

UNIVERSITÀ POLITECNICA DELLE MARCHE



---

# Progetto Intelligenza Artificiale

*Analisi e classificazione della salute mentale dei lavoratori con alberi decisionali*

---

## AUTORI

Davide Ticchiarelli - S1121687

Giampaolo Marino - S1121678

Riccardo Angelini - S1122137

**Anno accademico 2024/2025**

# Indice

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introduzione</b>                                      | <b>1</b>  |
| 1.1      | Scopo del progetto . . . . .                             | 1         |
| 1.2      | Prolog . . . . .   | 1         |
| 1.3      | Apprendimento automatico . . . . .                       | 2         |
| 1.4      | Induzione di alberi decisionali . . . . .                | 3         |
| 1.4.1    | Vantaggi . . . . .                                       | 4         |
| 1.4.2    | Limiti . . . . .   | 4         |
| 1.5      | Criterio di Gini . . . . .                               | 5         |
| 1.6      | Criterio di Entropia . . . . .                           | 5         |
| <b>2</b> | <b>Implementazione in Prolog</b>                         | <b>7</b>  |
| 2.1      | Dataset . . . . .  | 7         |
| 2.2      | Pulizia dei dati . . . . .                               | 9         |
| 2.3      | Traduzione da CSV a Prolog . . . . .                     | 11        |
| 2.4      | Creazione dei dataset di Test e Training . . . . .       | 11        |
| 2.5      | Induzione dell'Albero Decisionale con Entropia . . . . . | 13        |
| <b>3</b> | <b>Conclusioni</b>                                       | <b>18</b> |
| 3.1      | Risultati con Entropia . . . . .                         | 18        |
| 3.2      | Risultati con Gini . . . . .                             | 18        |
| 3.3      | Confronto dei risultati . . . . .                        | 19        |
| 3.4      | Esempio di albero decisionale . . . . .                  | 19        |
| 3.5      | Osservazioni finali . . . . .                            | 20        |
| 3.6      | Possibili sviluppi futuri . . . . .                      | 21        |

---

# 1 Introduzione

In questa sezione viene fornita una breve introduzione agli aspetti chiave del progetto. Si inizia con la definizione dello scopo del progetto e si prosegue con una spiegazione teorica dei concetti che verranno trattati durante l'esecuzione.

## 1.1 Scopo del progetto

Lo scopo di questo progetto è sviluppare un sistema di apprendimento automatico basato su *alberi decisionali* in *Prolog*, per la classificazione di malattie mentali nel contesto lavorativo. Utilizzando un dataset contenente informazioni su lavoratori e le loro condizioni di salute mentale, il sistema sarà in grado di apprendere dalle caratteristiche dei dati e di classificare i soggetti in base alla probabilità di essere affetti da specifiche patologie mentali, come depressione, ansia o stress. Il progetto si propone di affrontare le seguenti sfide:

1. *Preprocessing dei dati*: Il primo passo consiste nell'importare e preparare i dati, che possono includere variabili come età, genere, ambiente di lavoro, ore di lavoro, segnali di stress e altri indicatori rilevanti. I dati potrebbero essere parzialmente mancanti o rumorosi, quindi il progetto include anche attività di pulizia e normalizzazione dei dati.
2. *Creazione di alberi decisionali*: L'algoritmo utilizzerà tecniche di induzione, per costruire alberi decisionali che possano fare previsioni basate su variabili quali sesso, età, anni di esperienza, ruolo, settore e simili. Gli alberi decisionali sono scelti per la loro capacità di offrire decisioni facilmente interpretabili, cruciali in ambito medico e lavorativo. Inoltre, per la costruzione degli alberi, inizialmente verrà utilizzato il criterio di Gini, per poi confrontarlo con l'entropia.
3. *Validazione e testing del modello*: Una volta costruito l'albero decisionale, il modello verrà validato per valutarne la precisione e l'affidabilità. Inoltre, si testerà la capacità del sistema di generalizzare su nuovi dati non visti in fase di addestramento, per garantire che le sue previsioni siano robuste. Il confronto tra il criterio di Gini ed entropia sarà parte integrante di questa fase, al fine di determinare quale approccio risulti più efficace per la classificazione delle malattie mentali nel contesto lavorativo.

## 1.2 Prolog

Prolog (PROgramming in LOGic) è un linguaggio di programmazione dichiarativo basato sulla logica del primo ordine. Utilizzato principalmente in ambiti di intelligenza artificiale, Prolog permette di definire conoscenze sotto forma di fatti e regole, e di eseguire inferenze tramite un motore di unificazione e risoluzione. Il suo paradigma si basa sulla programmazione logica, rendendolo particolarmente adatto a problemi di ragionamento automatizzato, sistemi esperti e apprendimento automatico.

## 1.3 Apprendimento automatico

L'apprendimento automatico, o *Machine Learning*, è una branca dell'intelligenza artificiale che si occupa dello sviluppo di algoritmi in grado di apprendere dai dati e migliorare le proprie prestazioni senza essere esplicitamente programmati. Esistono diversi tipi di apprendimento automatico, tra cui l'apprendimento supervisionato, non supervisionato e per rinforzo.

I principali approcci di apprendimento automatico sono i seguenti:

- *Apprendimento supervisionato (Supervised Learning)*: In questo tipo di apprendimento, l'algoritmo viene addestrato su un insieme di dati etichettati, cioè dati in cui ogni esempio è associato a una risposta corretta (etichetta). L'obiettivo è imparare una funzione che, dato un input, possa prevedere correttamente l'output associato. È particolarmente utile per problemi di classificazione e regressione.
- *Apprendimento non supervisionato (Unsupervised Learning)*: In questo caso, l'algoritmo non riceve etichette o risposte corrette. L'obiettivo è scoprire strutture o pattern nascosti nei dati, come raggruppamenti o correlazioni. Alcuni degli approcci più comuni includono il clustering (come l'algoritmo K-means) e l'analisi delle componenti principali (PCA).
- *Apprendimento per rinforzo (Reinforcement Learning)*: Questo tipo di apprendimento si basa su un agente che interagisce con un ambiente e impara a prendere decisioni ottimali tramite un processo di trial and error. L'agente riceve un feedback sotto forma di ricompensa o penalità in base alle sue azioni, e l'obiettivo è massimizzare la somma delle ricompense nel lungo periodo. L'apprendimento per rinforzo è particolarmente utilizzato in applicazioni come i giochi, la robotica e la pianificazione automatica.

