Università degli Studi di Bari *Aldo Moro*

AI.rrigation
Sistema per la predizione della quantità di acqua necessaria alle colture

Repository GitHub:
https://github.com/massaro-m/AI.rrigation

Colab Notebook:
NotebookCondiviso

Esame di
Ingegneria della Conoscenza

Michelangelo Massaro
Matr. 779603
m.massaro50@studenti.uniba.it

A.A. 2025/2026

Indice

1 Introduzione

AI.rrigation è un sistema progettato per la predizione della necessità idrica di una coltura in uno specifico campo, sulla base di diverse caratteristiche, tra cui il tipo di terreno, le condizioni climatiche e il tipo di irrigazione adottato.
Il sistema prevede la sperimentazione di diversi modelli di apprendimento automatico valutando differenti configurazioni al fine di individuare la soluzione con le prestazioni migliori.
Oltre agli approcci di Machine Learning tradizionali, vengono impiegate anche tecniche di apprendimento probabilistico, tra le quali le Bayesian Network (BN) utilizzate oltre che come modello di classificazione, anche come strumento per la rappresentazione esplicita della conoscenza disponibile

NOTE TECNICHE Il sistema è stato sviluppato in linguaggio **Python**, ideale per applicazioni di questo tipo grazie alle numerose librerie dedicate che è possibile importare, in ambiente **Google Colab** che non richiede alcuna configurazione.

2 Sommario

Il caso di studio è realizzato per mettere in pratica le conoscenze apprese nel corso di *Ingegneria della Conoscenza* del corso di laurea in *Informatica* dell'Università di Bari tenuto dal prof. Nicola Fanizzi.
I concetti richiamati nel caso di studio sono:

- **Apprendimento Supervisionato (cap.7 [1])**: si implementano tre diversi modelli di machine learning (Random Forest, Decision Tree e AdaBoost) che vengono addestrati e valutati utilizzando il dataset a disposizione. Per ciascun modello si esplorano diverse configurazioni di iperparametri, al fine di individuare la versione con le migliori prestazioni. La valutazione tiene conto di differenti suddivisioni dei dati in training set e test set, includendo una fase di validazione per garantire una stima robusta delle prestazioni e ridurre il rischio di overfitting. Prima dell'addestramento, ai dati si applicano operazioni di pre-processing per renderli adatti all'applicazione dei modelli.

- **Rappresentazione della Conoscenza (capp.9-10 [2][3])**: si rappresenta la conoscenza come un grafo orientato aciclico (DAG) tramite la costruzione di due Belief Network usando due diverse misure di valutazione. Si confrontano rispetto alla loro complessità. Questa fase è preceduta da un'operazione di discretizzazione delle variabili numeriche del dataset.

- **Apprendimento Probabilistico (capp.9-10 [2][3])**: la migliore BN appresa precedentemente viene impiegata come classificatore e le sue prestazioni vengono analizzate su diversi test set. Successivamente, si implementa un algoritmo di classificazione probabilistica basato su Naive Bayes, valutato nello stesso modo.

Di tutto ciò si parla nel dettaglio nelle sezioni a seguire.

3 Il Dataset

Il dataset di partenza è stato trovato su Kaggle. Dopo averlo scaricato, è stato caricato sul repository GitHub.
Si compone di 19 colonne e 10000 record/esempi con la relativa variabile target. Le 18 **FEATURE** si riferiscono a vari aspetti. Di seguito, si riportano descrizione e tipo di ogni feature.

- **SUOLO**
  - **Discrete**
    - `Soil_Type`: il tipo di suolo
    - `Mulching_Used`: indica l'uso della pacciamatura.
    - `Season`: il paese in cui si trova
  - **Continue**
    - `Soil_Moisture`: l'umidità del suolo
    - `Organic_Carbon`: la quantità di carbonio organico
    - `Electrical_Conductivity`: la conduttività elettrica
    - `Field_Area_hectare`: l'area del campo (in ettari)
- **CLIMA** Feature continue
  - `Temperature_C`: la temperature rilevata (in gradi Celsius)
  - `Humidity`: l'umidità rilevata
  - `Rainfall_mm`: la quantità di pioggia caduta (in millimetri)
  - `Sunlight_Hours`: le ore di luce
  - `Wind_Speed_kmh`: la velocità del vento rilevata (in km/h)
- **IRRIGAZIONE**
  - `Irrigation_Type`: il tipo di irrigazione. E' discreta
  - `Water_Source`: la fonte da cui è attinta l'acqua. E' discreta
  - `Previous_Irrigation_mm`: la quantità di acqua della precedente irrigazione (in mm). E' continua
- **COLTURA** Feature discrete
  - `Crop_Type`: il tipo di coltura
  - `Crop_Growth_Stage`: la fase della crescita in cui si trova
  - `Region`: l'esposizione (Nord/Sud/Ovest/Est/Centro)

La variabile **TARGET** è `Irrigation_Need`. E' una variabile categorica che può assumere come possibili valori **LOW, MEDIUM, HIGH**.

4 Analisi e pre-processing dei dati

Prima di passare all'implementazione di quanto previsto, sono state attuate operazioni di analisi della conoscenza a disposizione, intervenendo con operazioni di pre-processing dove necessario al fine di migliorare l'efficienza dei modelli da addestrare.
Per prima cosa è stata verificata l'eventuale presenza di valori nulli nelle colonne ed esempi duplicati. Non erano presenti. Poi, è stato verificato il bilanciamento delle classi: possiamo notare (Figura 1) la presenza di una classe minoritaria rispetto alle altre. Sulla base dei risultati che si otterranno, si valuterà un possibile bilanciamento con dati sintetici.

