SISTEMA INTELLIGENTE PER LA DIAGNOSTICA CLINICA DI MALATTIE MENTALI

Icon 24-25

Repository GitHub
https://github.com/valemaggi/icon

MAGGI VALENTINA M: 758378
v.maggi14@studenti.uniba.it

Sommario

Argomenti di interesse **2**
Introduzione **2**
Definizione del problema 2
Requisiti 2
Preprocessing **3**
Descrizione del dataset 3
Pulizia del dataset 5
Ragionamento **7**
Individuazione dei pattern 7
Regole e classificatore in Prolog 8
Apprendimento Supervisionato **9**
Modelli utilizzati 9
Risultati ottenuti **17**
Confronto tra modelli 17
Valutazione del miglior modello 18

Argomenti di interesse

- Dal capitolo 7: Preprocessing, Metriche di valutazione, Apprendimento Supervisionato, Modelli di apprendimento
- Dal capitolo 15: Prolog
- Dal capitolo 9: Classificatori probabilistici, Naive Bayes

Introduzione

Definizione del problema

La depressione rappresenta una delle principali cause di rischio psicologico, coinvolgendo circa 280 milioni di adulti nel mondo (5% della popolazione). Tra gli studenti, questa patologia assume dimensioni preoccupanti, con tassi di prevalenza stimati tra 12 e il 50% in diverse ricerche globali. È un fenomeno che riguarda principalmente giovani tra i 18 e 25 anni e i livelli di disagio psicologico sono aumentati considerevolmente a seguito della pandemia.

Gli studenti sono particolarmente vulnerabili a questa condizione a causa della combinazione di più fattori: la pressione accademica costante, la difficoltà nel bilanciare studio, lavoro e vita sociale, la scarsa qualità e quantità del sonno e l'alimentazione irregolare. La depressione non colpisce solo l'aspetto emotivo: provoca un netto calo nel rendimento, assenze, difficoltà a concentrarsi e peggioramento delle relazioni interpersonali. Inoltre, il 70% dei giovani con depressione grave non riceve il trattamento adeguato.

Per questi motivi, è cruciale intervenire tempestivamente. La diagnosi precoce può prevenire conseguenze gravi e permettere l'attivazione di supporti psicologici o terapeutici prima che i sintomi si aggravino.

L'obiettivo di questo lavoro è quello di fornire un mezzo efficace per il riconoscimento della depressione tra gli studenti a fini preventivi ed evidenziare le relazioni che intercorrono tra i diversi fattori analizzati.

Requisiti

I requisiti necessari per l'esecuzione, trovati con la libreria **pipreqs**, sono:
matplotlib==3.10.3
numpy==2.3.1
pandas==2.3.0
pyswip==0.3.2
scikit_learn==1.7.0
seaborn==0.13.2

Preprocessing

Descrizione del dataset

Link del dataset su Kaggle:
https://www.kaggle.com/datasets/adilshamim8/student-depression-dataset

Il dataset scelto fornisce una serie di informazioni riguardanti un campione di studenti, con lo scopo di analizzare i potenziali fattori e comportamenti correlati al disturbo depressivo. La presenza di diverse features di input e di una classe target bilanciata, ha reso il dataset adatto ad un task di classificazione binaria.

