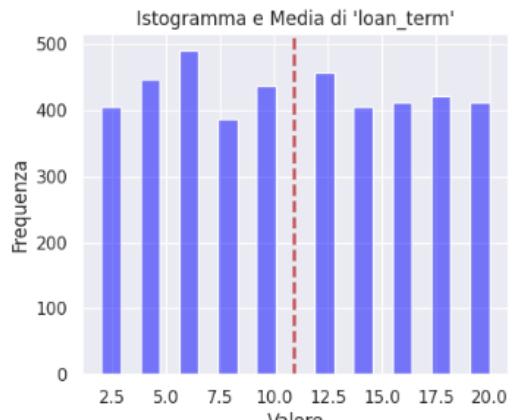
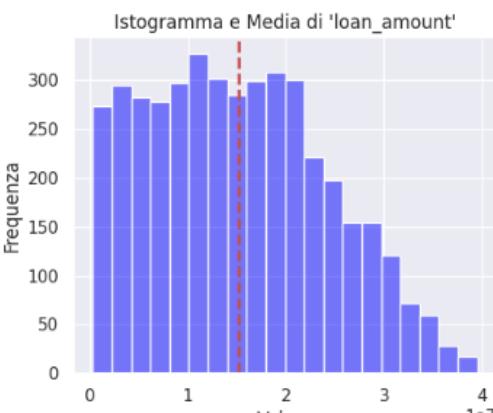
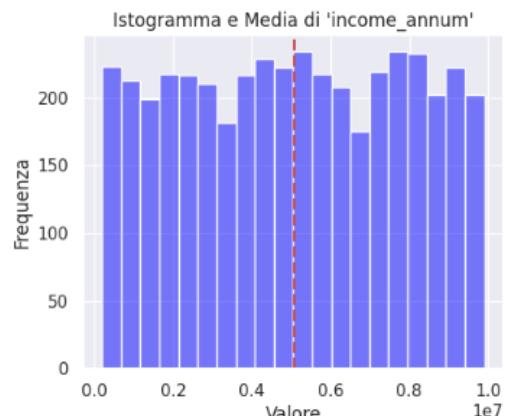
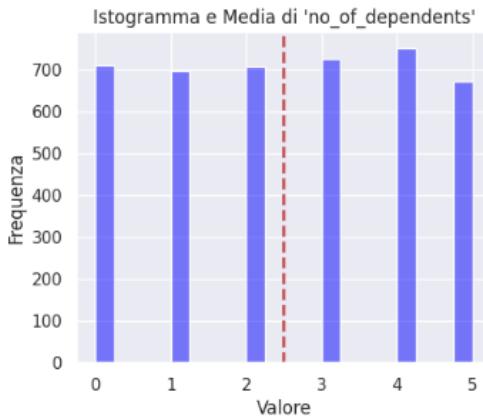
Osservazioni