Le principali differenze tra i tre approcci sono le seguenti:

- *Supervised learning*:
  - L'algoritmo impara da un insieme di dati etichettati, con esempi già corretti.
  - L'obiettivo è prevedere un output a partire da un input, generalmente in contesti di classificazione o regressione.
- *Unsupervised learning*:
  - L'algoritmo lavora su dati senza etichette, cercando di identificare strutture o gruppi nascosti.
  - Non esiste un "output corretto" da predire, ma piuttosto si cerca di analizzare i dati in modo da trovarne delle caratteristiche significative.
- *Reinforcement learning*:

- L'algoritmo interagisce con l'ambiente e apprende attraverso feedback di tipo ricompensa o penalità.
- L'obiettivo è massimizzare la ricompensa totale nel tempo, migliorando le decisioni in base alle esperienze passate.

Nel contesto di Prolog, l'apprendimento automatico può essere implementato attraverso tecniche di induzione di regole (Inductive Logic Programming, o ILP), alberi decisionali e altri modelli basati sulla logica. In particolare, le tecniche di ILP consentono di generare regole logiche a partire da esempi concreti. Prolog può essere utilizzato per rappresentare le conoscenze acquisite e inferire nuove regole attraverso un processo di generalizzazione.

## 1.4 Induzione di alberi decisionali

Un *albero decisionale* è un modello di apprendimento automatico utilizzato per la classificazione e la regressione, che rappresenta le decisioni e le possibili conseguenze di una serie di azioni. L'albero decisionale è strutturato come un albero in cui:

- I *nodi interni* rappresentano le *domande* o *condizioni* sui dati.
- Le *foglie* rappresentano il risultato finale, che può essere la *classe* (nel caso della classificazione) o un *valore continuo* (nel caso della regressione).

In altre parole, l'albero fa una serie di domande sui dati per decidere la classe o il valore da attribuire a una nuova osservazione.

L'*induzione* di un albero decisionale è il processo di costruzione di un albero a partire da un dataset di addestramento. Durante questo processo, si selezionano le migliori caratteristiche (attributi) da utilizzare per suddividere il dataset in modo da ottenere sottoinsiemi omogenei. Il processo di induzione di un albero decisionale si svolge generalmente nei seguenti passi:

1. *Selezione della caratteristica migliore*: In ogni nodo dell'albero, è necessario scegliere la caratteristica (attributo) che meglio suddivide i dati. L'obiettivo è ridurre l'incertezza riguardo alla classe o al valore che si vuole prevedere. Questa selezione viene fatta utilizzando metriche come:
  - *Criterio di Gini*, che misura la "purezza" dei nodi.
  - *Criterio di Entropia*, che misura l'incertezza nei dati.
2. *Suddivisione dei Dati*: una volta selezionato l'attributo, i dati vengono suddivisi in sottoinsiemi in base ai valori di tale attributo. Ogni sottoinsieme corrisponde a un ramo dell'albero. Questo processo viene ripetuto per ogni sottoinsieme, costruendo progressivamente l'albero decisionale.

3. *Condizione di arresto*: Il processo di induzione continua finché uno dei seguenti criteri non viene soddisfatto:

- Il nodo contiene solo esempi appartenenti alla stessa classe.
- Non ci sono più attributi da utilizzare per la suddivisione.
- Il numero di esempi in un nodo è inferiore a una soglia predefinita.

Quando una di queste condizioni è soddisfatta, il nodo diventa una foglia dell'albero, con la classe o il valore finale assegnato in base ai dati contenuti nel nodo.

#### 1.4.1 Vantaggi

Gli alberi decisionali offrono numerosi vantaggi, tra cui:

- *Facilità di interpretazione*: gli alberi decisionali sono facilmente interpretabili, poiché le decisioni possono essere seguite in modo trasparente, consentendo una comprensione chiara di come vengono prese le decisioni.
- *Nessuna necessità di normalizzazione dei dati*: gli alberi decisionali non richiedono che i dati siano normalizzati o standardizzati, a differenza di altri algoritmi di machine learning come le reti neurali.
- *Capacità di gestire dati misti*: possono gestire facilmente dati sia numerici che categorici, rendendoli molto flessibili.
- *Velocità di apprendimento*: gli algoritmi di induzione degli alberi decisionali sono generalmente veloci nell'apprendimento, poiché richiedono solo pochi passaggi per suddividere i dati.
- *Robustezza ai dati mancanti*: gli alberi decisionali sono in grado di gestire in modo efficiente i dati mancanti, effettuando decisioni anche con informazioni incomplete.

#### 1.4.2 Limiti

Nonostante i vantaggi, gli alberi decisionali presentano anche alcune limitazioni:

- *Overfitting*: gli alberi decisionali possono essere suscettibili all'*Overfitting*, soprattutto quando sono molto profondi. In questi casi, l'albero si adatta troppo ai dati di addestramento, perdendo la capacità di generalizzare bene su nuovi dati.
- *Instabilità*: piccole variazioni nei dati di addestramento possono portare a grandi modifiche nella struttura dell'albero, rendendo il modello instabile.

- *Alberi complessi e lunghi*: gli alberi molto profondi possono diventare difficili da interpretare e gestire. Inoltre, alberi molto complessi possono risultare in modelli poco efficienti.
- *Difficoltà nella gestione di interazioni complesse*: gli alberi decisionali non sono particolarmente bravi nel gestire interazioni complesse tra variabili. In presenza di relazioni non lineari molto complesse, altri modelli come le reti neurali o i modelli lineari possono essere più efficaci.
- *Bias verso attributi con più livelli*: gli alberi decisionali possono avere un bias verso attributi con più livelli, poiché la suddivisione dei dati tende a favorire caratteristiche con più possibili valori.

## 1.5 Criterio di Gini

Il criterio di Gini è una misura dell'impurità di un insieme di dati e viene comunemente utilizzato negli alberi decisionali per valutare la qualità di una suddivisione dei dati. L'indice di Gini è definito come:

$$Gini = 1 - \sum_{i=1}^n P_i^2$$

dove  $P_i$  rappresenta la probabilità dell'elemento di appartenere alla classe  $i$ .

L'indice di Gini varia tra 0 e 1:

- *Gini basso (pari a 0)*: indica purezza massima, ovvero tutti gli elementi appartengono alla stessa classe.
- *Gini alto (vicino a 1)*: indica massima impurità, ovvero le classi sono equidistribuite.
- *Gini intermedio*: indica una distribuzione delle classi parzialmente uniforme.

Nei processi di costruzione degli alberi decisionali, il criterio di Gini viene utilizzato per scegliere l'attributo che meglio suddivide i dati. L'algoritmo seleziona la suddivisione che porta alla minore impurità media ponderata tra i sottoinsiemi generati; esso cerca di minimizzare questa quantità per ottenere una suddivisione più pura. L'indice di Gini è ampiamente utilizzato perché ha un costo computazionale inferiore rispetto ad altre metriche come l'entropia, rendendolo particolarmente utile per grandi dataset.