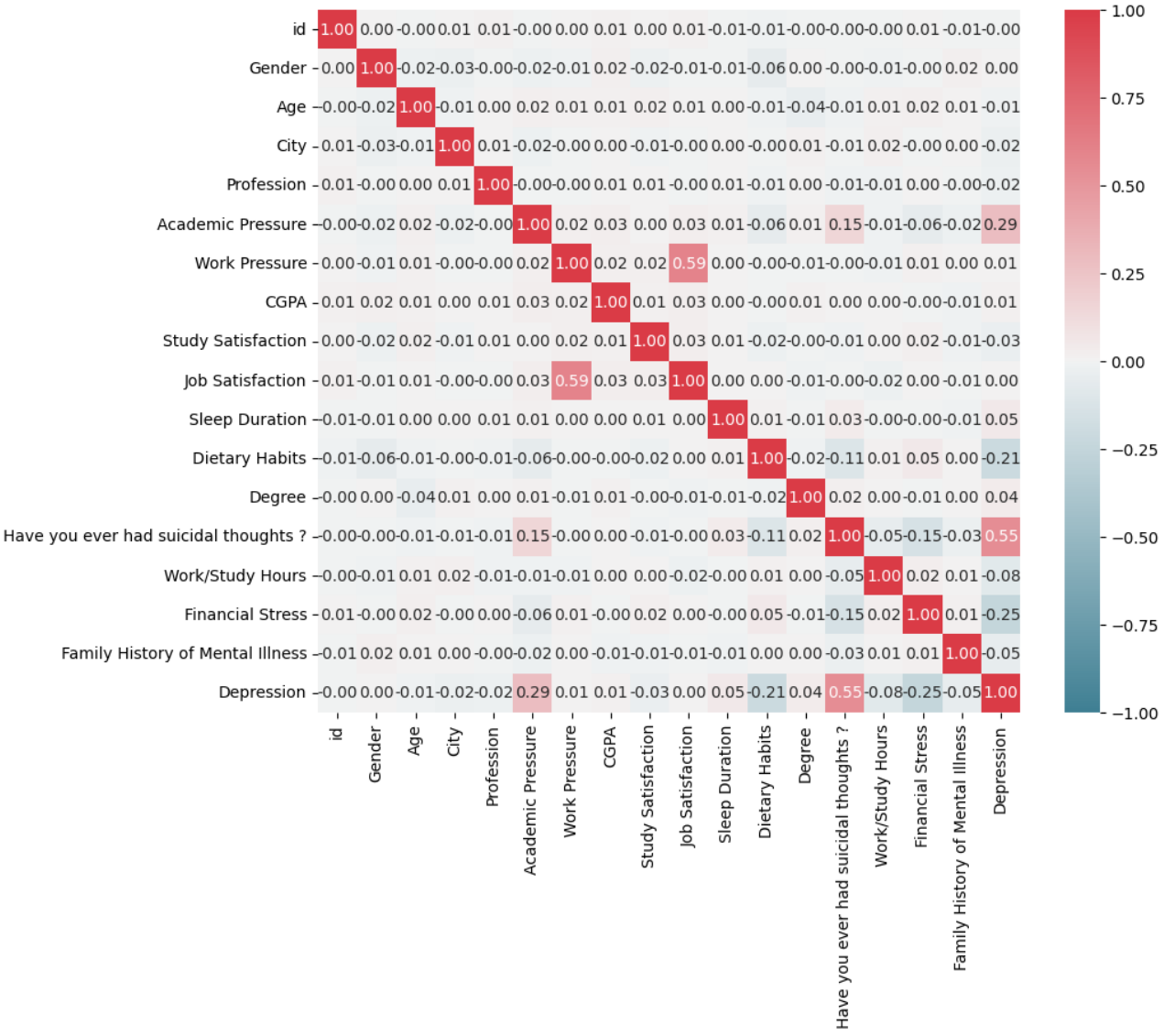
Il dataset scelto è composto da 27902 righe e 18 colonne, di cui 17 per features di input e una per il target binario. Nello specifico:

- **Id:** identifica univocamente gli studenti
- **Gender:** specifica il genere dello studente
- **Age:** specifica l'età dello studente in un range da 18 a 34 anni
- **City:** indica il luogo di residenza dello studente
- **Profession:** indica la professione (per la maggior parte studenti, senza altri lavori)
- **Academic Pressure:** misura la pressione accademica subita dallo studente in un range da 0 a 5
- **Work Pressure:** misura la pressione lavorativa subita dallo studente in un range da 0 a 5
- **CGPA:** identifica la performance accademica in una scala da 0 a 10
- **Study Satisfaction:** indica il grado di soddisfazione dello studente nell'ambito scolastico
- **Job Satisfaction:** indica il grado di soddisfazione dello studente nell'ambito lavorativo
- **Sleep Duration:** tiene conto di quante ore lo studente dorme mediamente
- **Dietary Habits:** indica le abitudini alimentari, suddivise in Healthy/Moderate/Unhealthy
- **Degree:** specifica il tipo di istruzione conseguito
- **Have you ever had suicidal thoughts ?:** evidenzia se lo studente abbia mai avuto pensieri suicidi, con valori binari (Yes/No)
- **Work/Study Hours:** tiene conto di quante ore lo studente impiega nello studio o nel lavoro
- **Financial Stress:** misura lo stress relativo alle risorse finanziarie

- **Family History of Mental Illness:** indica se membri della famiglia dello studente hanno ricevuto una diagnosi per disturbi mentali, con valori binari (Yes/No)
- **Depression:** variabile target binaria che indica se lo studente soffre di depressione (1) o no (0)

Matrice di correlazione

Per farsi una prima idea delle relazioni tra le features del dataset, è stata realizzata una matrice di correlazione:

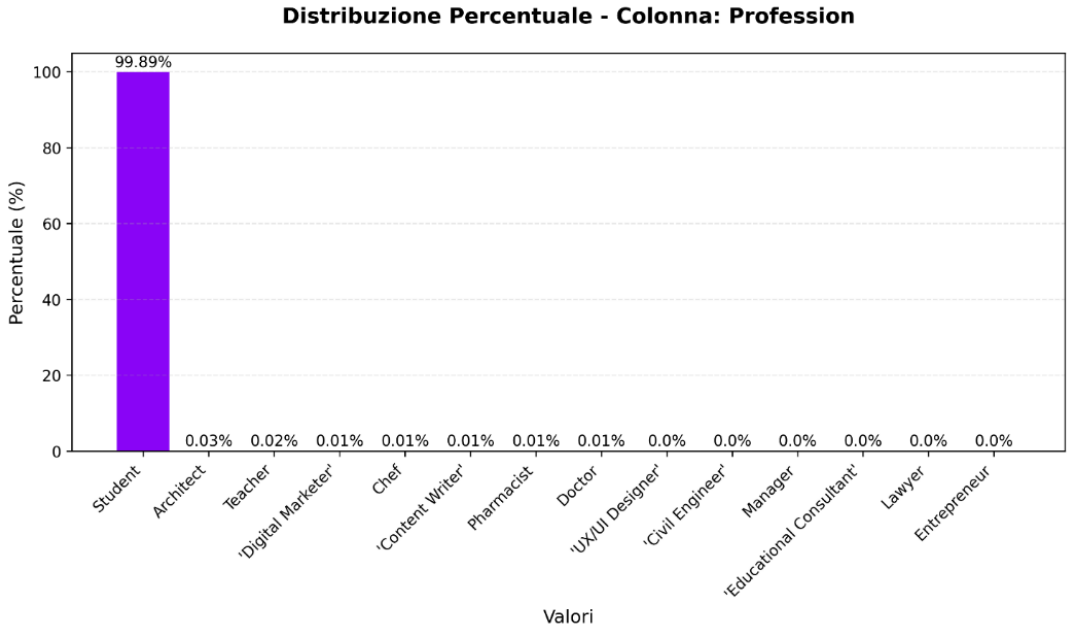


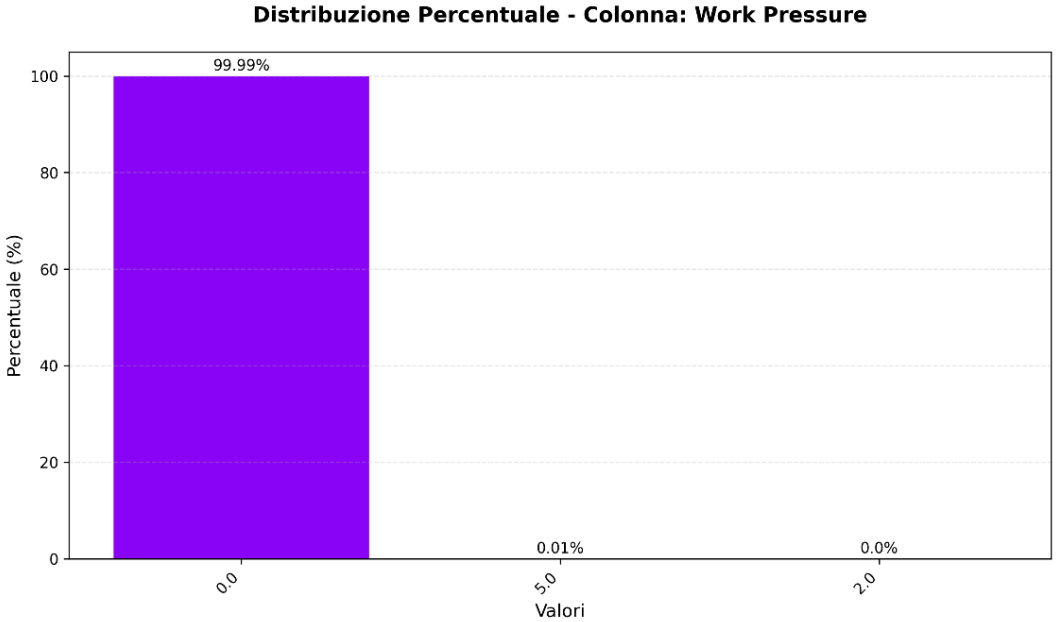
Osservando la matrice si possono notare delle relazioni più forti tra certe variabili del dataset:

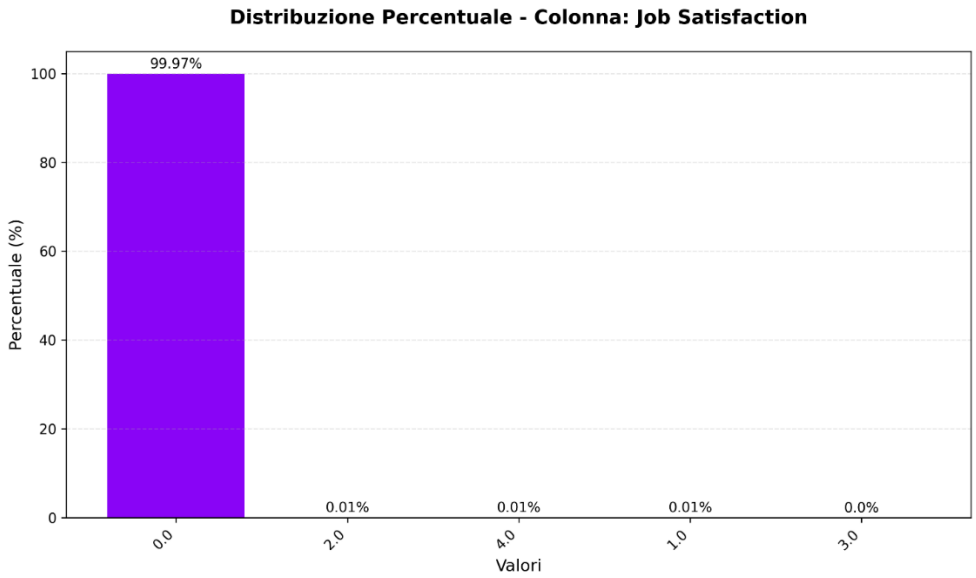
"Depression" e "Academic Pressure"
"Depression" e "Have you ever had suicidal thoughts?"
"Depression" e "Financial Stress"
"Depression" e "Dietary Habits"

Pulizia del dataset

Analizzando il dataset, sono state individuate tre colonne con valori quasi tutti uguali. In particolare, le colonne “Profession”, “Work Pressure” e “Job Satisfaction” avevano rispettivamente i valori “Student” e “0,0” presenti più del 99% delle volte.







