



UNIVERSITÀ POLITECNICA DELLE MARCHE

CORSO DI LAUREA MAGISTRALE DI INTELLIGENZA ARTIFICIALE

Relazione di Progetto

# **Loan Prediction Problem**

Gruppo di lavoro

Simone Di Battista (s1120038)

Gianluca Baldelli (s1121772)

Professore

Aldo Franco Dragoni

Anno Accademico 2024/2025

# Indice

1	Descrizione Problema	1
2	Dataset completo	1
3	Dataset adattato al problema	2
3.1	Script trasforma.pl	2
4	Dataset senza normalizzazione	2
4.1	Prolog	2
4.2	Entropia	5
5	Dataset normalizzato (grande scala)	5
5.1	Normalizzazione	5
5.2	Prolog	6
6	Dataset normalizzato (scala media)	8
6.1	Normalizzazione	8
6.2	Prolog	9
6.3	Script trasformaNorm2.py & Probabilità	10
7	GitHub	11

## 1 Descrizione Problema

Il *Loan Prediction Problem* è un problema che coinvolge l'approvazione o il rifiuto di una richiesta di prestito in base a specifiche caratteristiche del richiedente e del prestito stesso. Questo tipo di problema è molto rilevante nel settore finanziario, dove le banche e gli istituti di credito cercano di minimizzare i rischi di insolvenza.

Il nostro progetto adotta un approccio che sfrutta il linguaggio di programmazione *Prolog* per implementare un modello di apprendimento cognitivo. L'obiettivo è sfruttare la potenza delle regole logiche e dell'inferenza per analizzare i dati forniti, utilizzando metodi come **rules induction** e **tree induction** per apprendere le condizioni che influenzano l'approvazione dei prestiti. Questo approccio simula un processo decisionale simile a quello umano, basato su regole e conoscenza, integrando tecniche di apprendimento automatico.

## 2 Dataset completo

Il dataset utilizzato per il progetto è stato ottenuto da **Kaggle**, una piattaforma online che offre una vasta gamma di dataset utilizzati per competizioni e progetti di data science. Questo dataset è stato progettato per risolvere il *Loan Prediction Problem*.

Il dataset è strutturato in 13 colonne, ognuna delle quali rappresenta un attributo specifico legato al richiedente e alla sua richiesta di prestito.

Queste colonne contengono informazioni quali:

- **Loan\_ID**: La colonna Loan\_ID rappresenta un identificativo univoco per ogni richiesta di prestito nel dataset. Ogni prestito ha un codice distinto che lo rende facilmente riconoscibile e tracciabile.
- **Gender**: La colonna Gender rappresenta il genere del richiedente.
- **Married**: La colonna Married rappresenta lo stato civile del richiedente, ossia se il richiedente è sposato.
- **Dependents**: La colonna Dependents rappresenta il numero di persone a carico del richiedente.
- **Education**: La colonna Education rappresenta il livello di istruzione del richiedente.
- **Self\_Employed**: La colonna Self\_Employed rappresenta se il richiedente è un lavoratore autonomo o ha un lavoro dipendente.
- **ApplicantIncome**: La colonna ApplicantIncome rappresenta il reddito mensile del richiedente.
- **CoapplicantIncome**: La colonna CoapplicantIncome rappresenta il reddito mensile del coapplicante (coniuge).
- **LoanAmount**: La colonna LoanAmount rappresenta l'importo del prestito richiesto in migliaia.
- **Loan\_Amount\_Term**: La colonna Loan\_Amount\_Term rappresenta la durata del prestito in mesi.
- **Credit\_History**: La colonna Credit\_History rappresenta la storia creditizia del richiedente.
- **Property\_Area**: La colonna Property\_Area rappresenta l'area in cui si trova la proprietà.
- **Loan\_Status**: La colonna Loan\_Status rappresenta l'esito della richiesta di prestito.

Il dataset è suddiviso in **training set** e **test set**. Il training set comprende **614 record**, mentre il test set ne conta **367**. Entrambi i set sono strutturati con gli stessi attributi descritti in precedenza, che forniscono le informazioni necessarie per prevedere l'approvazione o meno del prestito.

### 3 Dataset adattato al problema

Partendo dal dataset originale, è stato deciso di non selezionare per il problema le colonne relative agli attributi Loan\_ID e Education, in quanto Loan\_ID è un identificativo univoco che non influisce sulla previsione dell'approvazione del prestito, mentre Education non è stato ritenuto un fattore determinante per il nostro modello, che si concentra principalmente su variabili economiche e finanziarie. Da ciò è stato determinato il dataset utilizzato nel problema.

In conclusione, è stato rilevato che alcuni campi risultavano vuoti (**missing values**) in diverse colonne, e tali valori mancanti sono stati opportunamente riempiti con il valore **nil** per garantire la completezza dei dati durante il processo di analisi.

#### 3.1 Script trasforma.pl

Lo script *trasforma.pl* viene utilizzato per leggere i nomi delle colonne e, per ogni riga del dataset (in formato .txt), creare un **fatto** Prolog associando i valori agli attributi corrispondenti. Ogni record del *training set* viene trasformato in un fatto del tipo **'e(Target, [Nome1=Valore1, Nome2=Valore2, ...])'**, mentre ogni record del *test set* viene trasformato nel formato **'s(Target, [Nome1=Valore1, Nome2=Valore2, ...])'**. In entrambi i casi, 'Target' rappresenta il valore della classe target ossia Loan\_Status e i nomi delle colonne sono associati ai relativi valori. I fatti vengono successivamente salvati in un file nel formato Prolog.