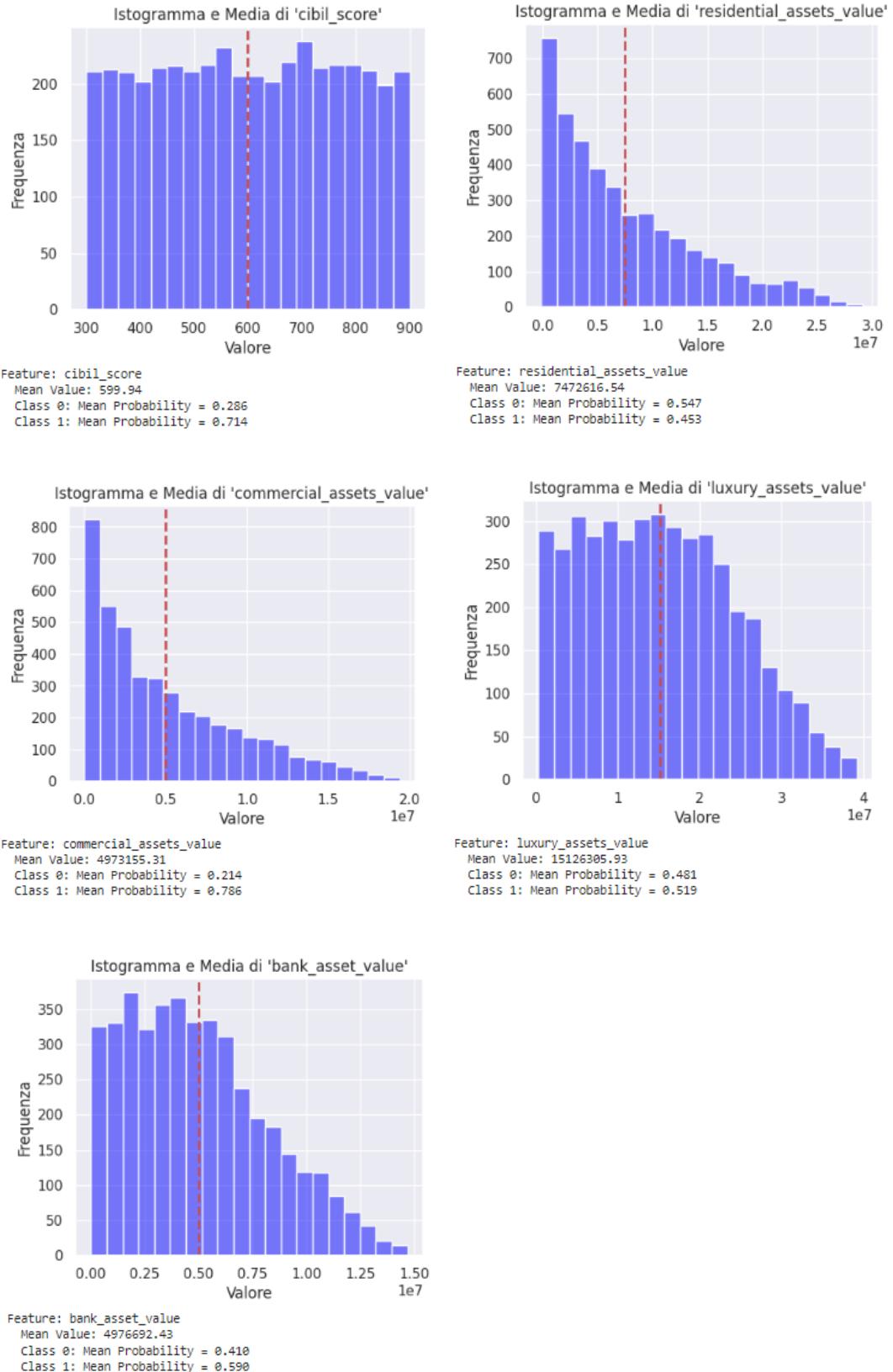
Si è deciso di approfondire questo modello con la verifica della relazione che c'è tra la composizione degli esempi per ogni feature e la media delle predizioni effettuate dal modello per la classe di approvazione del prestito per ogni feature. Di seguito sono mostrati i relativi grafici:

Features binarie



Features continue





I grafici qui sopra mostrati possono sembrare banali e non informativi, in realtà tutt'altro. In particolare possiamo notare che:

1. C'è un **ottimo bilanciamento** tra i valori delle features degli esempi per diverse features
2. Le probabilità medie stimate per alcune feature in riferimento all'appartenenza degli esempi ad una classe sembrano **indicativi della composizione degli esempi della feature stessa**.

Queste due osservazioni portano a pensare che i valori spartiacque che potrebbe utilizzare un istituto bancario per la decisione dell'approvazione di un mutuo o meno sono esattamente i valori medi per ogni feature (che in alcuni casi corrisponde alla media).

Si proverà quindi successivamente a costruire una condizione basata sui valori medi delle feature le quali hanno un valore simile tra media e valore medio, ovvero le seguenti :

- income_ annum
- loan_ amount
- residential_asset_value
- luxury_asset_value
- bank_asset_value

Le osservazioni fatte in questo blocco verranno tenute in considerazione nel momento della creazione delle regole della base di conoscenza.