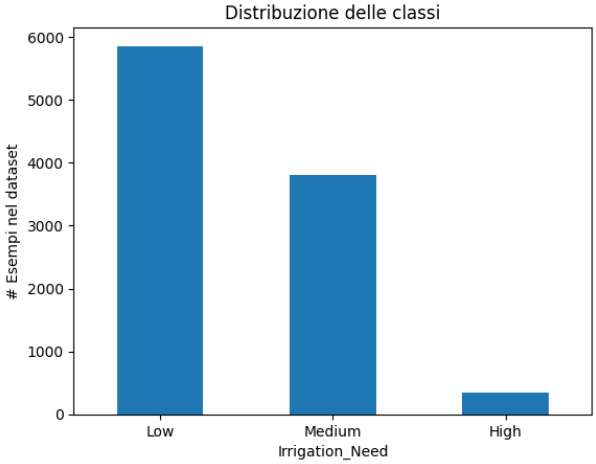


Figura 1: Bilanciamento delle classi

Successivamente, sfruttando i metodi della libreria ***pandas***, si sono visti i tipi di valore di feature e target per poi ottenere le colonne numeriche e quelle categoriche.

Distribuzione delle feature numeriche Sulle feature numeriche è stata verificata la distribuzione dei valori. Tutte risultano distribuite più o meno sulla stessa scala ad eccezione della feature `Rainfall_mm` che presenta valori molto più grandi degli altri come si può vedere nella Figura 2.

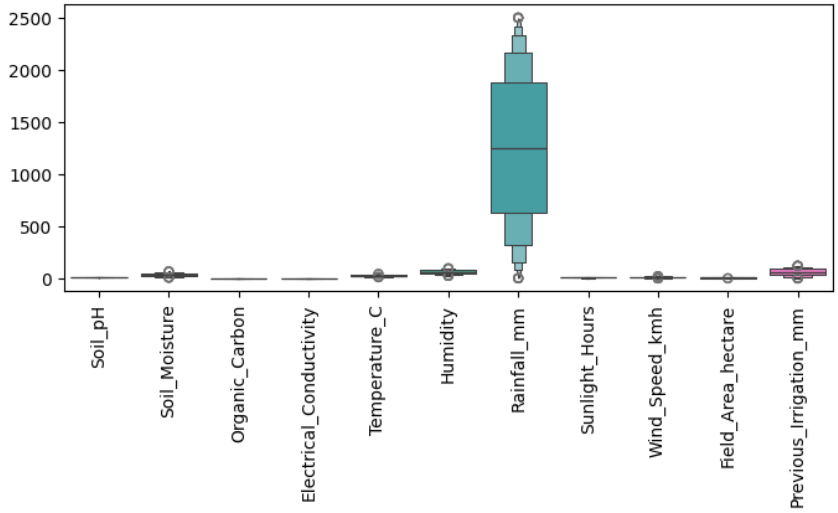


Figura 2: Distribuzione dei valori delle feature numeriche

Codifica della feature categoriche La variabile target `Irrigation_Need` è una variabile categorica ordinale ($Low \rightarrow Medium \rightarrow High$). Essendo presente un ordine semantico nei valori, viene applicata una codifica manuale che preserva tale ordine ($0 \rightarrow 1 \rightarrow 2$). In questo modo il modello può interpretare correttamente la progressione del fabbisogno di irrigazione. Inoltre, la variabile `Mulching_Used` può assumere come valori Yes/No e per questo viene resa booleana (1/0). Infine, le restanti variabili categoriche sono state trasformate in numeriche tramite ***LabelEncoder*** di ***sklearn.preprocessing***.

Grazie a queste trasformazioni è stato possibile calcolare la matrice di correlazione che mostra le relazioni tra le feature e rispetto alla variabile target. Dalla Figura 3 si evince che, apparentemente, non ci sono dipendenze importanti.