Partendo da un esempio di riga del dataset come:

```
male,no,0,no,5849,0,nil,360,1,urban,y
```

L'applicazione dello script *trasforma.pl* genera il seguente fatto Prolog:

```
e(y, [gender=male, married=no, dependents=0, self_employed=no,
applicantincome=5849, coapplicantincome=0, loanamount=nil,
loan_amount_term=360, credit_history=1, property_area=urban]).
```

### 4 Dataset senza normalizzazione

Inizialmente, si è deciso di avviare il progetto senza applicare alcuna normalizzazione al dataset, utilizzando direttamente i valori originali degli attributi presenti nel dataset iniziale. In particolare:

Attributi	Possibili Valori
Gender	male, female, nil
Married	yes, no, nil
Dependents	0, 1, 2, 3, nil
Self_Employed	yes, no, nil
ApplicantIncome	valori tra 0 e 81000
CoapplicantIncome	valori tra 0 e 41667
LoanAmount	valori tra 9 e 700, nil
Loan_Amount_Term	valori tra 6 e 480, nil
Credit_History	0, 1, nil
Property_Area	urban, semiurban, rural

Tabella 1: Attributi - Valori

#### 4.1 Prolog

Si è proceduto all'utilizzo del Prolog con il dataset privo di normalizzazione. Gli attributi vengono memorizzati in un file Prolog nella seguente maniera:

```
a(gender, [male, female, nil]).
a(married, [yes, no, nil]).
a(dependents, [0, 1, 2, 3, nil]).
a(self_employed, [yes, no, nil]).
% Altri attributi seguono, ma non vengono elencati per brevità.
```

Mentre gli esempi contenuti all'interno del training set e test set seguono la struttura prima citata, nella sezione *Script trasforma.pl*.

Gli script **Rules\_induction** e **Tree\_induction** sono stati utilizzati con successo per apprendere le regole di classificazione e per generare l'albero decisionale a partire dal training set. Entrambi gli approcci hanno prodotto risultati validi.

Per quanto riguarda lo script *Rules\_induction*, dall'apprendimento effettuato tramite i predicati di seguito riportati, abbiamo ottenuto i seguenti risultati:

```
?- apprendi(y).
?- apprendi(n).
```

```
?- apprendi(y).
y==
[loanamount=130]
[loanamount=144]
[loanamount=108]
[loanamount=200]
[applicantincome=2583]
[applicantincome=3333]
[applicantincome=8333]
[loanamount=90]
[loanamount=122]
[loanamount=131]
[loanamount=137]
[applicantincome=2500,loan_amount_term=360]
[married=nil]
[applicantincome=3500]
[applicantincome=6250]
[coapplicantincome=1750]
[coapplicantincome=5625]
[loanamount=107]
[loanamount=115,credit_history=1]
[loanamount=162]
[loanamount=168]
[loan_amount_term=120]
[loanamount=120,self_employed=no,credit_history=1]
[applicantincome=1025]
[applicantincome=1820]
[applicantincome=2083]
[applicantincome=2213]
[applicantincome=2383]
[applicantincome=2479]
[applicantincome=2666]
[applicantincome=2833]
[applicantincome=3083]
[applicantincome=3859]
[applicantincome=4652]
[applicantincome=9323]
[applicantincome=14583]
[coapplicantincome=1459]
[coapplicantincome=1560]
[coapplicantincome=1640]
[coapplicantincome=1833]
[coapplicantincome=2054]
[coapplicantincome=2531]
[coapplicantincome=2917]
[loanamount=30]
[loanamount=40]
[loanamount=44]
[loanamount=56]
[loanamount=97]
[loanamount=125,credit_history=1]
[loanamount=139]
[loanamount=141]
[loanamount=148]
[loanamount=154]
[loanamount=157]
[loanamount=175,property_area=semiurban]
[loanamount=184]
[loanamount=259]
[loanamount=600]
[loanamount=90,married=yes]
[loanamount=124,married=yes]
```

Figura 1: Apprendi esempi Yes

```
?- apprendi(n).
n==
[credit_history=0,property_area=urban,dependents=0]
[credit_history=0,self_employed=yes]
[credit_history=0,married=yes,dependents=2]
[credit_history=0,property_area=rural,loan_amount_term=360,married=no,coapplicantincome=0,gender=female]
[credit_history=0,loan_amount_term=nil]
[applicantincome=10000]
[coapplicantincome=1800]
[credit_history=0,gender=male,married=no,dependents=0,property_area=urban,loan_amount_term=180]
[applicantincome=2378]
[applicantincome=2947]
[applicantincome=3418]
[coapplicantincome=2451]
[coapplicantincome=2569]
[coapplicantincome=2925]
[coapplicantincome=3000]
[loanamount=281]
[loan_amount_term=36]
[credit_history=0,dependents=1,loan_amount_term=180]
[credit_history=0,dependents=nil]
[credit_history=0,loanamount=100]
[applicantincome=416]
[applicantincome=1000]
[applicantincome=1378]
[applicantincome=1442]
[applicantincome=1600]
[applicantincome=1836]
[applicantincome=1853]
[applicantincome=1880]
[applicantincome=1916]
[applicantincome=2000]
[applicantincome=2130]
[applicantincome=2149]
[applicantincome=2281]
[applicantincome=2435]
[applicantincome=2473]
[applicantincome=2518]
[applicantincome=2647]
[applicantincome=2755]
[applicantincome=2769]
[applicantincome=3036]
[applicantincome=3069]
[applicantincome=3087]
[applicantincome=3089]
[applicantincome=3125]
[applicantincome=3250]
[applicantincome=3366]
[applicantincome=3427]
[applicantincome=3430]
[applicantincome=3522]
[applicantincome=3539]
[applicantincome=3547]
[applicantincome=3583]
[applicantincome=3588]
[applicantincome=3593]
[applicantincome=3598]
[applicantincome=3600]
[applicantincome=3867]
[applicantincome=3875]
```