Discussione risultati

Di seguito sono riportati i risultati ottenuti da tutti i vari modelli visti finora comprese le versioni undersampled ed oversampled

	Modello	Recall	Precision	Accuracy	F1-score
0	KNN df originale	0.849772	0.625925	0.590544	0.720811
1	KNN df undersampled	0.538785	0.51107	0.512076	0.524238
2	KNN df oversampled	0.549328	0.574479	0.570598	0.561209
3	RF df originale	0.984184	0.981078	0.978213	0.982555
4	RF df undersampled	0.979008	0.984385	0.977368	0.98165
5	RF df oversampled	0.963859	0.99266	0.978349	0.977998
6	MLP df originale	0.908449	0.626333	0.60506	0.738066
7	MLP df undersampled	0.572721	0.50918	0.504022	0.429154
8	MLP df oversampled	0.478611	0.592535	0.518074	0.385289
9	NB df originale	0.935619	0.943447	0.924804	0.93936

Conclusioni finali sul capitolo dedicato all'apprendimento supervisionato di modelli classici e del modello probabilistico Naive Bayes:

1. I migliori classificatori per questo dataset sono stati il **Random Forest** e il **Naive Bayes**.
2. Effettuare l'**oversampled** o l'**undersampled** del dataset non ha portato quasi mai ad un **miglioramento delle performance** come ipotizzato.

- I modelli **KNN** ed **MLP** portano risultati discreti solo nella loro versione con il dataset originale, questo potrebbe indicare che non sono stati dati alla grid search i giusti parametri da testare.

Rappresentazione della conoscenza tramite Knowledge Base

In questo capitolo ci si concentrerà sulla rappresentazione della conoscenza, acquisita a priori e tramite i capitoli precedenti basati sull'apprendimento supervisionato. Per rappresentare la conoscenza si utilizzerà una Knowledge Base creata tramite l'utilizzo di Prolog.

Una knowledge base è una raccolta strutturata di informazioni, dati, fatti e relazioni che vengono utilizzati per supportare il processo decisionale e il ragionamento di un sistema intelligente o di un'applicazione software.

Prolog è un linguaggio di programmazione logica che si presta particolarmente bene per rappresentare e manipolare conoscenze sotto forma di fatti e regole logiche.

Decisione progettuale

Si è deciso di sviluppare questa parte tra gli ultimi capitoli di questo caso di studio in quanto l'obiettivo scelto è stato quello di confrontare due rappresentazioni della realtà diverse. In particolare si utilizzerà la stessa KB ma la prima rappresentazione farà uso solo di regole definite a priori, a differenza della seconda rappresentazione che farà uso dell'unica regola creata in base alle osservazioni fatte nella fase di apprendimento supervisionato. Si procederà quindi con l'interrogazione della KB per la predizione dei prestiti approvati, utilizzando entrambe le rappresentazioni, e si confronteranno le performance di queste per identificare la rappresentazione migliore.

Creazione della knowledge base

Si è deciso di rappresentare la conoscenza attraverso la definizione dei seguenti predicati rappresentanti fatti:

- richiesta(Loan_id, No_of_dependents, Education, Self_employed, Income_anum, Loan_amount, Loan_term, Cibil_score, Residential_assets_value, Commercial_assets_value, Luxury_assets_value, Bank_asset_value, Loan_status).
- attivo(Loan_id, Income_anum, Commercial_assets_value, Bank_asset_value).
- garanzie(Loan_id, Residential_assets_value, Commercial_assets_value, Luxury_assets_value, Bank_asset_value).
- passivo(Loan_id, No_of_dependents, Loan_amount, Luxury_assets_value).
- solvibilita(Loan_id, Loan_amount, Loan_term, Cibil_score).