Queste percentuali hanno evidenziato come la maggior parte degli studenti inseriti nel dataset non abbia un lavoro. È stato scelto, quindi, di eliminarle dal dataset originale e concentrarsi sugli attributi inerenti all'ambito scolastico piuttosto che lavorativo. Conseguentemente, la colonna "Work/Study Hours" è stata rinominata "Study Hours" - oltre ad aver rinominato anche la colonna "Have you ever had suicidal thoughts ?" per evitare eventuali problemi durante l'elaborazione del dataset. I caratteri di punteggiatura, come il punto interrogativo, possono infatti causare errori o comportamenti imprevisti durante l'accesso ai nomi delle colonne.

Successivamente sono stati individuati valori come "?" e "Others", ritenuti poco informativi (oltre che scarsamente utilizzati all'interno del dataset), ed eliminate le relative righe.

Le modifiche sono state salvate in: **'./datacleaning/dataset.csv'**

Inoltre, per poter addestrare i modelli che non accettano stringhe in input, il dataset è stato ulteriormente modificato trasformando i **dati da stringhe a valori numerici**. In particolare, sono stati utilizzati dizionari per la maggior parte delle colonne, aventi un limitato numero di valori possibili. Per le colonne "City" e "Degree", caratterizzate da una elevata molteplicità di termini, la trasformazione è stata effettuata con la funzione **fit_transform()** della classe **LabelEncoder**.

Le modifiche sono state salvate in: **'./apprendimento_supervisionato/dataset_model.csv'**

Un ulteriore lavoro sul dataset è stato svolto sfruttando la tecnica del **binning**.
Il binning è una tecnica di feature engineering che raggruppa diversi sotto-intervalli numerici in bin, nel mio caso, trasformando i dati numerici in dati categorici. Nello specifico sono state modificate le colonne "Age", "Academic Pressure", "CGPA" e "Study Satisfaction" per rendere il dataset più leggibile e facilitare la fase di ricerca di pattern (che verrà descritta nel prossimo capitolo).

Le modifiche sono state salvate in: **'./ragionamento/bin_dataset.csv'**

Ragionamento

Individuazione dei pattern

La nuova versione del dataset, più leggibile e chiara, è stata utilizzata come punto di partenza per un'analisi più approfondita dei dati. L'analisi dei pattern è stata condotta per individuare regolarità significative, con lo scopo di: estrarre **pattern frequenti**, generare **regole simboliche** in Prolog e calcolare **probabilità a priori** da utilizzare come pesi in un modello di classificazione bayesiano (descritto nel capitolo sull'apprendimento supervisionato).

Una prima analisi ha portato alla creazione di un file **'pattern_analysis.txt'**, utile a studiare le relazioni tra features e facilmente consultabile per individuare quelle di maggiore influenza.

Successivamente, è stata svolta un'analisi più dettagliata. Per ciascuna combinazione di 2 o 3 colonne di feature, è stata eseguita una groupby sul dataframe, calcolando la frequenza di occorrenza di ogni combinazione di valori. Dei risultati trovati, sono stati selezionati soltanto i pattern con:

- supporto minimo (numero di occorrenze): 2000
- confidence ≥ 0.75

A seguito di alcuni test, tali valori sono risultati i migliori.

La confidence è calcolata secondo la formula:

$$\text{Confidence } (A \rightarrow B) = \frac{\text{Support}(A \cup B)}{\text{Support}(A)}$$

dove A è la combinazione dei valori delle prime colonne (antecedente) e B è il valore dell'ultima colonna nella combinazione (conseguente).

Da questo insieme di pattern più ristretto, sono state poi generate delle regole logiche, spiegate più dettagliatamente nel paragrafo successivo.

Inoltre, è stato calcolato il numero totale di occorrenze per ciascun valore della classe target Depression, in base alla presenza di pattern validi, per poter stimare le **probabilità a priori** di tale classe.

$$P(\text{Depression} = 1) = \frac{\text{count}(\text{Depression}=1)}{\text{count}(\text{Depression}=0) + \text{count}(\text{Depression}=1)}$$

$$P(\text{Depression} = 0) = 1 - P(\text{Depression} = 1)$$

Il risultato è stato salvato nel file **'prior_probabilities.json'** e poi caricato dal modello Naive Bayes tramite il parametro priors della classe GaussianNB di Scikit-learn.