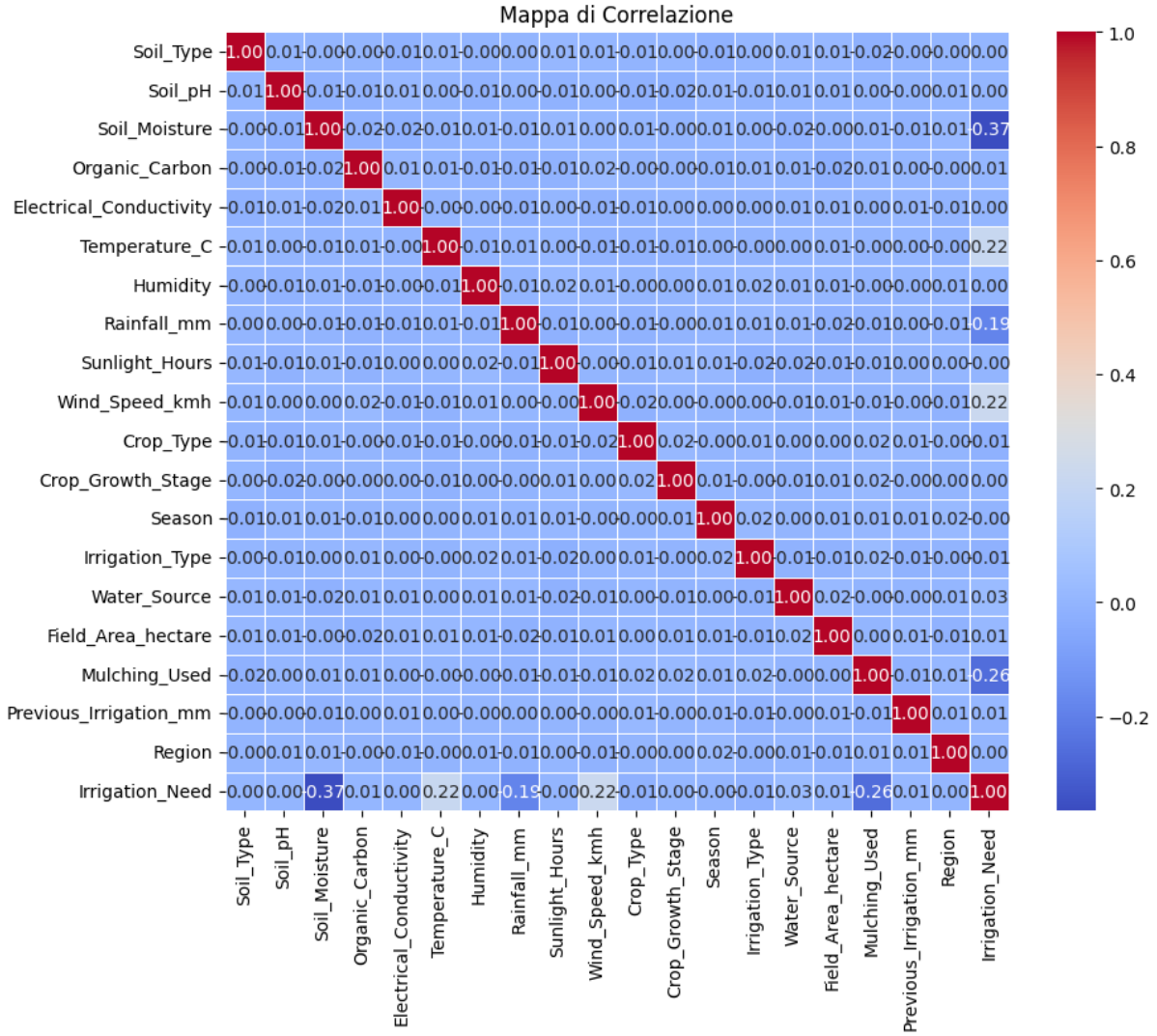


Figura 3: Matrice di correlazione

L'output di questa fase è un nuovo dataset (`optimized_dataset.csv`) ottenuto dalla applicazione delle operazioni.

5 Apprendimento Supervisionato

In questa fase vengono adottate tecniche di apprendimento automatico supervisionato con l'obiettivo di predire la variabile target `Irrigation_Need`. Vengono addestrati e confrontati tre classificatori, appartenenti a differenti paradigmi di apprendimento.
Il primo modello considerato è un **Decision Tree**, un weak learner (modello debole). Poi, al fine di migliorare le prestazioni predittive e la capacità di generalizzazione, vengono adottate due tecniche di ***Ensemble Learning***, che combinano più modelli deboli per ottenere un classificatore più robusto: **Random Forest** (tecnica di Bagging) e **AdaBoost** (tecnica di Boosting).

5.1 Strumenti utilizzati

Per implementare l'albero di decisione è stata usata la classe **DecisionTreeClassifier** [4] della libreria *sklearn.tree*.
Invece, per Random Forest e AdaBoost le classi **RandomForestClassifier** e **AdaBoostClassifier** di *sklearn.ensemble* [5].

5.2 Decisioni di progetto

Ai fini dell'addestramento e successiva valutazione, per ciascun modello è stata definita una griglia di iperparametri rilevanti da ottimizzare.
Nel caso del **Decision Tree**, sono stati ottimizzati:

- `max_depth`: la profondità massima dell'albero.
  I valori valutati sono 10 e 15;
- `min_samples_split`: il numero minimo di esempi necessari per la suddivisione.
  I valori valutati sono 2, 5, 10.

Per la **Random Forest**:

- `n_estimators`: il numero di alberi di decisione che vengono generati.
  I valori considerati sono 50, 100, 150;
- `max_depth`: la profondità massima dell'albero.
  I valori valutati sono None (nessun limite), 10 e 20.

Infine, per **AdaBoost**:

- `n_estimators`: il numero di alberi di decisione che vengono generati.
  I valori valutati sono 50, 100, 200;
- `learning_rate`: regola il contributo di ciascun modello nella combinazione finale e consente di bilanciare velocità di apprendimento e robustezza.
  I valori valutati sono 0.01, 0.1 e 1.0.

Vista la presenza della feature con valori su una scala diversa da quelle degli altri valori (sezione 4), saranno normalizzati i valori delle feature numeriche eslusivamente per gli esempi del training set per evitare data leakage (uso di informazioni durante l'addestramento che non sarebbero disponibili al momento della previsione). La normalizzazione avviene mediante normalizzazione **MinMax** che porta i valori nell'intervallo [0,1] usando ***MinMaxScaler*** della libreria ***sklearn.preprocessing***.