Figura 2: Apprendi esempi No

Le immagini, per brevità, sono tagliate. Inoltre, abbiamo calcolato la matrice di confusione per il modello, ottenendo i seguenti risultati:

```
?- stampa_matrice_di_confusione.
Test effettuati :367
Test non classificati :204
Veri Negativi 33 Falsi Positivi 14
Falsi Negativi 13 Veri Positivi 103
Accuratezza: 0.8343558282208589
Errore: 0.16564417177914115
true.
```

Figura 3: Matrice di Confusione basata su Regole

Per quanto riguarda *Tree Induction*, sono stati creati due alberi decisionali distinti, uno utilizzando il criterio di **Gini** e l'altro basato sull'**Entropia**. Questi due metodi sono stati applicati per confrontare le performance degli alberi decisionali in base alla misura di impurità scelta. Attraverso l'apprendimento effettuato tramite il predicato, di seguito riportato, chiamato nei due script 'treeInduction\_entropia' e 'treeInduction\_gini', abbiamo ottenuto i seguenti risultati:

```
?- induci_albero(X).
```

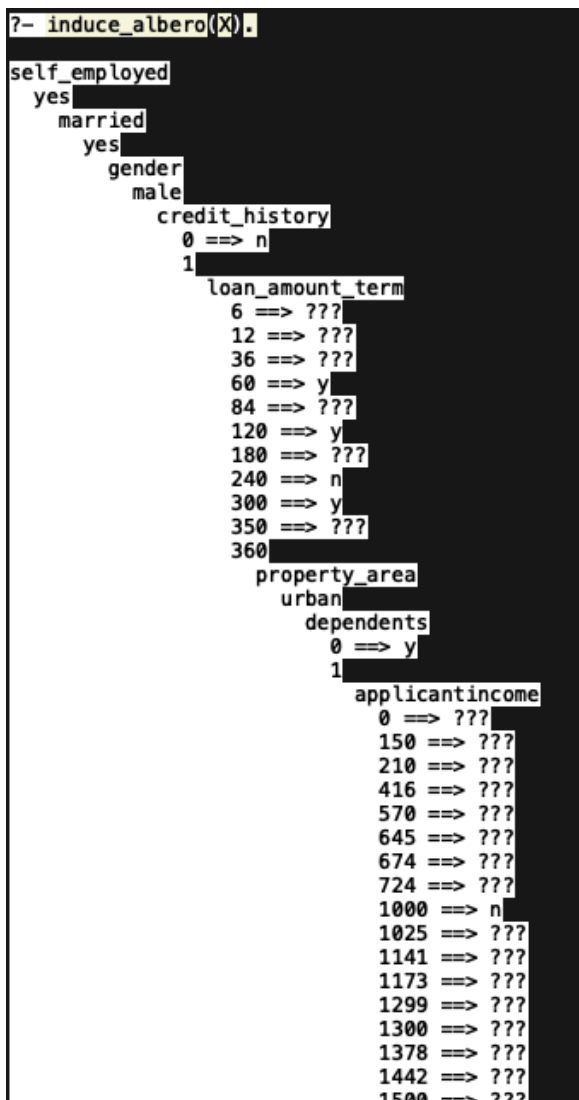


Figura 4: Albero Entropia

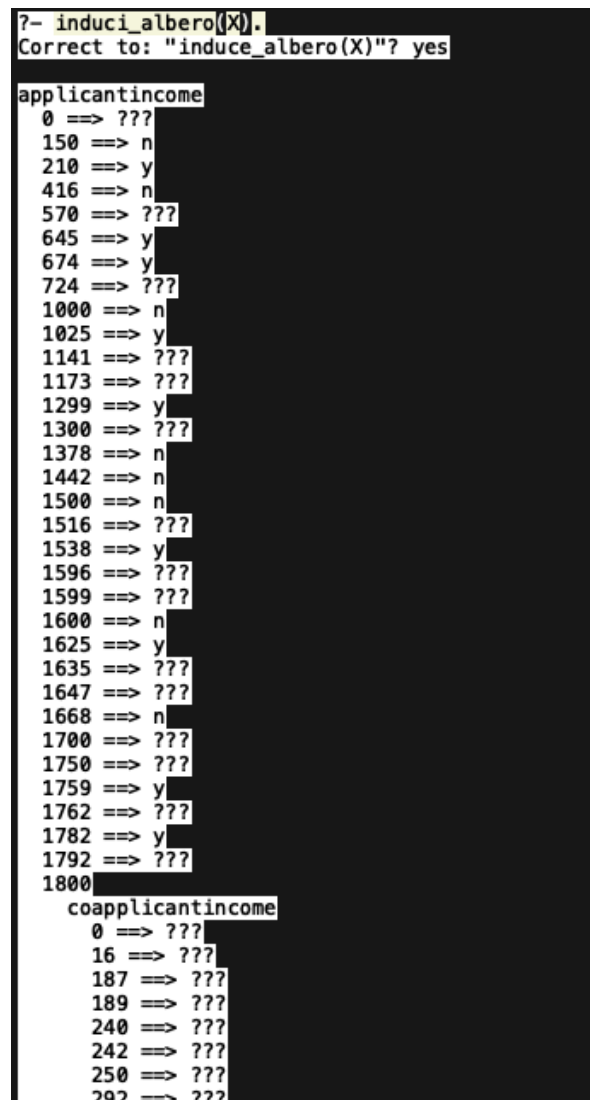


Figura 5: Albero Gini

Le immagini, per brevità, sono tagliate.

Le strutture compatte degli alberi ottenuti sono le seguenti:

- Struttura compatta entropia

```
X = t(self_employed, [yes:t(married, [yes:t(gender, [male:t(..., ...), ... : ...|...]),
no:t(loan_amount_term, [... : ...|...]), nil:null]),
no:t(gender, [male:t(married, [... : ...|...]), female:t(loan_amount_term, [...|...]),
nil:t(..., ...)], nil:t(loan_amount_term, [6:null, 12:null, ... : ...|...])]).
```

- Struttura compatta Gini

```
X = t(applicantincome, [0:null, 150:l(n), 210:l(y), 416:l(n) , 570:null, 645:l(y), 674:l(...),
... : ...|...]) .
```

In conclusione, abbiamo calcolato la matrice di confusione per il modello basato sugli alberi, ottenendo i seguenti risultati:

```
?- stampa_matrice_di_confusione.
Test effettuati :367
Test non classificati :234
Veri Negativi 33   Falsi Positivi 10
Falsi Negativi 14   Veri Positivi 76
Accuratezza: 0.8195488721804511
Errore: 0.18045112781954886
true
```

Figura 6: Matrice di Confusione basata su Alberi