Regole e classificatore in Prolog

I pattern trovati nell'analisi sono stati trasformati in regole logiche (solo per i pattern con Depression = 1). La regola per la variabile Depression = 0 è stata invece ricavata, come negazione di Depression = 1.

```
depression('1') :- academic_pressure('alta').
depression('1') :- financial_stress('5.0').
depression('1') :- have_you_ever_had_suicidal_thoughts_('yes').
depression('1') :- study_satisfaction('bassa'), dietary_habits('unhealthy').
depression('1') :- age('universitario'), dietary_habits('unhealthy').
depression('1') :- sleep_duration('less_than_5_hours'), dietary_habits('unhealthy').

depression('0') :- \+ depression('1').
```

A seconda del numero minimo di occorrenze e confidence selezionati, vengono trovati diversi pattern e conseguentemente diverse regole. Se una regola è costituita da un unico valore, non vengono prese in considerazione eventuali regole risultanti da combinazioni di quel valore con altri.

Questa analisi ha individuato come principali fattori di depressione, presi singolarmente, valori alti delle colonne "Academic Pressure" e "Financial Stress" e *yes* di "Have you ever had suicidal thoughts ?". Questa relazione trova la sua conferma anche all'interno della matrice di correlazione, realizzata nella prima fase di analisi del dataset (Descrizione del dataset).

Inoltre, le regole trovate hanno costituito la base per la creazione di un **classificatore in Prolog**, il cui funzionamento viene spiegato in seguito.

Il dataset viene caricato, assicurandosi che i valori siano trattati come stringhe per evitare problemi durante la normalizzazione. Ogni valore viene quindi trasformato, per garantirne la compatibilità con la sintassi di Prolog.

Successivamente, si analizza il file contenente le regole Prolog per estrarre automaticamente i predicati che vengono utilizzati nel corpo delle regole. Questa estrazione consente di sapere quali variabili devono essere presenti nei fatti Prolog per rendere possibile il ragionamento.

Per ogni riga del dataset, si procede in questo modo:

- si rimuovono eventuali fatti precedenti dal motore logico (tramite **retractall**)
- si inseriscono i nuovi fatti generati dalla riga corrente con **assertz**
- si interroga Prolog con una query chiedendo il valore per il predicato **depression(X)**
- il valore di X (0 o 1) viene utilizzato come previsione

Risultati ottenuti da Prolog:

```
predicted_depression
1    22643
0     5258
```

Distribuzione reale della variabile target:

```
Depression
1    16336
0    11565
```

Si valutano, quindi, le prestazioni del classificatore confrontando le predizioni con le etichette reali della colonna “Depression” del dataset: '**./ragionamento/bin_dataset.csv**'.

Si può notare come questo classificatore in Prolog, a differenza dei successivi che prenderanno in input unicamente valori numerici, sia in grado di eseguire l’analisi su un dataset più facilmente interpretabile.

Per la valutazione, sono state calcolate le metriche di accuracy e recall per poterle poi confrontare con quelle degli altri classificatori testati.

```
Accuracy: 0.7450
Recall: 0.9760
```

L’obiettivo di questo approccio era quello di sfruttare le regole di Prolog per realizzare un classificatore interpretabile, capace di fare predizioni, ma anche spiegare in modo più leggibile le relazioni tra le caratteristiche del dataset.

Apprendimento supervisionato

L’apprendimento supervisionato è stato applicato a un **task di classificazione binaria**: prevedere la presenza (1) o assenza (0) di sintomi depressivi tra gli studenti. Le feature di input includono attributi eterogenei relativi al profilo demografico, accademico e comportamentale degli individui.

Modelli utilizzati

Sono stati utilizzati quattro modelli di classificazione: **Logistic Regression, Gaussian Naive Bayes, Random Forest** e **Support Vector Machine**.

Data la natura del problema, ovvero la rilevazione di uno stato depressivo potenzialmente grave, si è posta particolare attenzione alla minimizzazione dei falsi negativi, ritenendo preferibile segnalare un caso come potenzialmente a rischio, piuttosto che non rilevare un caso problematico. Per questo motivo, i modelli sono stati addestrati e testati assegnando diversi **pesi** al fine di migliorare il recall.

Per valutare le prestazioni dei modelli e ridurre la varianza legata alle previsioni del modello, è stata utilizzata la **k-fold cross validation** con 10 fold, assicurando una valutazione più affidabile e generalizzabile delle prestazioni dei modelli.

Le metriche di valutazione scelte per il confronto tra modelli sono *accuracy* e *recall*, ponendo come obiettivo l’aumento di quest’ultima (data la natura del problema affrontato).

In seguito, verranno analizzati nel dettaglio i modelli implementati.

Logistic Regression

Il Logistic Regression è un modello di classificazione adatto a problemi di classificazione binaria.

Dopo aver separato le feature dal target, i dati vengono suddivisi in train e test con un test size di 0,2.

Inoltre, sono stati inseriti e testati diversi pesi per la variabile target, per assegnare un peso maggiore alla classe positiva (depression = 1). A seguito di alcune prove, è stato deciso di assegnare questi pesi: **class_weight= {0: 2, 1: 5}**, poiché risultavano in valutazioni migliori.

```
# Split del dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Peso delle classi per la variabile target
class_weight={0: 2, 1: 5}

lr=LogisticRegression(class_weight=class_weight, random_state=42)
```

- **Risultati** del modello prima di definire i pesi:

  Model accuracy score with default hyperparameters: 0.8568
  Model recall score with default hyperparameters: 0.8991

- **Risultati** del modello con i pesi specifici:

  Model accuracy score with default hyperparameters: 0.8308
  Model recall score with default hyperparameters: 0.9540

Successivamente, viene effettuata la ***GridSearch*** per la scelta dei migliori iperparametri del modello.