## 1.6 Criterio di Entropia

L'entropia è una misura formale del disordine o dell'incertezza associata a una distribuzione di probabilità. In particolare, nella teoria dell'informazione, l'entropia di un sistema discreto con  $n$  stati possibili è definita come:

---



$$H(X) = - \sum_{i=1}^n P_i \log_2 P_i$$

dove  $P_i$  rappresenta la probabilità dell'evento  $i$ -esimo.

L'entropia quantifica la quantità media di informazione necessaria per descrivere il sistema:

- *Entropia alta*: si verifica quando la distribuzione di probabilità è più uniforme, ovvero quando gli eventi hanno probabilità simili. In questo caso, l'incertezza è massima, poiché non è possibile fare previsioni affidabili sugli stati del sistema.
- *Entropia bassa*: si verifica quando la distribuzione è squilibrata, ovvero alcuni eventi hanno probabilità molto maggiori di altri. In tal caso, il sistema è più prevedibile, e la quantità di informazione necessaria per descriverlo è minore.

L'entropia è, quindi, uno strumento fondamentale in vari campi come la teoria dell'informazione e viene utilizzata per valutare l'ordine o il grado di impurità di un insieme di dati.



---

## 2 Implementazione in Prolog

Questa sezione descrive l'intero processo di analisi dei dati relativi all'impatto del lavoro sulla salute mentale dei dipendenti, utilizzando Prolog per l'elaborazione e la costruzione degli alberi decisionali.

### 2.1 Dataset

Il dataset considerato contiene dati sull'impatto del lavoro sul benessere mentale dei dipendenti. In particolare, esplora come il lavoro e altri aspetti ad esso correlati influenzino le condizioni di salute mentale dei lavoratori. Il dataset è disponibile al seguente link: <https://www.kaggle.com/datasets/iramshahzadi9/remote-work-and-mental-health>.

Tale dataset è composto da un singolo file CSV:

- *Impact\_of\_Remote\_Work\_on\_Mental\_Health.csv*: contiene informazioni dettagliate che permettono di analizzare la salute mentale dei lavoratori.

Tra i motivi che ci hanno spinto a utilizzare questo dataset vi sono:

- *Numero di righe*: il dataset selezionato doveva contenere almeno 1.000 righe di dati per garantire un'analisi significativa, mentre quello scelto ne presenta 5.000. Questo assicura una base solida su cui effettuare le analisi, riducendo il rischio di risultati distorti da una quantità insufficiente di dati.
- *Omogeneità dei dati*: il dataset presenta un livello di omogeneità del 50%, rispettando il criterio stabilito. Nello specifico, le condizioni mentali analizzate sono quattro e sono distribuite equamente al 25%.
- *Numero di colonne*: un fattore molto importante poiché un maggior numero di colonne consente di essere più selettivi su quali colonne tenere e quali rimuovere. In questo caso, il dataset contiene 20 colonne.
- *Usabilità*: una valutazione calcolata direttamente da Kaggle su parametri come completezza, credibilità e compatibilità. In questo caso, l'usabilità è pari a 7,06.

Di seguito, nella Tabella 1, sono riportate le colonne che compongono il dataset e una loro breve descrizione.

| Attributo                         | Descrizione                                   |
|-----------------------------------|---|
| Employee_ID                       | Identificativo univoco del dipendente         |
| Age                               | Età del dipendente                            |
| Gender                            | Sesso del dipendente                          |
| Job_Role                          | Ruolo lavorativo                              |
| Industry                          | Settore industriale                           |
| Years_of_Experience               | Numero di anni di esperienza lavorativa       |
| Work_Location                     | Tipo di ambiente di lavoro                    |
| Hours_Worked_Per_Week             | Numero di ore lavorate settimanalmente        |
| Number_of_Virtual_Meetings        | Numero di riunioni virtuali                   |
| Work_Life_Balance_Rating          | Valutazione dell'equilibrio tra vita e lavoro |
| Stress_Level                      | Livello di stress percepito                   |
| Mental_Health_Condition           | Condizione di salute mentale                  |
| Access_to_Mental_Health_Resources | Accesso a risorse per la salute mentale       |
| Productivity_Change               | Variazione della produttività                 |
| Social_Isolation_Rating           | Livello di isolamento sociale                 |
| Satisfaction_with_Remote_Work     | Grado di soddisfazione con il lavoro remoto   |
| Company_Support_for_Remote_Work   | Supporto dell'azienda per il lavoro remoto    |
| Physical_Activity                 | Livello di attività fisica svolta             |
| Sleep_Quality                     | Qualità del sonno                             |
| Region                            | Regione geografica                            |

Tabella 1: Descrizione degli attributi del dataset

## 2.2 Pulizia dei dati

Una volta individuato il dataset, è stato necessario eseguire un processo di pulizia in modo da garantire che non ci fossero incongruenze o valori mancanti, assicurando così risultati più precisi. Le operazioni eseguite sono le seguenti:

- *Rimozione di colonne non necessarie*: Alcune colonne, come *Employee\_ID*, *Productivity\_Change* e *Company\_Support\_for\_Remote\_Work*, sono state eliminate poiché considerate non utili ai fini dell'analisi.
- *Ridenominazione delle colonne*: Alcune colonne sono state rinominate per renderle più comprensibili. Ad esempio, *Social\_Isolation\_Rating* è stata rinominata in *Social\_Isolation* e *Work\_Life\_Balance\_Rating* in *Work\_Life\_Balance*.
- *Categorizzazione della variabile "Hours\_Worked\_Per\_Week"*: I dati relativi alle ore lavorate settimanalmente sono stati suddivisi in categorie come *Part-time*, *Full-Time*, *Overtime* e *Unknown*, utilizzando la funzione *pd.cut* di pandas.
- *Categorizzazione dell'età*: La colonna *Age* è stata suddivisa in categorie di età, come *Youth*, *Adult*, *Senior* e *Unknown*.
- *Sostituzione dei valori "Prefer not to say"*: Nella colonna *Gender*, i valori *Prefer not to say* sono stati sostituiti con *Unknown*.
- *Categorizzazione degli anni di esperienza*: La colonna *Years\_of\_Experience* è stata suddivisa in categorie, come *Junior*, *Mid*, *Expert*, *Senior* e *Unknown*, utilizzando *pd.cut*.
- *Mappatura dei valori numerici*: I valori numerici nelle colonne *Work\_Life\_Balance* e *Social\_Isolation* sono stati mappati su etichette descrittive, come *Minimal*, *Low*, *Moderate*, *High*, *Extreme*.
- *Gestione dei valori mancanti*: per tutte le colonne che presentavano valori mancanti, tali valori sono stati riempiti con *Unknown*.