## 4.2 Entropia

Per la costruzione dell'albero decisionale, oltre a Gini, è stata adottata l'entropia come criterio per la scelta degli attributi a ogni nodo. L'**entropia**, che misura il grado di incertezza o disordine in un dataset, è stata calcolata tramite una funzione implementata in Prolog seguendo la formula:

$$H(S) = - \sum_{i=1}^k p_i \cdot \log_2(p_i)$$

dove  $S$  rappresenta il dataset,  $k$  è il numero di classi, e  $p_i$  è la probabilità degli esempi appartenenti alla classe  $i$ .

Nel codice Prolog, l'entropia iniziale viene calcolata sull'intero dataset. Successivamente, per ciascun attributo, il dataset viene suddiviso in sottoinsiemi in base ai valori distinti di quell'attributo; per ciascun sottoinsieme, l'entropia viene calcolata utilizzando la stessa formula e combinata in un'entropia totale ponderata tramite il predicato:

$$H_{ponderata}(S, A) = \sum_{v \in \text{Val}(A)} \frac{|S_v|}{|S|} \cdot H(S_v)$$

dove  $A$  è l'attributo considerato,  $\text{Val}(A)$  è l'insieme dei valori distinti di  $A$ ,  $S_v$  è il sottoinsieme degli esempi con valore  $v$  per  $A$ , e  $|S|$  è la cardinalità del dataset originale.

Nel codice, questo processo è stato implementato con una funzione ricorsiva che itera sui valori distinti di un attributo, accumulando la somma ponderata delle entropie dei sottoinsiemi. Infine, il guadagno informativo (Gain), utilizzato per scegliere l'attributo migliore, è stato calcolato come differenza tra l'entropia iniziale e quella ponderata:

$$\text{Gain}(S, A) = H(S) - H_{ponderata}(S, A)$$

Il codice utilizza un predicato dedicato per il calcolo del guadagno informativo, che seleziona l'attributo con il massimo valore di Gain; tale selezione garantisce che la radice o il nodo principale dell'albero sia quello che riduce maggiormente l'incertezza complessiva del sistema. Questo approccio iterativo, implementato completamente in Prolog, ha consentito di costruire un albero interpretabile e focalizzato sulla distinzione efficace delle classi.

## 5 Dataset normalizzato (grande scala)

Nel secondo approccio, a partire dal dataset originale, è stata applicata una normalizzazione su quattro attributi principali. Questa trasformazione aveva l'obiettivo di migliorare la leggibilità degli alberi decisionali e delle regole acquisite durante l'apprendimento, rendendo i risultati più chiari e facilmente interpretabili.