5.3 Valutazione

L'addestramento e la valutazione sono stati effettuati mediante una strategia di **cross-validation annidata**, in cui una cross-validation **esterna** stratificata e ripetuta [6] suddivide il dataset in più partizioni di training e test, mentre una cross-validation **interna** viene utilizzata per l'ottimizzazione degli iperparametri tramite **Grid Search Cross-Validation** [7].
Nella **CV esterna**, il dataset viene diviso in training e test set (quello che sarà usato per la valutazione finale). Si tratta di una CV a 5 fold che viene ripetuta 5 volte per un totale di 25 modelli che vengono addestrati e valutati su altrettanti test-set
Nella **CV interna** ogni training set ottenuto sopra viene suddiviso ulteriormente in 3 fold.
Quindi, in ciascuna iterazione, il modello ottimizzato viene addestrato sul training set e successivamente applicato al test set esterno, riducendo l'influenza di fattori casuali nella stima delle predizioni.

Per la valutazione, si usa come metrica la **F1-score** ossia la media armonica di **precisione** e **richiamo** che vengono anche calcolati. La metrica è calcolata per ogni test set e alla fine si restituiscono la media e la deviazione standard. Sulla base della misura F1 saranno confrontati i tre diversi classificatori. Non si calcola l'accuratezza perchè sconveniente su dataset sbilanciati.

I risultati ottenuti sono i seguenti.
Per **Decision Tree** la coppia migliore di iperparametri si è rivelata essere `max_depth` = 10 e `min_samples_split` = 2 (in 10 modelli su 25). La F1-score media è stata di 0.99 con una dev.std. pari a 0.0015.
Invece, per **Random Forest** la coppia migliore di iperparametri è stata `n_estimators` = 100 e `max_depth` = None (in 10 modelli su 25). La F1-score media è stata di 0.98 con una dev.std. pari a 0.0034.
Infine, per **AdaBoost** la coppia migliore di iperparametri è stata `n_estimators` = 50 e `learning_rate` = 1.0 (in 21 modelli su 25). La F1-score media è stata di 0.86 con una dev.std. pari a 0.04.

Si riassumono in una tabella i risultati (media ± dev.std) includendo anche la precisione e richiamo da cui deriva la misura F1.

	DecisionTree	**RandomForest**	**AdaBoost**
F1	**0.99** ± 0.0015	0.98 ± 0.0034	0.86 ± 0.04
P	0.99 ± 0.0015	0.98 ± 0.0028	0.90 ± 0.031
R	0.99 ± 0.0015	0.98 ± 0.0029	0.87 ± 0.042

Quindi, sulla base dei risultati si può affermare che i modelli con le prestazioni migliori sono stati Decision Tree e Random Forest mentre l'AdaBoost è stato quello che ha dato i risultati peggiori con misure di F1 molto diverse tra i diversi modelli come si può vedere nella Figura 4. In generale, i risultati sono da ritenersi soddisfacenti nonostante la presenza della classe minoritaria. Statisticamente, quasi a parità di F1, il **Decision Tree** risulta migliore del Random Forest anche per la stabilità (si veda la dev.std).

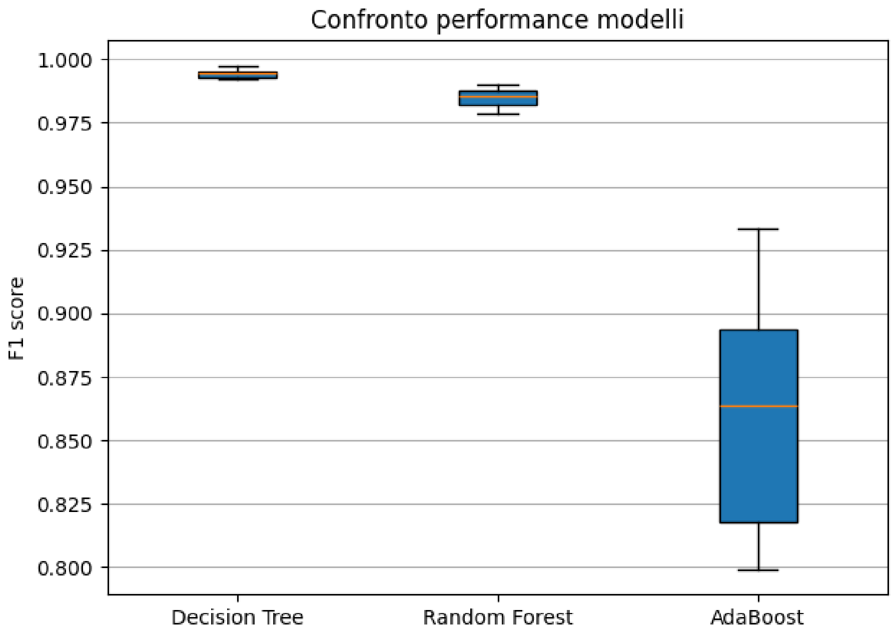
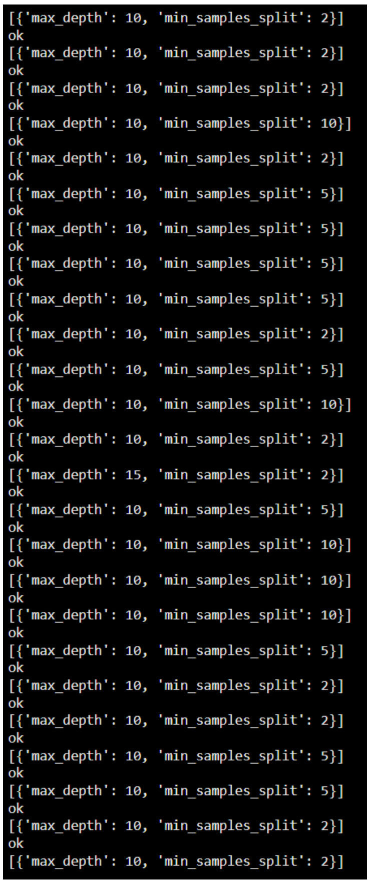