Di seguito viene mostrato il codice utilizzato per effettuare queste operazioni di pulizia (vedi *Figura 1*):

```

1  import pandas as pd
2
3  # Load the CSV file from the specified path
4  csv = pd.read_csv(
5      r"C:\Users\david\OneDrive\Documenti\GitHub\mental-health-IA\database.csv")
6
7  # Drop the specified columns from the dataframe
8  csv = csv.drop(
9      columns=['Employee_ID',
10              'Number_of_Virtual_Meetings',
11              'Access_to_Mental_Health_Resources',
12              'Productivity_Change',
13              'Company_Support_for_Remote_Work'])
14
15  # Rename specific columns to more user-friendly names
16  csv = csv.rename(columns={
17      'Social_Isolation_Rating': 'Social_Isolation',
18      'Work_Life_Balance_Rating': 'Work_Life_Balance'
19  })
20
21  # Definisci i bin e le etichette
22  hours_bins = [0, 30, 40, 60, float('inf')] # Aggiunto 0 come primo bin per includere tutti i valori > 0
23  hours_labels = ['Part-time', 'Full-Time', 'Overtime', 'Unknown']
24
25  # Utilizza pd.cut per categorizzare i dati
26  csv['Hours_Worked_Per_Week'] = pd.cut(csv['Hours_Worked_Per_Week'], bins=hours_bins, labels=hours_labels, right=False)
27
28
29  # Categorize 'Age' into bins using pd.cut and assign corresponding labels
30  age_bins = [20, 29, 49, 60, float('inf')]
31  age_labels = ['Youth', 'Adult', 'Senior', 'Unknown']
32  csv['Age'] = pd.cut(csv['Age'], bins=age_bins, labels=age_labels, right=True)
33
34  # Replace 'Prefer not to say' in the Gender column with 'Unknown'
35  csv['Gender'] = csv['Gender'].replace('Prefer not to say', 'Unknown')
36
37  # Categorize 'Years_of_Experience' into bins using pd.cut and assign corresponding labels
38  yoe_bins = [1, 10, 20, 30, 40, float('inf')]
39  yoe_labels = ['Junior', 'Mid', 'Expert', 'Senior', 'Unknown']
40  csv['Years_of_Experience'] = pd.cut(csv['Years_of_Experience'], bins=yoe_bins, labels=yoe_labels, right=True)
41
42  # Map numerical values to descriptive labels for Work-Life Balance
43  wlb_map = {1: 'Minimal', 2: 'Low', 3: 'Moderate', 4: 'High', 5: 'Extreme'}
44  csv['Work_Life_Balance'] = csv['Work_Life_Balance'].map(wlb_map).fillna('Unknown')
45
46  # Map numerical values to descriptive labels for Social Isolation
47  si_map = {1: 'Minimal', 2: 'Low', 3: 'Moderate', 4: 'High', 5: 'Extreme'}
48  csv['Social_Isolation'] = csv['Social_Isolation'].map(si_map).fillna('Unknown')
49
50  # Fill any missing values in 'Physical_Activity' with 'Unknown'
51  csv['Physical_Activity'] = csv['Physical_Activity'].fillna('Unknown')
52
53  csv = csv.fillna('Unknown')
54
55  # Save the cleaned dataframe to a new CSV file
56  csv.to_csv(r"C:\Users\david\OneDrive\Documenti\GitHub\mental-health-IA\database_cleaned.csv", index=False)
57

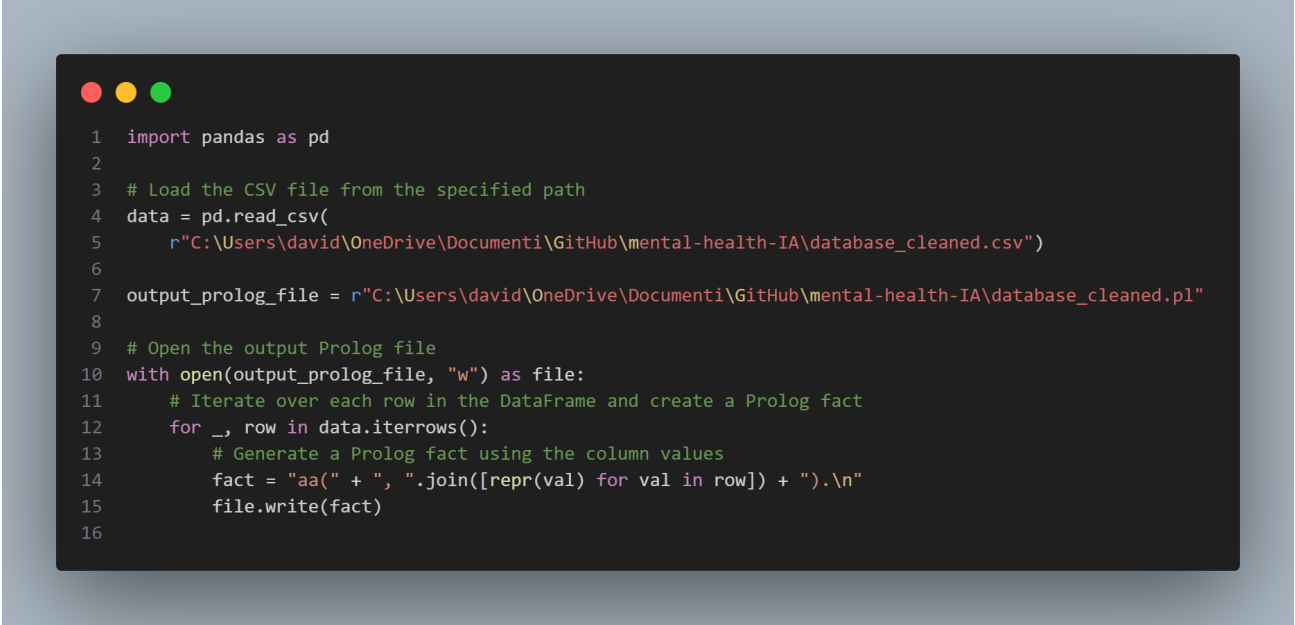
```

Figura 1: Codice utilizzato per la pulizia dei dati

## 2.3 Traduzione da CSV a Prolog

Successivamente, è stato necessario tradurre il file CSV in un formato compatibile con Prolog per facilitare l'analisi dei dati. Per fare ciò, abbiamo utilizzato il seguente script Python che ha letto il file CSV e ha scritto i dati in un file Prolog, creando un fatto per ogni riga del dataset.