### 5.1 Normalizzazione

La normalizzazione è stata effettuata su quattro attributi:

- *ApplicantIncome*

Per l'attributo ApplicantIncome, che presentava valori compresi tra 0 e 81.000, è stata effettuata una normalizzazione tramite classificazione in intervalli. I valori sono stati raggruppati nelle seguenti categorie:

- **low**: per i valori compresi tra 0 e 2.876.
- **medium\_low**: per i valori compresi tra 2.877 e 3.813.
- **medium\_high**: per i valori compresi tra 3.814 e 5.800.
- **high**: per i valori compresi tra 5.801 e 69.000.
- **very\_high**: per i valori compresi tra 69.001 e 81.000.

- *coApplicantIncome*

Per l'attributo *coApplicantIncome*, che presentava valori compresi tra 0 e 41667, è stata effettuata una normalizzazione tramite classificazione in intervalli. I valori sono stati raggruppati nelle seguenti categorie:

- **null**: per i valori uguali a 0.
- **low**: per i valori tra 1 e il 25° percentile (fino a 1210).
- **medium\_low**: per i valori tra il 25° percentile (1211) e il 50° percentile (2400).
- **medium\_high**: per i valori tra il 50° percentile (2401) e il 75° percentile (32000)
- **high**: per i valori tra il 75° percentile (32001) e il massimo (41667).

- *LoanAmount*

Per l'attributo *LoanAmount*, che presentava valori compresi tra 9 e 700, è stata effettuata una normalizzazione tramite classificazione in intervalli. I valori sono stati raggruppati nelle seguenti categorie:

- **low**: per i valori tra 9 e 50.
- **medium\_low**: per i valori tra 51 e 100.
- **medium\_high**: per i valori tra 101 e 200.
- **high**: per i valori tra 201 e 400.
- **very\_high**: per i valori tra 401 e 700.

- *Loan\_Amount\_Term*

Per l'attributo *Loan\_Amount\_Term*, che presentava valori compresi tra 6 e 480, è stata effettuata una normalizzazione tramite classificazione in intervalli. I valori sono stati raggruppati nelle seguenti categorie:

- **low\_term**: per i valori tra 9 e 60.
- **medium\_low\_term**: per i valori tra 61 e 180.
- **medium\_high\_term**: per i valori tra 181 e 361.
- **high\_term**: per i valori tra 361 e 480.

I valori **nil** sono stati mantenuti come tali.

Di seguito un'immagine che riporta un esempio del dataset normalizzato :

Gender	Married	Dependents	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
Male	No	0	No	High	Null	NIL	Medium_Long_term	1	Urban
Male	Yes	1	No	Medium_High	Medium_Low	Medium_High	Medium_Long_term	1	Rural
Male	Yes	0	Yes	Medium_Low	Null	Medium_Low	Medium_Long_term	1	Urban
Male	Yes	0	No	Low	Medium_Low	Medium_High	Medium_Long_term	1	Urban
Male	No	0	No	High	Null	Medium_High	Medium_Long_term	1	Urban
Male	Yes	2	Yes	Medium_High	Medium_High	High	Medium_Long_term	1	Urban
Male	Yes	0	No	Low	Medium_Low	Medium_Low	Medium_Long_term	1	Urban
Male	Yes	3	No	Medium_Low	Medium_High	Medium_High	Medium_Long_term	0	Semiurban
Male	Yes	2	No	Medium_High	Medium_Low	Medium_High	Medium_Long_term	1	Urban

Figura 7: Esempio dataset normalizzato (grande scala)

## 5.2 Prolog

Si è utilizzato Prolog con il dataset normalizzato, seguendo le modalità descritte in precedenza. Gli attributi sono stati memorizzati in un file Prolog con la stessa struttura del dataset non normalizzato, ma adattando opportunamente i valori degli attributi normalizzati. La rappresentazione degli esempi nel training set e nel test set è rimasta invariata rispetto all'approccio precedente. Abbiamo proceduto con lo script **Rules\_induction**.

Successivamente, abbiamo provato a effettuare l'apprendimento utilizzando i predicati indicati di seguito; tuttavia, in entrambi i casi, il risultato ottenuto è stato *false*.

```
?- apprendi(y).
?- apprendi(n).
```

```
?- apprendi(y).
false.
```

Figura 8: Apprendi esempi Yes

```
?- apprendi(n).
false.
```

Figura 9: Apprendi esempi No

Il fallimento dell'apprendimento è principalmente dovuto al processo di selezione delle condizioni basate sui punteggi assegnati alle coppie attributo-valore. Durante l'elaborazione, i punteggi calcolati non riescono a evidenziare una distinzione significativa tra le diverse classi, rendendo complessa l'identificazione di regole utili per il riconoscimento. Questa mancanza di chiarezza può derivare da una normalizzazione non ottimale o dall'assenza di correlazioni sufficientemente forti tra gli attributi e le classi. Di conseguenza, lo script non è in grado di generare regole valide per l'induzione.

Inoltre, il predicato *stampa\_matrice\_di\_confusione* non produce alcun risultato, poiché il suo funzionamento richiede un modello con regole apprese per poter calcolare i valori della matrice di confusione.