Iperparametri scelti:

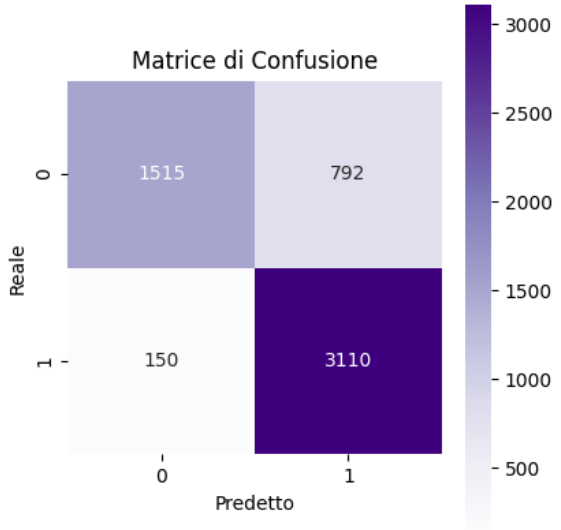
```
param_grid = {
    'C': [0.1],
    'solver': ['saga'],
    'penalty': ['l2'],
    'max_iter': [1000000]
}
```

Infine, vengono stampati e salvati i **risultati finali** per la valutazione del modello.

- **Risultati** del modello con gli iperparametri scelti:

  Model accuracy score after grid search: 0.8302
  Model recall score after grid search: 0.9555

	precision	recall	f1-score	support
0	0.91	0.65	0.76	2307
1	0.80	0.96	0.87	3260
accuracy			0.83	5567
macro avg	0.85	0.80	0.81	5567
weighted avg	0.84	0.83	0.82	5567



Gaussian Naive Bayes

Il Naive Bayes è un classificatore probabilistico basato sul teorema di Bayes e sull'assunzione di indipendenza tra le features. La versione *Gaussian* assume che i dati seguano una distribuzione gaussiana (verosimile per una grande quantità di dati).

Per l'implementazione, i passaggi iniziali sono gli stessi del modello precedente.

In questo caso, però, invece che pesi definiti arbitrariamente, sono state inserite delle probabilità a priori precedentemente calcolate. Come già spiegato nel capitolo *Ragionamento,* l'analisi dei pattern fatta sul dataset è servita anche a trovare le probabilità a priori della classe target.

```
Probabilità a priori:
  1: 0.67447
  0: 0.32553
```

- **Risultati** del modello prima di definire priors:

  Model accuracy score with default hyperparameters: 0.8525
  Model recall score with default hyperparameters: 0.8776

- **Risultati** del modello con priors:

  Model accuracy score with default hyperparameters: 0.8516
  Model recall score with default hyperparameters: 0.9025

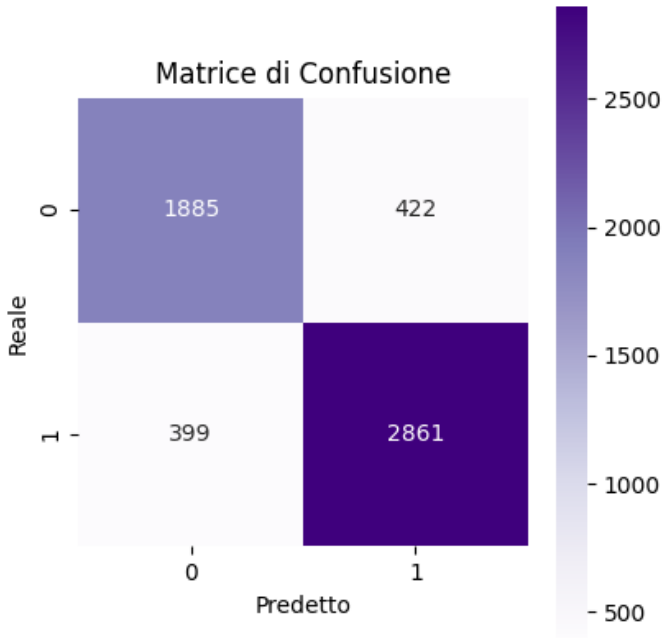
Le probabilità a priori calcolate hanno portato ai migliori risultati e sono state quindi scelte come valore per l'iperparametro priors del modello.

Ecco, infine, i **risultati** ottenuti.

- **Risultati** del modello con gli iperparametri scelti:

  Model accuracy score after grid search: 0.8516
  Model recall score after grid search: 0.9025

	precision	recall	f1-score	support
0	0.85	0.78	0.81	2307
1	0.85	0.90	0.88	3260
accuracy			0.85	5567
macro avg	0.85	0.84	0.85	5567
weighted avg	0.85	0.85	0.85	5567



Random Forest

Il Random Forest è un ensemble learner costituito da un insieme di alberi di decisione. La predizione finale è ottenuta in base al voto in maggioranza degli alberi.