Il codice utilizzato per eseguire questa traduzione è il seguente (vedi *Figura 2*):

A screenshot of a code editor with a dark background and light-colored text. The script is a Python program that uses the pandas library to read a CSV file and write its contents into a Prolog file. The code is numbered from 1 to 16. It imports pandas as pd, loads a CSV file from a specific path, opens an output Prolog file, and iterates over each row of the DataFrame to create a Prolog fact using the column values. The fact is written to the output file in the format aa(...).

```
1 import pandas as pd
2
3 # Load the CSV file from the specified path
4 data = pd.read_csv(
5     r"C:\Users\david\OneDrive\Documenti\GitHub\mental-health-IA\database_cleaned.csv")
6
7 output_prolog_file = r"C:\Users\david\OneDrive\Documenti\GitHub\mental-health-IA\database_cleaned.pl"
8
9 # Open the output Prolog file
10 with open(output_prolog_file, "w") as file:
11     # Iterate over each row in the DataFrame and create a Prolog fact
12     for _, row in data.iterrows():
13         # Generate a Prolog fact using the column values
14         fact = "aa(" + ", ".join([repr(val) for val in row]) + ").\n"
15         file.write(fact)
16
```

Figura 2: Codice Python per la traduzione da CSV a Prolog

Questo script:

- Carica il file CSV utilizzando la libreria `pandas`.
- Per ogni riga nel file CSV, crea un fatto Prolog nel formato `aa(...)`.
- Scrive ogni fatto nel file di output con estensione `.pl`.

## 2.4 Creazione dei dataset di Test e Training

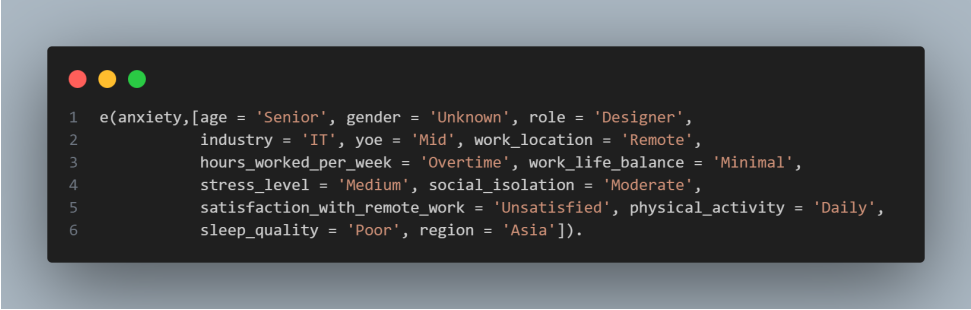
Dopo aver tradotto i dati dal formato CSV a Prolog, il passo successivo è stato quello di suddividere i dati in due dataset separati: uno per il training e uno per il test. Questa suddivisione è fondamentale per poter addestrare e testare correttamente un modello predittivo.

Il processo di suddivisione è stato eseguito utilizzando il codice Prolog, che ha seguito i seguenti passaggi:

- Recupero di tutti i record dal database e rimozione dei duplicati.
- Raggruppamento dei dati in base alla condizione mentale (`Mental_Health_Condition`).
- Suddivisione di ogni gruppo in due sottoinsiemi: uno destinato al training (70%) e uno al test (30%).
- Scrittura dei dati suddivisi in due file separati: `training_set.pl` e `test_set.pl`.

Al termine di questo script, dunque, sono stati generati i due file contenenti i dati nel seguente formato:

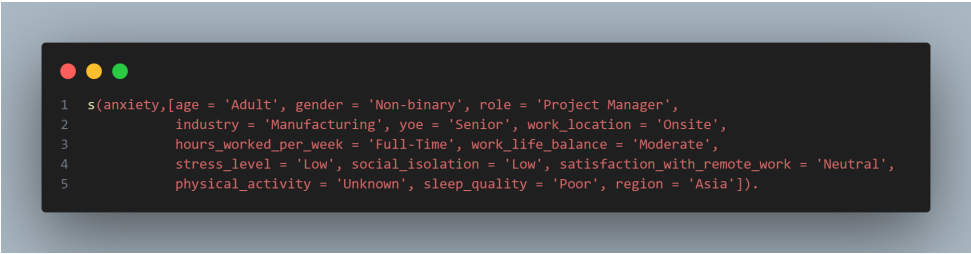
- `training_set.pl`:



```
1 e(anxiety,[age = 'Senior', gender = 'Unknown', role = 'Designer',
2           industry = 'IT', yoe = 'Mid', work_location = 'Remote',
3           hours_worked_per_week = 'Overtime', work_life_balance = 'Minimal',
4           stress_level = 'Medium', social_isolation = 'Moderate',
5           satisfaction_with_remote_work = 'Unsatisfied', physical_activity = 'Daily',
6           sleep_quality = 'Poor', region = 'Asia']).
```

Figura 3: Esempio di record nel training set

- `test_set.pl`:



```
1 s(anxiety,[age = 'Adult', gender = 'Non-binary', role = 'Project Manager',
2           industry = 'Manufacturing', yoe = 'Senior', work_location = 'Onsite',
3           hours_worked_per_week = 'Full-Time', work_life_balance = 'Moderate',
4           stress_level = 'Low', social_isolation = 'Low', satisfaction_with_remote_work = 'Neutral',
5           physical_activity = 'Unknown', sleep_quality = 'Poor', region = 'Asia']).
```

Figura 4: Esempio di record nel test set

Questa suddivisione assicura che ogni condizione mentale sia rappresentata equamente nei due set, garantendo così una corretta validazione del modello.

I file generati, `training_set.pl` e `test_set.pl`, sono quindi pronti per essere utilizzati nella fase successiva della predizione.