e le seguenti regole:

```
approva_per_attivo(Loan_id) :-  
    attivo(Loan_id, Income_anum, Commercial_assets_value, Bank_asset_value),  
    passivo(Loan_id, _, Loan_amount, _),  
    Commercial_assets_value + Income_anum > Loan_amount,  
    Bank_asset_value > 0.1*Loan_amount.
```

```

approva_per_reddito(Loan_id) :-
    attivo(Loan_id, Income_annum, _, _),
    solvibilita(Loan_id, Loan_amount, Loan_term, _),
    Income_annum > 3.5*(Loan_amount/Loan_term).

approva_per_garanzie(Loan_id) :-
    garanzie(Loan_id, Residential_assets_value, Commercial_assets_value,
    Luxury_assets_value, Bank_asset_value),
    solvibilita(Loan_id, Loan_amount, _, _),
    Residential_assets_value + Commercial_assets_value + Luxury_assets_value +
    Bank_asset_value > 2*(Loan_amount).

approva_per_gestione_finanze(Loan_id) :-
    attivo(Loan_id, _, Commercial_assets_value, Bank_asset_value),
    passivo(Loan_id, _, _, Luxury_assets_value),
    Commercial_assets_value + Bank_asset_value > Luxury_assets_value.

approva_per_piccoli_prestiti(Loan_id) :-
    richiesta(Loan_id, _, _, _, _, Loan_amount, _, _, Residential_assets_value, _, _, _, _),
    Loan_amount < 3000000,
    Residential_assets_value > Loan_amount.

approva_per_solvibilita(Loan_id) :-
    richiesta(Loan_id, _, _, _, _, Cibil_score, _, _, _, _, _),
    Cibil_score > 700.

approva_per_osservazioni(Loan_id) :-
    richiesta(Loan_id, _, _, _, Income_annum, Loan_amount, _, _, Residential_assets_value, _, _,
    Luxury_assets_value, Bank_asset_value, _),
    (Income_annum > 50000000;
    Loan_amount < 15000000;
    Residential_assets_value > 70000000;
    Luxury_assets_value > 15000000;
    Bank_asset_value > 60000000).

```

Si è quindi proceduto con il popolamento della KB tramite l'utilizzo di uno script creato appositamente per questo progetto, è possibile visualizzarlo tra i file presenti su github al nominativo di *"Creazione e popolamento dei file prolog.ipynb"*. Si procede quindi con il caricamento dei file con estensione .pl e la successiva fase

Creazione ed uso delle query per l'applicazione delle regole

Si crea quindi una query per ogni regola della kb e la si sottopone a quest'ultima tramite il processo d'interrogazione per la predizione dei prestiti approvati. I risultati sono stati i seguenti:

- Prestiti approvati: 3724
- Prestiti approvati per osservazioni: 3999

Dove “prestiti approvati per osservazioni” sono appunto quelli approvati tramite l’applicazione della regola creata sulla base delle osservazioni effettuate nella fase di apprendimento.

Una prima osservazione che è possibile fare è che in entrambi i casi, i prestiti approvati sono circa il 50% in più di quelli che realmente sono approvati nel dataset. Questo potrebbe indicare **score di recall più alti a danno di una precision più modesta**.

Calcolo delle predizioni dei prestiti approvati e delle relative performance

Le performance ottenute sulla base dei prestiti appena approvati sono le seguenti:

```
% ----- KB regole classiche ----- %

Recall: 0.8840361445783133
Precision: 0.6305048335123523
Accuracy: 0.6055282267509956
F1-score: 0.7360501567398119

% ----- KB regole osservazioni ----- %

Recall: 0.9352409638554217
Precision: 0.6211552888222055
Accuracy: 0.6048254860623097
F1-score: 0.7465063861758076

%
```

Discussione risultati e conclusione

I risultati ottenuti, come ipotizzato, presentano score molto alti per la recall e più modesti per la precision. **Questo indica appunto il dato discusso precedentemente secondo cui i prestiti approvati fossero considerevolmente superiori a quelli approvati realmente.**

Sviluppi futuri potrebbero portare ad un miglioramento delle regole presenti nella base di conoscenza, per via empirica o tramite metodologia scientifica/sperimentale, così da rendere le previsioni più affidabili in termini di precision e accuracy.

Possibili sviluppi futuri

In futuro si potrebbe procedere con l’approfondimento e il test dei parametri utilizzati dalla grid search per addestrare il modello al fine di ottenere migliori performance soprattutto con i modelli KNN e MLP.

Si potrebbe inoltre ampliare la parte di apprendimento supervisionato con l’implementazione di un classificatore SVM che in questo progetto non è stato possibile aggiungere per questioni di tempo e risorse computazionali ridotte, in quanto l’addestramento di questo tramite grid search dovrebbe essere effettuato su una macchina in locale discretamente potente.

Questo progetto non ha potuto implementare parti computazionalmente impegnative in quanto si è utilizzato esclusivamente google colab per l’esecuzione del codice.

In conclusione si potrebbe espandere in lungo e in largo la base di conoscenza tramite l’utilizzo di nuove regole per ricavare per esempio liste di fatti con determinate caratteristiche o regole che ricevessero in input più attributi.