Per quanto riguarda *Tree Induction*, sono stati creati due alberi decisionali distinti, uno utilizzando il criterio di **Gini** e l'altro basato sull'**Entropia**, proprio come nel caso precedente. Attraverso l'apprendimento effettuato tramite il predicato, di seguito riportato, chiamato nei due script 'treeInduction\_entropia' e 'treeInduction\_gini', abbiamo ottenuto i seguenti risultati:

```
?- induci_albero(X).
```

```
?- induci_albero(X).
self_employed
yes
gender
male
married
yes
loan_amount_term
short_term ==> y
medium_short_term ==> n
medium_long_term
credit_history
0 ==> n
1
property_area
urban
coapplicantincome
null ==> y
low ==> ???
medium_low ==> ???
medium_high
applicantincome
low
loanamount
low ==> ???
medium_low ==> ???
medium_high
dependents
0 ==> y
1 ==> n
2 ==> ???
3 ==> ???
```

Figura 10: Albero Entropia

```
?- induce_albero(X).
credit_history
0
applicantincome
low
coapplicantincome
null ==> n
low ==> ???
medium_low ==> n
medium_high
married
yes
loan_amount_term
short_term ==> ???
medium_short_term
loanamount
low ==> ???
medium_low ==> y
medium_high ==> n
high ==> ???
very_high ==> ???
nil ==> ???
medium_long_term ==> n
long_term ==> ???
nil ==> ???
no ==> y
nil ==> ???
high ==> ???
medium_low ==> n
medium_high
coapplicantincome
null
```

Figura 11: Albero Gini

Le immagini, per brevità, sono tagliate.



Le strutture compatte degli alberi ottenuti sono le seguenti:

- Struttura compatta entropia

```
X = t(self_employed, [yes:t(gender, [male:t(married, [yes:t(..., ...), ... : ...|...]), female:t(
  coapplicantincome, [... : ...|...]), nil:null]), no:t(gender, [male:t(married, [... :
  ...|...]), female:t(loan_amount_term, [...|...]), nil:t(..., ...)]), nil:t(credit_history,
  [0:null, 1:1(...), ... : ...|...])) .
```

- Struttura compatta Gini

```
X = t(credit_history, [0:t(applicantincome, [low:t(coapplicantincome, [null:1(...), ... :
  ...|...]), medium_low:1(n), medium_high:t(coapplicantincome, [...|...]), high:t(..., ...),
  ... : ...]), 1:t(property_area, [urban:t(coapplicantincome, [... : ...|...]), semiurban:t(
  loan_amount_term, [...|...]), rural:t(..., ...)]), nil:t(self_employed, [yes:t(dependents,
  [...|...]), no:t(..., ...), ... : ...|...])) .
```

Anche in questo caso però, il predicato *stampa\_matrice\_di\_confusione* non produce alcun risultato, dando come risultato *false*.

## 6 Dataset normalizzato (scala media)

Il terzo approccio è stato necessario a causa del fallimento nell'apprendimento delle regole riscontrato nella normalizzazione precedente (grande scala), tuttavia l'obiettivo è rimasto lo stesso: migliorare la leggibilità degli alberi decisionali e delle regole acquisite durante l'apprendimento. In questo caso, è stata adottata una nuova strategia di normalizzazione su scala più ridotta, con l'intento di preservare una maggiore granularità dei dati e facilitare l'estrazione di regole significative.

### 6.1 Normalizzazione

La normalizzazione è stata effettuata su quattro attributi:

- Per gli attributi *ApplicantIncome* e *coApplicantIncome* è stata effettuata una normalizzazione tramite classificazione in intervalli numerici. I valori sono stati raggruppati nelle seguenti categorie:
  - 1: per i valori compresi tra 0 e 100.
  - 2: per i valori compresi tra 101 e 200.
  - 3: per i valori compresi tra 201 e 300.
  - ...: con lo stesso schema fino ai relativi valori massimi di ciascun attributo.
- Per gli attributi *LoanAmount* e *Loan\_Amount\_Term* è stata effettuata una normalizzazione tramite classificazione in intervalli numerici. I valori sono stati raggruppati nelle seguenti categorie:
  - 1: per i valori compresi tra 0 e 10.
  - 2: per i valori compresi tra 11 e 20.
  - 3: per i valori compresi tra 21 e 30.
  - ...: con lo stesso schema fino ai relativi valori massimi di ciascun attributo.

I valori **nil** sono stati mantenuti come tali.

Di seguito un'immagine che riporta un esempio del dataset normalizzato :

Gender	Married	Dependents	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
Male	No	0	No	59	1	nil	37	1	Urban
Male	Yes	1	No	46	16	13	37	1	Rural
Male	Yes	0	Yes	30	1	7	37	1	Urban
Male	Yes	0	No	26	24	13	37	1	Urban
Male	No	0	No	60	1	15	37	1	Urban
Male	Yes	2	Yes	55	42	27	37	1	Urban
Male	Yes	0	No	24	16	10	37	1	Urban
Male	Yes	3	No	31	26	16	37	0	Semiurban
Male	Yes	2	No	41	16	17	37	1	Urban

Figura 12: Esempio dataset normalizzato (media scala)