## 2.5 Induzione dell'Albero Decisionale con Entropia

L'induzione dell'albero decisionale con entropia è un processo fondamentale per creare modelli predittivi basati su dati strutturati. Questo metodo permette di costruire un albero che classifica gli esempi in base agli attributi, riducendo l'incertezza ad ogni livello dell'albero. Di seguito sono descritti i passaggi principali dell'algoritmo, con i relativi predicati Prolog:

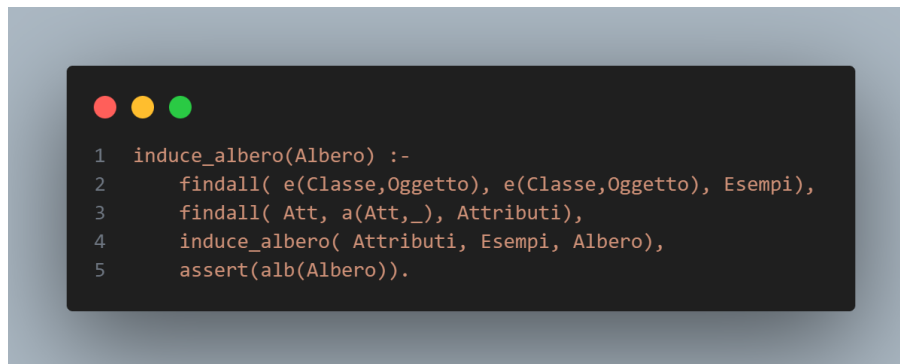
- *Caricamento dei File*: Il programma carica i file necessari all'esecuzione



```
1 :- ensure_loaded(attributes).
2 :- ensure_loaded(training_set).
3 :- ensure_loaded(test_set).
4
5 :- dynamic alb/1.
```

Figura 5: Caricamento file necessari

- *Induzione dell'Albero di Decisione*: La funzione principale per l'induzione dell'albero di decisione è `induce_albero`, in *Figura 6*. Essa esamina tutti gli esempi e gli attributi, e costruisce l'albero ricorsivamente in base alle condizioni di separazione degli esempi.



```
1 induce_albero(Albero) :-
2     findall( e(Classe,Oggetto), e(Classe,Oggetto), Esempi),
3     findall( Att, a(Att,_), Attributi),
4     induce_albero( Attributi, Esempi, Albero),
5     assert(alb(Albero)).
```

Figura 6: Induzione albero

Inoltre, la funzione ricorsiva `induce_albero/3`, presente in *Figura 7*, utilizza quattro casi distinti per gestire l'induzione:

1. Albero = null: quando l'insieme degli esempi è vuoto.
2. Albero = l(Classe): quando tutti gli esempi appartengono alla stessa classe.
3. Albero = t(Attributo, SAlberi): quando si devono separare gli esempi in base a un attributo.
4. Albero = l(Classi): quando non ci sono attributi utili per discriminare ulteriormente.

```

1 induce_albero( _, [], null ) :- !.
2 induce_albero( _, [e(Classe,_)|Esempi], l(Classe)) :-
3     \+ ( member(e(ClassX,_),Esempi), ClassX \== Classe ),!.
4 induce_albero( Attributi, Esempi, t(Attributo,SAlberi) ) :-
5     sceglie_attributo( Attributi, Esempi, Attributo), !,
6     del( Attributo, Attributi, Rimanenti ),
7     a( Attributo, Valori ),
8     induce_alberi( Attributo, Valori, Rimanenti, Esempi, SAlberi).
9 induce_albero( _, Esempi, l(Classi)) :-
10    findall( Classe, member(e(Classe,_),Esempi), Classi).

```

Figura 7: Induzione albero: ricorsione

- *Selezione dell'Attributo Migliore*: Il programma utilizza l'entropia per selezionare l'attributo migliore per la divisione degli esempi. La funzione `sceglie_attributo`, in *Figura 8*, ordina gli attributi in base all'entropia e seleziona quello con la minore entropia.

```

1 sceglie_attributo( Attributi, Esempi, MigliorAttributo ) :-
2     setof( Entropia/A,
3         (member(A,Attributi), disuguaglianza(Esempi,A,Entropia)),
4         [_/MigliorAttributo|_] ).

```

Figura 8: Scelta dell'attributo

- *Calcolo dell'Entropia* La funzione `entropia`, in *Figura 9*, calcola l'entropia di un insieme di esempi basata sulle probabilità delle classi.



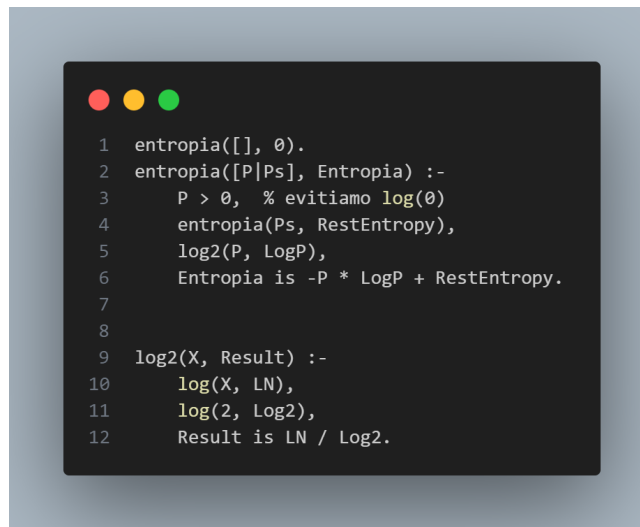


Figura 9: Scelta dell'attributo

- *Classificazione di un Oggetto*: La funzione `classifica`, presente in *Figura 10*, assegna una classe a un oggetto, utilizzando l'albero di decisione. Il programma esplora ricorsivamente l'albero, applicando gli attributi fino a giungere a una classe.