Per questo modello, a differenza degli altri, si è preferito non assegnare pesi specifici alla classe target. Infatti, testando diversi pesi, il recall peggiorava (con un lieve miglioramento dell'accuracy).

- **Risultati** del modello prima di definire i pesi:

  Model accuracy score with default hyperparameters: 0.8493
  Model recall score with default hyperparameters: 0.8926

- **Risultati** del modello con i pesi specifici:

  Model accuracy score with default hyperparameters: 0.8421
  Model recall score with default hyperparameters: 0.8676

Successivamente, la ***GridSearch*** ha portato alla scelta di questi iperparametri per il modello.

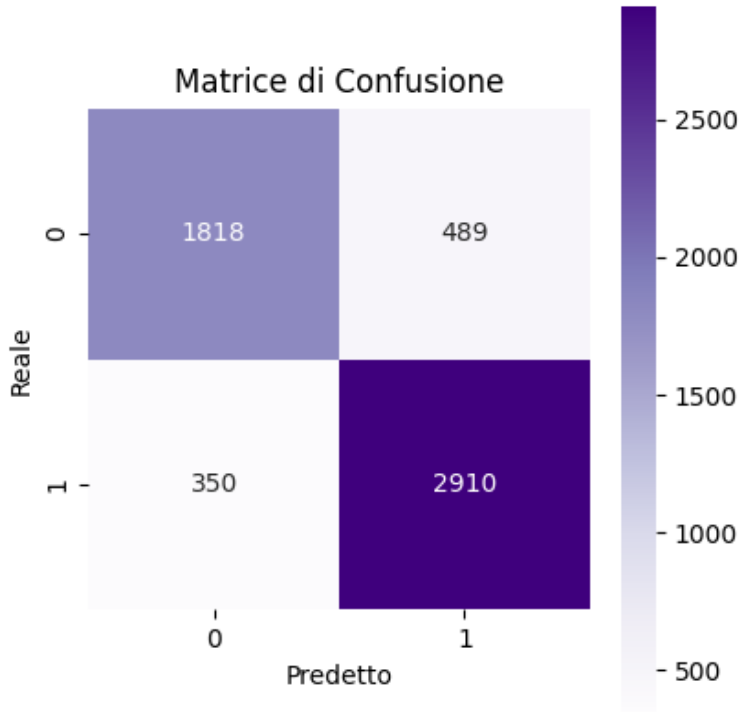
```
param_grid = {
    'n_estimators': [200],
    'criterion': ['entropy'],
    'max_depth': [25]
}
```

Infine, sono riportati i **risultati**.

- **Risultati** del modello con gli iperparametri scelti:

  Model accuracy score after grid search: 0.8511
  Model recall score after grid search: 0.8951

	precision	recall	f1-score	support
0	0.84	0.79	0.81	2307
1	0.86	0.90	0.88	3260
accuracy			0.85	5567
macro avg	0.85	0.84	0.85	5567
weighted avg	0.85	0.85	0.85	5567



Support Vector Classifier

L'SVM è un algoritmo che trova il confine ottimale tra classi attraverso l'iperpiano di massimo margine.

L'implementazione del modello avviene in modo analogo alle precedenti. Anche in questo caso, il dataset viene suddiviso e vengono assegnati dei pesi alla classe target.

```
# Split del dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Peso delle classi per la variabile target
class_weight={0: 2, 1: 5}

svc=SVC(class_weight=class_weight)
```

- **Risultati** del modello prima di definire i pesi:

  Model accuracy score with default hyperparameters: 0.8568
  Model recall score with default hyperparameters: 0.9040

- **Risultati** del modello con i pesi specifici:

  Model accuracy score with default hyperparameters: 0.8259
  Model recall score with default hyperparameters: 0.9620

Successivamente, viene effettuata la ***GridSearch*** per la scelta dei migliori iperparametri del modello.

Iperparametri scelti:

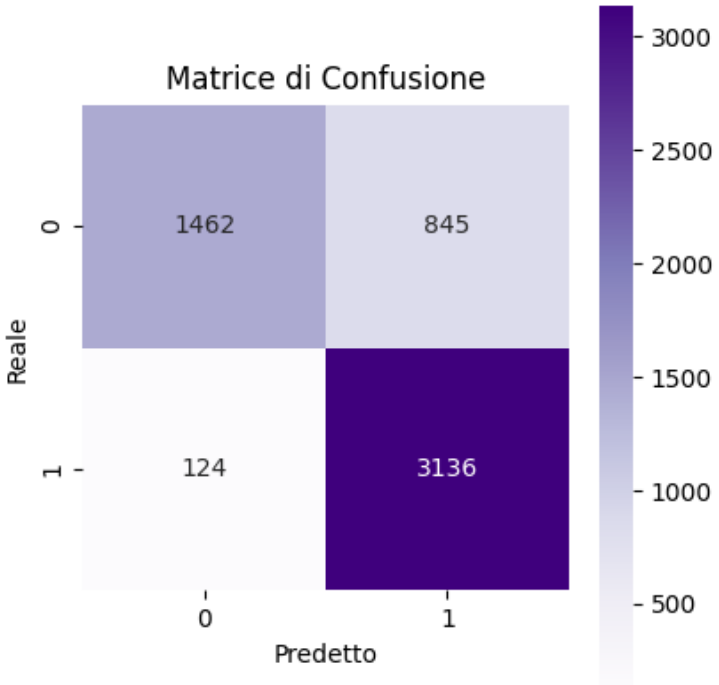
```
param_grid = {
    'C': [10],
    'kernel': ['rbf'],
    'gamma': ['scale'],
    'decision_function_shape': ['ovo']
}
```

Vengono quindi riportarti i **risultati** ottenuti.