## 6.2 Prolog

Si è utilizzato Prolog con il dataset normalizzato, seguendo le modalità descritte in precedenza. Gli attributi sono stati memorizzati in un file Prolog con la stessa struttura del dataset normalizzato (grande scala), ma adattando opportunamente i valori degli attributi normalizzati. La rappresentazione degli esempi nel training set e nel test set è rimasta invariata rispetto all'approccio precedente. Abbiamo eseguito il processo con lo script **Rules\_induction**, utilizzando i predicati specificati di seguito. In questo caso, a differenza della fase di normalizzazione precedente, abbiamo ottenuto un risultato valido e non un risultato *false*.

```
?- apprendi(y).
?- apprendi(n).
```

```
?- apprendi(y).
y<=
[applicantincome=37]
[coapplicantincome=11]
[loanamount=6,self_employed=no]
[coapplicantincome=24,credit_history=1]
[coapplicantincome=28]
[coapplicantincome=57]
[applicantincome=57]
[applicantincome=94]
[applicantincome=96]
[coapplicantincome=12]
[applicantincome=39,dependents=0]
[applicantincome=21,dependents=0]
[applicantincome=33,property_area=urban]
[married=nil]
[applicantincome=45]
[applicantincome=72]
[applicantincome=167]
[coapplicantincome=7]
[loan_amount_term=4]
[loan_amount_term=19,self_employed=yes]
[applicantincome=7]
[applicantincome=49,married=no]
[applicantincome=54]
[applicantincome=69]
[applicantincome=71]
[applicantincome=78]
[applicantincome=81]
[applicantincome=146]
[applicantincome=149]
[coapplicantincome=5]
[coapplicantincome=29,married=yes]
[coapplicantincome=72]
[loanamount=41]
[loanamount=61]
[coapplicantincome=21,loanamount=13]
[applicantincome=29,credit_history=1]
[applicantincome=60,married=no]
[applicantincome=25,loanamount=11]
[applicantincome=59,married=yes]
[applicantincome=84,married=no]
[coapplicantincome=10,self_employed=no]
[coapplicantincome=31,dependents=0]
[coapplicantincome=45,married=yes]
[loanamount=5,credit_history=1,gender=male]
[coapplicantincome=25,property_area=semiurban]
[credit_history=1,coapplicantincome=23]
```

Figura 13: Apprendi esempi Yes

```
?- apprendi(n).
n<=
[credit_history=0,property_area=urban,dependents=0]
[credit_history=0,self_employed=yes]
[credit_history=0,loanamount=12]
[credit_history=0,loanamount=13]
[credit_history=0,loanamount=16]
[credit_history=0,coapplicantincome=1,gender=female]
[credit_history=0,married=yes,dependents=2]
[applicantincome=15]
[credit_history=0,property_area=rural,married=yes,coapplicantincome=1]
[credit_history=0,dependents=nil]
[credit_history=0,applicantincome=36]
[applicantincome=2]
[applicantincome=5]
[applicantincome=10]
[applicantincome=14]
[applicantincome=100,gender=female]
[applicantincome=110]
[applicantincome=113]
[applicantincome=125]
[applicantincome=147]
[applicantincome=184]
[applicantincome=198]
[applicantincome=283]
[applicantincome=207]
[applicantincome=339]
[coapplicantincome=54]
[coapplicantincome=78]
[coapplicantincome=110]
[coapplicantincome=113]
[coapplicantincome=339]
[loanamount=1]
[loan_amount_term=2]
[credit_history=0,loanamount=15]
[applicantincome=36,dependents=0]
[credit_history=0,loanamount=19]
[applicantincome=61,dependents=0]
[credit_history=0,loanamount=21]
[loanamount=7,dependents=1]
[applicantincome=16,dependents=2]
[applicantincome=51,married=yes]
[applicantincome=62,dependents=2]
[applicantincome=65,property_area=semiurban]
[applicantincome=68,self_employed=nil]
[applicantincome=74,married=no]
[applicantincome=75,dependents=0]
[applicantincome=88,married=no]
```

Figura 14: Apprendi esempi No

Le immagini, per brevità, sono tagliate.

Inoltre, abbiamo calcolato la matrice di confusione per il modello, ottenendo i seguenti risultati:

```
?- stampa_matrice_di_confusione.
Test effettuati :367
Test non classificati :173
Veri Negativi 11 Falsi Positivi 29
Falsi Negativi 27 Veri Positivi 127
Accuratezza: 0.711340206185567
Errore: 0.28865979381443296
true.
```

Figura 15: Matrice di Confusione basata su Regole

Le strutture compatte degli alberi ottenuti sono le seguenti:

- Struttura compatta entropia

```
X = t(gender, [male:t(self_employed, [yes:t(married, [yes:t(..., ...), ... : ...|...]), no:t(
  loan_amount_term, [... : ...|...]), nil:t(married, [...|...])]), female:t(married, [yes:t(
  self_employed, [... : ...|...]), no:t(self_employed, [...|...]), nil:l(...)], nil:t(
  loan_amount_term, [1:null, 2:null, ... : ...|...])]) .
```

- Struttura compatta Gini

```
X = t(credit_history, [0:t(applicantincome, [1:null, 2:null, 3:null, 5:null, ... : ...|...]), 1:t(
  applicantincome, [1:null, 2:1(n), 3:1(...), ... : ...|...]), nil:t(applicantincome, [1:null
  , 2:null, ... : ...|...])]) .
```

### 6.3 Script *trasformaNorm2.py* & Probabilità

In conclusione, con il terzo approccio abbiamo deciso di calcolare la **probabilità** di approvazione di un prestito in modo complessivo e condizionato a specifici attributi del richiedente e del prestito richiesto. Questo ci consente di analizzare l'influenza esercitata da ciascuna variabile considerata.