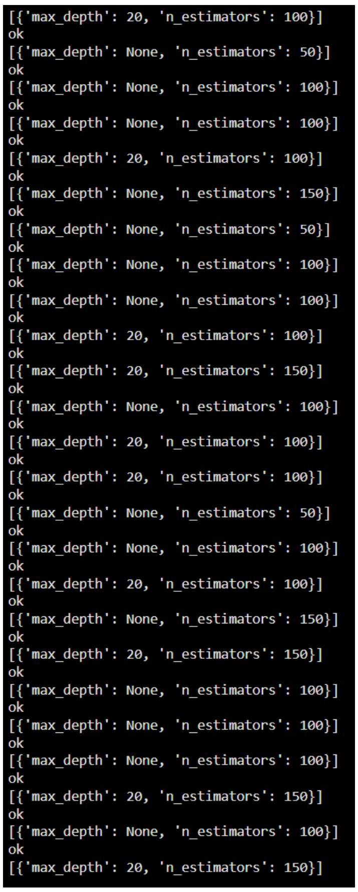
Figura 4: Distribuzione delle misure F1 dei modelli appresi

Nelle Figure 5,6,7 si possono vedere i migliori iperparametri ottenuti per ogni modello addestrato durante la CV.



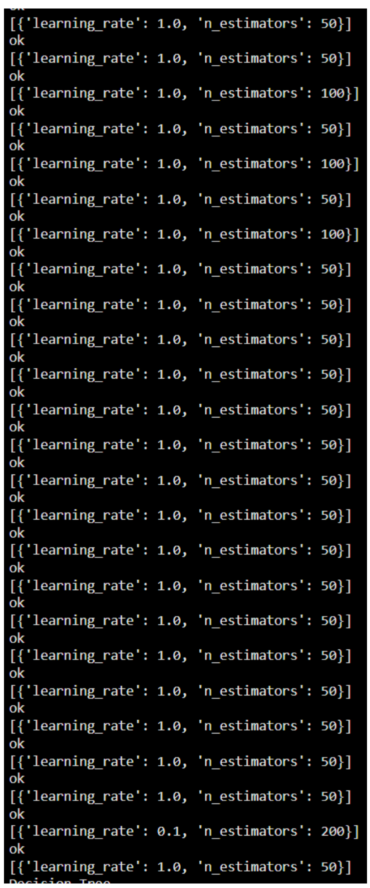
```
[{'max_depth': 10, 'min_samples_split': 2}]
ok
[{'max_depth': 10, 'min_samples_split': 2}]
ok
[{'max_depth': 10, 'min_samples_split': 2}]
ok
[{'max_depth': 10, 'min_samples_split': 10}]
ok
[{'max_depth': 10, 'min_samples_split': 2}]
ok
[{'max_depth': 10, 'min_samples_split': 5}]
ok
[{'max_depth': 10, 'min_samples_split': 5}]
ok
[{'max_depth': 10, 'min_samples_split': 5}]
ok
[{'max_depth': 10, 'min_samples_split': 5}]
ok
[{'max_depth': 10, 'min_samples_split': 2}]
ok
[{'max_depth': 10, 'min_samples_split': 5}]
ok
[{'max_depth': 10, 'min_samples_split': 10}]
ok
[{'max_depth': 10, 'min_samples_split': 2}]
ok
[{'max_depth': 15, 'min_samples_split': 2}]
ok
[{'max_depth': 10, 'min_samples_split': 5}]
ok
[{'max_depth': 10, 'min_samples_split': 10}]
ok
[{'max_depth': 10, 'min_samples_split': 10}]
ok
[{'max_depth': 10, 'min_samples_split': 10}]
ok
[{'max_depth': 10, 'min_samples_split': 5}]
ok
[{'max_depth': 10, 'min_samples_split': 2}]
ok
[{'max_depth': 10, 'min_samples_split': 2}]
ok
[{'max_depth': 10, 'min_samples_split': 5}]
ok
[{'max_depth': 10, 'min_samples_split': 5}]
ok
[{'max_depth': 10, 'min_samples_split': 2}]
ok
[{'max_depth': 10, 'min_samples_split': 2}]
```

Figura 5: DT



```
[{'max_depth': 20, 'n_estimators': 100}]
ok
[{'max_depth': None, 'n_estimators': 50}]
ok
[{'max_depth': None, 'n_estimators': 100}]
ok
[{'max_depth': None, 'n_estimators': 100}]
ok
[{'max_depth': 20, 'n_estimators': 100}]
ok
[{'max_depth': None, 'n_estimators': 150}]
ok
[{'max_depth': None, 'n_estimators': 50}]
ok
[{'max_depth': None, 'n_estimators': 100}]
ok
[{'max_depth': None, 'n_estimators': 100}]
ok
[{'max_depth': 20, 'n_estimators': 100}]
ok
[{'max_depth': 20, 'n_estimators': 150}]
ok
[{'max_depth': None, 'n_estimators': 100}]
ok
[{'max_depth': 20, 'n_estimators': 100}]
ok
[{'max_depth': 20, 'n_estimators': 100}]
ok
[{'max_depth': None, 'n_estimators': 50}]
ok
[{'max_depth': None, 'n_estimators': 100}]
ok
[{'max_depth': 20, 'n_estimators': 100}]
ok
[{'max_depth': None, 'n_estimators': 150}]
ok
[{'max_depth': 20, 'n_estimators': 150}]
ok
[{'max_depth': None, 'n_estimators': 100}]
ok
[{'max_depth': None, 'n_estimators': 100}]
ok
[{'max_depth': None, 'n_estimators': 100}]
ok
[{'max_depth': 20, 'n_estimators': 150}]
ok
[{'max_depth': None, 'n_estimators': 100}]
ok
[{'max_depth': 20, 'n_estimators': 150}]
```