- **Risultati** del modello con gli iperparametri scelti:

  Model accuracy score after grid search: 0.8313
  Model recall score after grid search: 0.9601

	precision	recall	f1-score	support
0	0.92	0.65	0.76	2307
1	0.79	0.96	0.87	3260
accuracy			0.83	5567
macro avg	0.86	0.80	0.82	5567
weighted avg	0.85	0.83	0.82	5567



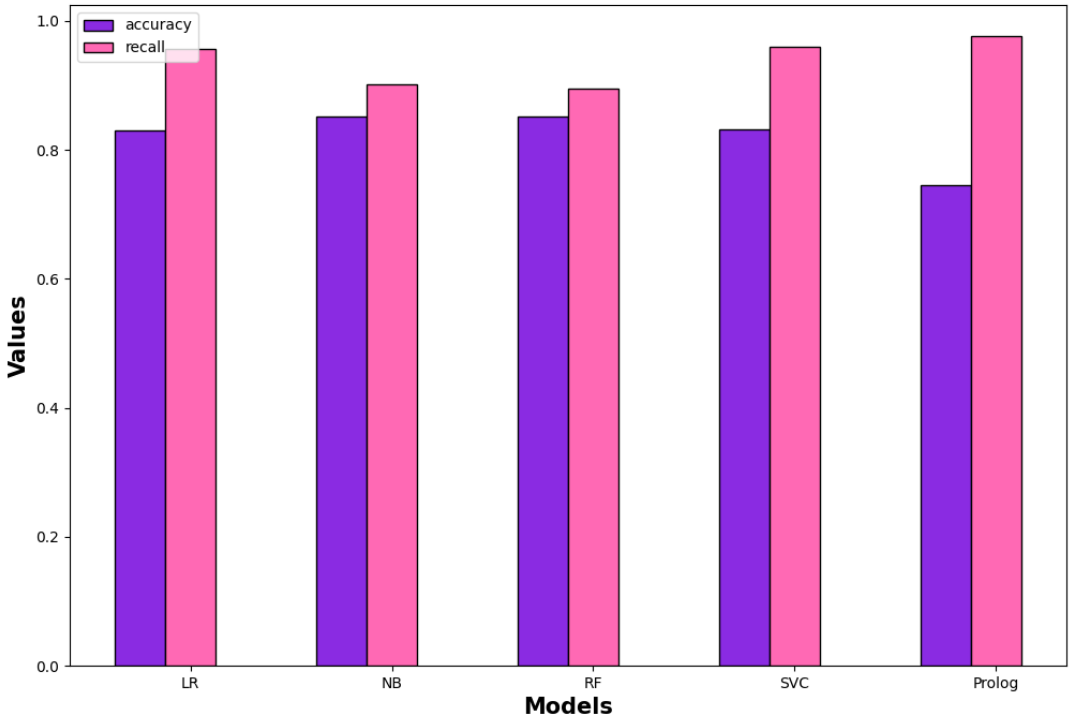
Risultati ottenuti

Confronto tra modelli

I risultati ottenuti dai diversi classificatori sono stati salvati all'interno di un file txt, per poter essere facilmente consultati e utilizzati per la creazione di un grafico a barre.

Il contenuto del file txt è:

RF: 0.851 0.895
NB: 0.852 0.902
LR: 0.83 0.956
SVC: 0.831 0.96
Prolog: 0.745 0.976



In generale, ciascuno dei modelli presenta compromessi diversi tra accuratezza e recall.

Naive Bayes ha il miglior equilibrio tra accuracy e recall, con prestazioni leggermente superiori al Random Forest.

Logistic Regression e SVC si distinguono per un recall molto alto, che indica una forte capacità nel riconoscere i casi positivi, a scapito di una lieve perdita di accuracy.

Il modello basato su Prolog, pur ottenendo il miglior recall, ha un'accuracy più bassa, a testimonianza di una classificazione nel complesso meno precisa.

Valutazione del miglior modello

Dal confronto dei vari modelli, è emerso che la scelta ottimale sia **SVC**. Esso risulta il modello migliore all'interno del contesto clinico: massimizza il recall, mantenendo una buona accuracy. Il maggior compromesso nell'utilizzo del modello SVC risulta essere la complessità computazionale più elevata rispetto ad altri modelli proposti, rendendo limitata la scalabilità su grandi dataset.

Il modello ha ottenuto:

- Recall = 0,960, il secondo valore più alto (dopo Prolog)
- Accuracy = 0,831, leggermente inferiore a due modelli, che però hanno un recall inferiore

In generale, il modello mantiene un bilanciamento efficace tra accuracy e recall e risulta adatto a compiti di classificazione in ambito sanitario.