Prima di procedere con il calcolo delle probabilità, gli esempi presenti nel training set e nel test set sono stati ristrutturati. Inizialmente, ogni esempio era rappresentato in una forma strutturata come segue:

```
e(y, [gender=male, married=no, dependents=0, self_employed=no,
applicantincome=59, coapplicantincome=1, loanamount=nil,
loan_amount_term=37, credit_history=1, property_area=urban]).
```

Questa struttura, che utilizza una lista di coppie chiave-valore per rappresentare i vari attributi, è stata trasformata in una forma più compatta e diretta, del tipo:

```
e(y,male,no,0,no,59,1,nil,37,1,urban).
```

Questa trasformazione è stata effettuata attraverso l'utilizzo dello script *trasformaNorm2.py*.

Una volta completata la trasformazione, tutti gli esempi appartenenti a training e test set sono stati raccolti in un unico file, rinominato *database.pl*.

A questo punto, utilizzando lo script *probabilita.pl*, che prende come input gli esempi contenuti in *database.pl*, abbiamo determinato diverse probabilità di approvazione del prestito:

- **Probabilità complessiva di approvazione del prestito:** Abbiamo calcolato la probabilità che un prestito venga approvato indipendentemente dagli attributi dei richiedenti o del prestito richiesto. Questo calcolo fornisce una misura generale della percentuale di prestiti approvati rispetto al totale dei prestiti richiesti.

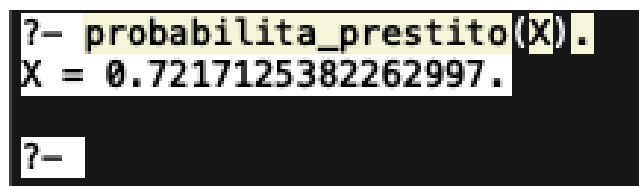


Figura 16: Probabilità approvazione prestito per un richiedente

- **Probabilità condizionata rispetto al genere del richiedente:** Abbiamo determinato la probabilità di approvazione del prestito considerando solo i richiedenti di un determinato genere (uomo o donna).

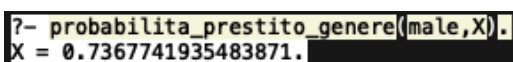


Figura 17: Probabilità approvazione prestito per un richiedente uomo

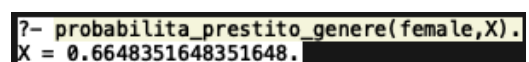


Figura 18: Probabilità approvazione prestito per una richiedente donna

- **Probabilità condizionata rispetto allo stato civile:** Questo calcolo ci ha permesso di analizzare l'effetto dello stato civile del richiedente (sposato o non sposato) sulla probabilità di approvazione.

```
?- probabilita_prestito_stato_civile(yes,X).
X = 0.7496038034865293.
```

Figura 19: Probabilità approvazione prestito per un richiedente sposato

```
?- probabilita_prestito_stato_civile(no,X).
X = 0.6685878962536023.
```

Figura 20: Probabilità approvazione prestito per un richiedente non sposato

- **Probabilità condizionata rispetto al numero di dipendenti:** Qui abbiamo calcolato come varia la probabilità di approvazione in base al numero di persone a carico del richiedente.

```
?- probabilita_prestito_dipendenti(3,X).
X = 0.6923076923076923.
```

Figura 21: Probabilità approvazione prestito per un richiedente con tre dipendenti a carico

```
?- probabilita_prestito_dipendenti(2,X).
X = 0.76875.
```

Figura 22: Probabilità approvazione prestito per un richiedente con due dipendenti a carico

- **Probabilità condizionata rispetto all'occupazione:** Abbiamo analizzato la probabilità di approvazione distinguendo tra richiedenti autonomi e non autonomi.

```
?- probabilita_prestito_occupazione(yes,X).
X = 0.7226890756302521.
```

Figura 23: Probabilità approvazione prestito per un richiedente autonomo

```
?- probabilita_prestito_occupazione(no,X).
X = 0.724907063197026.
```

Figura 24: Probabilità approvazione prestito per un richiedente non autonomo

- **Probabilità condizionata rispetto al reddito del richiedente:** Questo calcolo ci ha permesso di osservare come il livello di reddito del richiedente influenzi la probabilità di approvazione.

```
?- probabilita_prestito_applicantincome(15,X).
X = 0.
```

Figura 25: Probabilità approvazione prestito per un richiedente con un reddito ridotto

```
?- probabilita_prestito_applicantincome(518,X).
X = 1.
```

Figura 26: Probabilità approvazione prestito per un richiedente con un reddito elevato

- **Probabilità condizionata rispetto al reddito del co-richiedente:** Similmente, abbiamo determinato l'effetto del reddito del co-richiedente sulla probabilità di approvazione.
- **Probabilità condizionata rispetto all'importo del prestito richiesto:** Questo calcolo mostra come l'importo del prestito influenzi la probabilità di essere approvato.

```
?- probabilita_prestito_loanamount(46,X).
X = 1.
```

Figura 27: Probabilità approvazione prestito di importo pari a 46.000€

- **Probabilità condizionata rispetto alla durata del prestito:** Infine, abbiamo analizzato l'impatto della durata del prestito richiesto sulla probabilità di approvazione.

## 7 GitHub

È possibile consultare i dati utilizzati e accedere alla repository del progetto al seguente link: Repository del Progetto .