Figura 6: RF



```
[{'learning_rate': 1.0, 'n_estimators': 50}]
ok
[{'learning_rate': 1.0, 'n_estimators': 50}]
ok
[{'learning_rate': 1.0, 'n_estimators': 100}]
ok
[{'learning_rate': 1.0, 'n_estimators': 50}]
ok
[{'learning_rate': 1.0, 'n_estimators': 100}]
ok
[{'learning_rate': 1.0, 'n_estimators': 50}]
ok
[{'learning_rate': 1.0, 'n_estimators': 100}]
ok
[{'learning_rate': 1.0, 'n_estimators': 50}]
ok
[{'learning_rate': 1.0, 'n_estimators': 50}]
ok
[{'learning_rate': 1.0, 'n_estimators': 50}]
ok
[{'learning_rate': 1.0, 'n_estimators': 50}]
ok
[{'learning_rate': 1.0, 'n_estimators': 50}]
ok
[{'learning_rate': 1.0, 'n_estimators': 50}]
ok
[{'learning_rate': 1.0, 'n_estimators': 50}]
ok
[{'learning_rate': 1.0, 'n_estimators': 50}]
ok
[{'learning_rate': 1.0, 'n_estimators': 50}]
ok
[{'learning_rate': 1.0, 'n_estimators': 50}]
ok
[{'learning_rate': 1.0, 'n_estimators': 50}]
ok
[{'learning_rate': 1.0, 'n_estimators': 50}]
ok
[{'learning_rate': 1.0, 'n_estimators': 50}]
ok
[{'learning_rate': 1.0, 'n_estimators': 50}]
ok
[{'learning_rate': 1.0, 'n_estimators': 50}]
ok
[{'learning_rate': 1.0, 'n_estimators': 50}]
ok
[{'learning_rate': 0.1, 'n_estimators': 200}]
ok
[{'learning_rate': 1.0, 'n_estimators': 50}]
```

Figura 7: AB

6 Rappresentazione della Conoscenza

In questa fase si costruisce una Belief Network (o Rete Bayesiana) per la rappresentazione della conoscenza con l'obiettivo di comprendere le dipendenze tra le feature. Si confrontano le strutture ottenute usando due diverse metriche di scoring.

6.1 Strumenti utilizzati

Per l'apprendimento della struttura della rete bayesiana si usa la classe **HillClimbSearch** della libreria *pgmpy.estimators* [8].
Invece, per l'apprendimento dei parametri **DiscreteBayesianNetwork** di *pgmpy.model*.

6.2 Operazioni preliminari

Il dataset è stato sottoposto a una fase di **discretizzazione** delle feature numeriche, necessaria per l'utilizzo di modelli probabilistici con variabili categoriche. Le variabili continue sono state trasformate in intervalli (bin) usando il metodo **cut** di *pandas*.
Dove possibile, sono state definite soglie coerenti con il significato semantico delle grandezze rappresentate. Ad esempio, la feature numerica relativa al pH del terreno (`Soil_pH`) è stata discretizzata nelle categorie ACIDO, NEUTRO e BASICO, rispettando la reale scala del pH.
Negli altri casi, dopo aver trovato i valori min e max, è stato definito il numero di intervalli in cui discretizzare e spesso anche le etichette con cui sostituire i valori.

6.3 Decisioni di progetto

Per la fase di apprendimento della struttura sono state utilizzate due diverse metriche di scoring: **BIC** (Bayesian Information Criterion) che tende a preferire modelli più semplici e **BDeu** (Bayesian Dirichlet equivalent uniform). Quest'ultima si basa sull'uso di una distribuzione di Dirichlet, ideale in presenza di variabili target categoriche, come quelle presenti nel dataset.

6.4 Valutazione

Confrontando le due strutture ottenute rispetto al numero di archi, quella appresa con BIC presenta un numero maggiore di archi (11) rispetto a quella ottenuta con BDeu (7). Tali differenze riflettono il diverso compromesso tra complessità del modello e capacità di adattamento ai dati imposto dalle due metriche.
Successivamente sono stati appresi i parametri di entrambe le reti, cioè le tabelle di probabilità condizionate (CPT).
I grafi (DAG) risultanti includono solo un sottoinsieme delle variabili originali del dataset, in particolare: `Mulching_Used`, `Soil_Moisture`, `Temperature_C`, `Crop_Growt_Stage`, `Rainfall_mm`, `Wind_Speed_kmh` e `Irrigation_Need`. Ciò evidenzia come alcune variabili risultano maggiormente rilevanti nel modellare la variabile target.
Si riporta il grafo più semplice ottenuto.