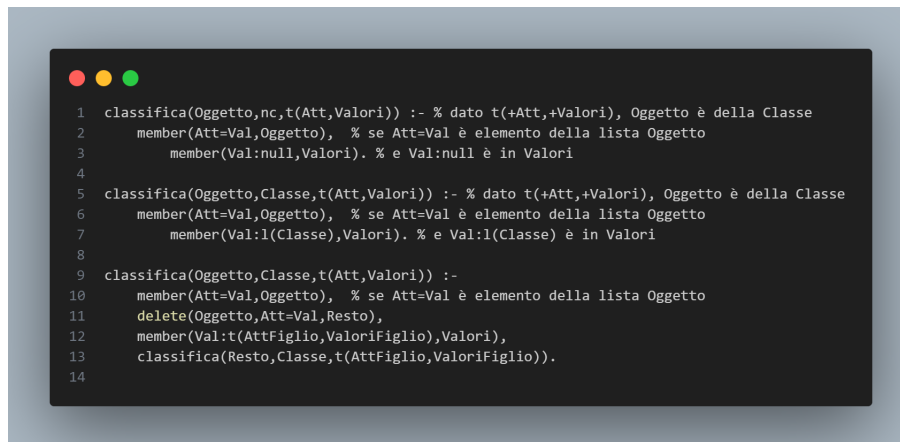


Figura 10: Classificazione

- *Matrice di Confusione*: Per testare l'efficacia dell'albero di decisione, viene utilizzata una matrice di confusione. Il programma calcola diverse metriche, in particolare: l'accuratezza, l'errore, e stampa i risultati.

```

1 stampa_matrice_di_confusione :-
2   alb(Albero),
3   findall(Classe/Oggetto,s(Classe,Oggetto),TestSet),
4   length(TestSet,N),
5   valuta(Albero, TestSet, VD, 0, VA, 0, VB, 0, VU, 0, DA, 0, DB, 0, DU, 0, AD, 0, AB, 0, AU, 0, BD, 0, BA,0, BU,0, UD, 0, UA, 0, UB, 0, NC, 0),
6   A is (VD + VA + VB + VU) / (N - NC), % Accuratezza
7   E is 1 - A, % Errore
8   write('Tests performed: '), writeln(N),
9   write('Tests not classified: '), writeln(NC),
10  write('True depression: '), writeln(VD),
11  write('True anxiety: '), writeln(VA),
12  write('True burnout: '), writeln(VB),
13  write('True unknown: '), writeln(VU),
14  write('Depression classified as anxiety: '), writeln(DA),
15  write('Depression classified as burnout: '), writeln(DB),
16  write('Depression classified as unknown: '), writeln(DU),
17  write('Anxiety classified as depression: '), writeln(AD),
18  write('Anxiety classified as burnout: '), writeln(AB),
19  write('Anxiety classified as unknown: '), writeln(AU),
20  write('Burnout classified as depression: '), writeln(BD),
21  write('Burnout classified as anxiety: '), writeln(BA),
22  write('Burnout classified as unknown: '), writeln(BU),
23  write('Unknown classified as depression: '), writeln(UD),
24  write('Unknown classified as anxiety: '), writeln(UA),
25  write('Unknown classified as burnout: '), writeln(UB),
26  write('Accuracy: '), writeln(A),
27  write('Error: '), writeln(E).

```

Figura 11: Stampa matrice di confusione

inoltre, per stampare la matrice di confusione, è stata implementata la funzione `valuta/3` (Figura 12), che viene utilizzata per calcolare le metriche di performance del modello. Essa confronta le previsioni fatte dall'albero di decisione con i risultati attesi.

- Essa tiene traccia di vari casi, come VD (veri depressioni), VA (veri ansiosi), VB (veri burnout), e VU (veri sconosciuti), per analizzare il comportamento della classificazione.
- Inoltre, calcola anche le predizioni errate, come nel caso in cui la depressione venga classificata come ansia (DA), o l'ansia venga classificata come depressione (AD).

```
1 valuta(Albero, [depression/Oggetto | Coda], VD, VDA, VA, VAA, VB, VBA, VU, VUA, DA, DAA, DB, DBA, DU, DUA, AD, ADA, AB, ABA, AU, AUA, BD, BDA, BA, BAA, BU, BUA, UD, UDA, UA, UAA, UB, UBA, NC, NCA) :-  
2 classifica(Oggetto, depression, Albero), !,  
3 VDA1 is VDA + 1,  
4 valuta(Albero, Coda, VD, VDA1, VA, VAA, VB, VBA, VU, VUA, DA, DAA, DB, DBA, DU, DUA, AD, ADA, AB, ABA, AU, AUA, BD, BDA, BA, BAA, BU, BUA, UD, UDA, UA, UAA, UB, UBA, NC, NCA).  
5  
6 valuta(Albero, [anxiety/Oggetto | Coda], VD, VDA, VA, VAA, VB, VBA, VU, VUA, DA, DAA, DB, DBA, DU, DUA, AD, ADA, AB, ABA, AU, AUA, BD, BDA, BA, BAA, BU, BUA, UD, UDA, UA, UAA, UB, UBA, NC, NCA) :-  
7 classifica(Oggetto, anxiety, Albero), !,  
8 VAA1 is VAA + 1,  
9 valuta(Albero, Coda, VD, VDA, VA, VAA1, VB, VBA, VU, VUA, DA, DAA, DB, DBA, DU, DUA, AD, ADA, AB, ABA, AU, AUA, BD, BDA, BA, BAA, BU, BUA, UD, UDA, UA, UAA, UB, UBA, NC, NCA).  
10  
11 valuta(Albero, [burnout/Oggetto | Coda], VD, VDA, VA, VAA, VB, VBA, VU, VUA, DA, DAA, DB, DBA, DU, DUA, AD, ADA, AB, ABA, AU, AUA, BD, BDA, BA, BAA, BU, BUA, UD, UDA, UA, UAA, UB, UBA, NC, NCA) :-  
12 classifica(Oggetto, burnout, Albero), !,  
13 VBA1 is VBA + 1,  
14 valuta(Albero, Coda, VD, VDA, VA, VAA, VB, VBA1, VU, VUA, DA, DAA, DB, DBA, DU, DUA, AD, ADA, AB, ABA, AU, AUA, BD, BDA, BA, BAA, BU, BUA, UD, UDA, UA, UAA, UB, UBA, NC, NCA).  
15  
16 valuta(Albero, [unknown/Oggetto | Coda], VD, VDA, VA, VAA, VB, VBA, VU, VUA, DA, DAA, DB, DBA, DU, DUA, AD, ADA, AB, ABA, AU, AUA, BD, BDA, BA, BAA, BU, BUA, UD, UDA, UA, UAA, UB, UBA, NC, NCA) :-  
17 classifica(Oggetto, unknown, Albero), !,  
18 VUA1 is VUA + 1,  
19 valuta(Albero, Coda, VD, VDA, VA, VAA, VB, VBA, VU, VUA1, DA, DAA, DB, DBA, DU, DUA, AD, ADA, AB, ABA, AU, AUA, BD, BDA, BA, BAA, BU, BUA, UD, UDA, UA, UAA, UB, UBA, NC, NCA).  
20
```

Figura 12: Alcuni esempi di valutazione delle metriche di performance

---

## 3 Conclusioni

In questa sezione vengono confrontati i risultati ottenuti utilizzando come criteri di selezione l'Entropia e Gini. I dati riportati per entrambe le metodologie sono i seguenti:

### 3.1 Risultati con Entropia

I risultati ottenuti utilizzando l'Entropia sono stati:

- *Accuratezza:* 80.95%
- *Errore:* 19.05%
- *Casi di depressione correttamente classificati:* 296
- *Casi di ansia correttamente classificati:* 301
- *Casi di burnout correttamente classificati:* 266
- *Casi di unknown correttamente classificati:* 280
- *Falsi positivi e negativi:*
  - La depressione è stata classificata come ansia (24), burnout (29) e unknown (20).
  - L'ansia è stata classificata come depressione (22), burnout (22) e unknown (23).
  - Il burnout è stato classificato come depressione (20), ansia (19) e unknown (24).
  - I casi di unknown sono stati classificati come depressione (22), ansia (21) e burnout (23).