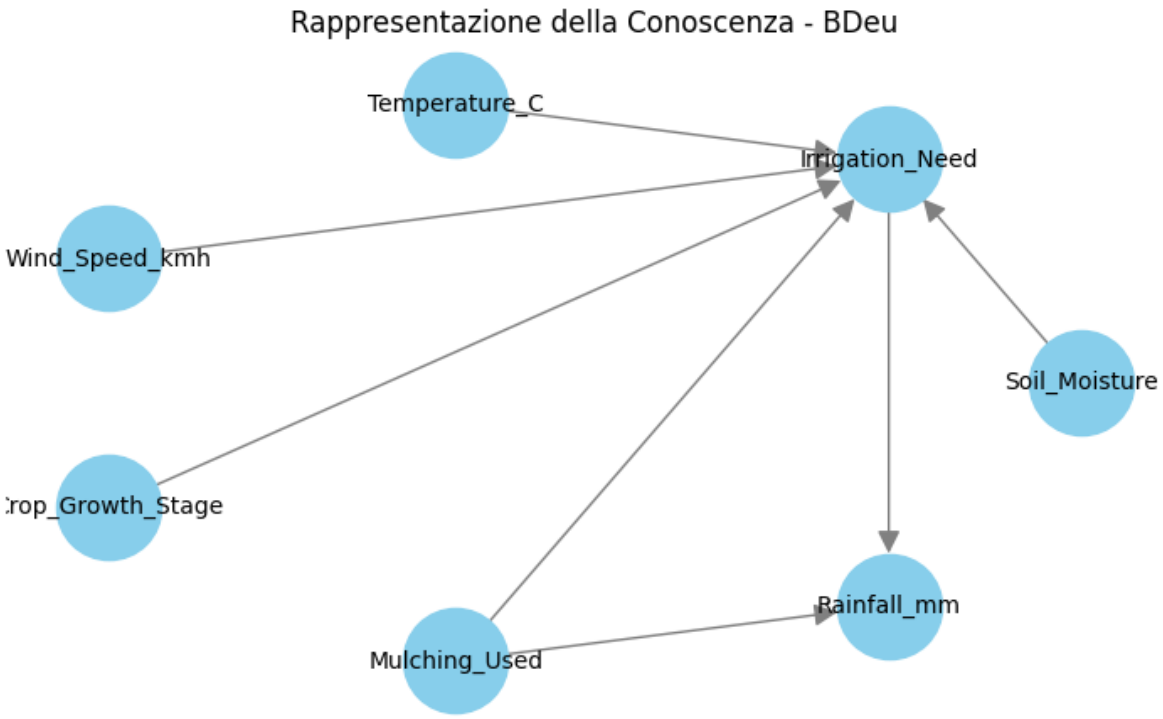


Figura 8: Grafo ottenuto usando la BDeu come metrica di scoring

È stata quindi calcolata la **Markov Blanket** della variabile target, l'insieme minimale di nodi che rendono la variabile condizionatamente indipendente dalle altre. L'insieme di nodi ottenuto coincide con quello presente nel modello appreso, confermando la coerenza strutturale della rete e il ruolo delle variabili selezionate.

Infine, le variabili appartenenti alla Markov Blanket sono state confrontate con le feature più importanti individuate dai modelli di apprendimento supervisionato analizzati nella sezione precedente (vedi Figura 9). Il confronto mostra una corrispondenza tra i due approcci: le stesse variabili risultano infatti rilevanti sia secondo i classificatori sia secondo la struttura della Rete Bayesiana.

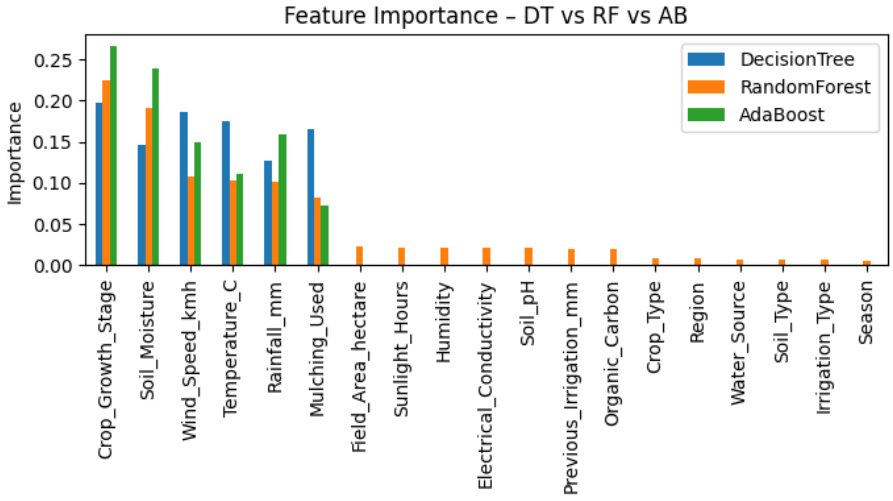


Figura 9: Feature importance per i classificatori appresi

7 Apprendimento Probabilistico

In questa ultima parte del caso di studio, si studiano le performance di modelli probabilistici nella predizione. In particolare, si confronta un approccio basato sulla BN costruita e appresa precedentemente con uno su Naive Bayes, che fa assunzioni di indipendenza condizionata tra le feature noto il valore della classe/variabile target.

7.1 Strumenti utilizzati

Oltre a quanto riportato nella sezione 6.1, per il modello Naive Bayes si usa la classe **CategoricalNB** [9] della libreria *sklearn.naive_bayes* che lavora con feature discrete.
Nel dataset precedentemente discretizzato, sono state comunque codificate le feature mediante ***LabelEncoder*** di ***sklearn.preprocessing***. Lo stesso è stato fatto per la variabile target ma con un mapping manuale.