### 3.2 Risultati con Gini

I risultati ottenuti utilizzando il criterio di Gini sono i seguenti:

- *Accuratezza:* 81.54%
- *Errore:* 18.45%
- *Casi di depressione correttamente classificati:* 297
- *Casi di ansia correttamente classificati:* 304
- *Casi di burnout correttamente classificati:* 267
- *Casi di unknown correttamente classificati:* 285

- *Falsi positivi e negativi:*

- La depressione è stata classificata come ansia (23), burnout (28) e unknown (23).
- L'ansia è stata classificata come depressione (19), burnout (21) e unknown (25).
- Il burnout è stato classificato come depressione (21), ansia (22) e unknown (18).
- I casi di unknown sono stati classificati come depressione (20), ansia (14) e burnout (27).

### 3.3 Confronto dei risultati

Confrontando i risultati ottenuti utilizzando l'Entropia e Gini, possiamo fare le seguenti osservazioni:

- *Accuratezza e errore:* L'accuratezza ottenuta utilizzando Gini (81.54%) è leggermente superiore a quella ottenuta con l'entropia (80.95%). Di conseguenza, l'errore con Gini è più basso (18.45%) rispetto a quello ottenuto con l'entropia (19.05%).
- *Classificazioni corrette:* Entrambi i modelli hanno mostrato una buona capacità di classificazione, con Gini che ha leggermente superato l'entropia per quanto riguarda la depressione (297 contro 296), l'ansia (304 contro 301) e i casi di "unknown" (285 contro 280). Tuttavia, le differenze nelle classificazioni corrette tra i due metodi sono minime.
- *Errori di classificazione:* I falsi positivi e negativi sono leggermente più equilibrati utilizzando Gini. In particolare, Gini ha commesso meno errori nella classificazione dei casi di unknown come ansia (14 invece di 21) e di burnout come unknown (18 invece di 24), riducendo alcuni degli errori di confusione tra le classi.
- *Robustezza:* Il modello basato su Gini sembra essere leggermente più robusto nell'affrontare la variabilità nei dati, con una minore tendenza a commettere errori di classificazione tra le categorie simili (ad esempio, ansia e depressione).

### 3.4 Esempio di albero decisionale

Indipendentemente dal criterio di suddivisione utilizzato (Entropia o Gini), l'albero decisionale risultante si presenterà secondo una struttura simile, con una serie di suddivisioni basate su variabili come il genere, la qualità del sonno, l'attività fisica e altri fattori legati alla salute mentale e al benessere. La seguente rappresentazione mostra un esempio di come potrebbe essere strutturato l'albero decisionale, evidenziando le principali caratteristiche e decisioni prese in base ai dati.

```
Retail
  region
    Africa
      gender
        Female ==> burnout
        Male
          sleep_quality
            Average ==> depression
            Good ==> unknown
            Poor ==> ???
        Non-binary
          physical_activity
            Daily ==> burnout
            Unknown
              satisfaction_with_remote_work
                Neutral ==> ???
                Satisfied ==> burnout
                Unsatisfied ==> unknown
            Weekly ==> depression
        Unknown
          work_life_balance
            Extreme ==> ???
            High ==> ???
            Low
              age
                Adult ==> depression
                Senior ==> ???
                Youth ==> burnout
                Minimal ==> burnout
                Moderate ==> anxiety
```

### 3.5 Osservazioni finali

In sintesi, entrambi i casi, sia utilizzando l'entropia che utilizzando Gini, hanno mostrato un buon livello di accuratezza, con valori superiori all'80%. Tuttavia, il modello basato sul Gini ha ottenuto prestazioni leggermente migliori, con una maggiore accuratezza (81.54% contro 80.95%) e, naturalmente, un margine di errore inferiore (18.45% contro 19.05%). Le differenze nelle classificazioni corrette tra i due metodi sono minime, ma Gini ha avuto una gestione più efficiente degli errori di classificazione, in particolare per le classi "unknown" e "burnout". Que-

sto suggerisce che l'uso di Gini come criterio di selezione potrebbe offrire risultati leggermente più affidabili in contesti in cui le categorie sono difficili da distinguere.

### 3.6 Possibili sviluppi futuri

Nonostante i buoni risultati ottenuti, ci possono essere diversi aspetti da esplorare per migliorare ulteriormente le prestazioni. Un primo sviluppo da seguire in futuro potrebbe essere l'introduzione di una strategia di *pruning*.

Il *pruning* è una tecnica utilizzata per migliorare l'efficienza e la generalizzazione degli alberi decisionali, riducendo il rischio di overfitting. Durante la costruzione dell'albero, alcuni rami possono diventare molto complessi, creando regole troppo specifiche che si adattano eccessivamente ai dati di training, ma che non generalizzano bene sui dati di test. Il pruning interviene rimuovendo questi rami inutili o ridondanti, semplificando la struttura dell'albero e migliorando la sua capacità di generalizzare su nuovi dati.

Esistono diversi metodi di pruning, tra cui:

- *Pre-pruning*: consiste nell'interrompere la costruzione dell'albero prima che raggiunga una profondità eccessiva, fermandosi quando una divisione non porta a un miglioramento significativo nella qualità della classificazione. Si può fare ad esempio limitando il numero massimo di livelli dell'albero o richiedendo una soglia minima di dati in ciascun nodo.
- *Post-pruning*: avviene dopo che l'albero è stato costruito, rimuovendo successivamente i nodi che non migliorano in modo sostanziale la capacità predittiva del modello. Un approccio comune di post-pruning è il *cost-complexity pruning*, che riduce la complessità dell'albero mantenendo un buon equilibrio tra accuratezza e dimensione del modello.

I benefici del pruning sono molteplici:

- *Miglioramento della generalizzazione*: Rimuovendo rami inutili o eccessivamente specifici, l'albero diventa meno sensibile ai rumori nei dati di addestramento, riducendo il rischio di overfitting.
- *Riduzione della complessità*: Un albero più semplice è più facilmente interpretabile e meno soggetto a errori causati da un'eccessiva complessità.
- *Riduzione dei falsi positivi e negativi*: Un albero potenzialmente più equilibrato e meno complesso potrebbe ridurre i casi di classificazioni errate, migliorando l'accuratezza complessiva del modello, soprattutto nelle classi difficili da distinguere.

In futuro, l'implementazione del pruning potrebbe portare a una maggiore robustezza del modello, con miglioramenti significativi nella sua capacità di generalizzare e nella riduzione dell'errore di classificazione, rendendo l'algoritmo ancora più efficace e pronto per affrontare scenari complessi con dati reali.