7.2 Decisioni di progetto

Per l'inferenza probabilistica con la BN si adotta un approccio di inferenza esatta, nello specifico l'algoritmo di eliminazione delle variabili [10], implementato attraverso la classe **VariableElimination** di *pgmpy.inference*.
Nel classificatore Naive Bayes si imposta il parametro **alpha** a 1.0 corrispondente allo smoothing di Laplace per evitare probabilità nulle. Anche se nel dataset non si osservano configurazioni sparse, la presenza della classe minoritaria e la suddivisione in training e test set potrebbero generare combinazioni non osservate in fase di addestramento. L'introduzione dello smoothing rende pertanto il modello più robusto e ne favorisce l'applicabilità anche in sviluppi futuri con dati più sparsi.

7.3 Valutazione

La metrica di valutazione adottata per il confronto è la misura F1, usata anche negli esperimenti precedenti.

Per la predizione mediante BN, per ciascun esempio del test set (di dimensioni pari al 20% del dataset), viene costruita un'evidenza composta esclusivamente dalle variabili appartenenti al Markov Blanket della variabile target, ossia quelle la cui informazione è rilevante per la predizione.
Per ogni esempio del test set, la classe target viene stimata tramite una query MAP (Maximum A Posteriori), che individua il valore della variabile target con massima probabilità a posteriori condizionata all'evidenza osservata, tramite il metodo **map_query** della classe *VariableElimination*.

La valutazione è stata condotta su cinque diversi test set, ottenuti manualmente attraverso cinque esecuzioni indipendenti (senza fissare il seme di randomizzazione per la riproducibilità). La F1-score media ottenuta è pari all'87%.
Nella tabella seguente sono riportate nel dettaglio le misure ottenute nelle singole esecuzioni.

Run	F1	P	R
1	0.867	0.869	0.866
2	0.869	0.872	0.873
3	0.880	0.882	0.884
4	0.884	0.888	0.888
5	0.872	0.874	0.876
media	**0.87**	0.88	0.87
dev.std	0.007	0.007	0.008

Anche la valutazione del classificatore Naive Bayes è avvenuta nelle stesse condizioni riportando una f1-score media dell'84% .
Di seguito la tabella con i risultati.

Run	F1	P	R
1	0.83	0.84	0.83
2	0.84	0.85	0.85
3	0.84	0.86	0.86
4	0.84	0.85	0.85
5	0.83	0.84	0.83
media	**0.84**	0.85	0.84
dev.std	0.005	0.007	0.01

Nel confronto tra i due approcci emerge una differenza prestazionale chiara a favore della Bayesian Network (BN) rispetto al Naive Bayes (NB), anche se non non estremamente ampia. La BN raggiunge infatti una F1-score media dell'87%, contro l'84% del Naive Bayes, evidenziando un miglior equilibrio tra precisione e richiamo.
Tale differenza può essere attribuita alla diversa capacità dei due modelli di rappresentare le dipendenze tra le variabili: mentre il Naive Bayes assume indipendenza condizionata tra le feature dato il target, la Bayesian Network modella esplicitamente le relazioni di dipendenza attraverso la struttura del grafo orientato aciclico (DAG) appreso.

Di conseguenza, la BN è in grado di catturare eventuali dipendenze tra le variabili che il Naive Bayes, per costruzione, non può considerare. Questo aspetto risulta particolarmente rilevante in domini in cui le feature non sono indipendenti, contribuendo a spiegare il miglioramento osservato nelle prestazioni.

8 Conclusioni

Nel caso di studio, per la **rappresentazione** della conoscenza si è fatto riferimento alle Bayesian Network (BN), che permettono di identificare sia le variabili che influenzano direttamente la variabile target sia le relazioni di dipendenza tra le variabili del dominio. Le strutture delle BN sono state apprese e valutate utilizzando due differenti metriche, BIC e BDeu. In particolare, la metrica BDeu ha prodotto una struttura più semplice in termini di numero di archi, risultando più compatta e interpretabile.

Per quanto riguarda l'**apprendimento**, sono state impiegate tecniche di apprendimento automatico supervisionato, addestrando diversi modelli basati su tre classificatori: Decision Tree, Random Forest e AdaBoost. Sulla base della conoscenza a disposizione e delle metriche di valutazione considerate, il modello che ha mostrato le prestazioni migliori è risultato essere il Decision Tree, con una F1-score media pari al 99% e una deviazione standard di 0.0015, evidenziando un'elevata stabilità. Prestazioni comparabili, seppur leggermente inferiori, sono state ottenute dal Random Forest, con una F1-score media del 98%. I valori ottenuti suggeriscono una buona capacità di generalizzazione su dati non visti. L'uso della cross-validation ha consentito di mitigare il rischio di overfitting, fornendo una stima delle prestazioni più robusta rispetto ad una singola suddivisione in training e test set. Infatti, cambiando i ruoli degli esempi, il modello mostra stabilità.

Oltre ai modelli di Machine Learning tradizionali, sono state adottate anche tecniche di apprendimento probabilistico. In particolare, la rete bayesiana appresa è stata utilizzata come classificatore, ottenendo una F1-score media dell'87% su più esecuzioni, valore in linea con quello raggiunto dal classificatore AdaBoost. Inoltre, è stato implementato il classificatore Naive Bayes, che ha mostrato prestazioni inferiori rispetto agli altri approcci, con una F1-score media dell'84%, risultando complessivamente il meno performante tra gli esperimenti condotti.

Nel complesso, i risultati evidenziano come i modelli di Machine Learning supervisionato garantiscano le migliori prestazioni predittive nel contesto considerato, mentre le Bayesian Network si rivelano particolarmente efficaci come strumento di rappresentazione e interpretazione della conoscenza, offrendo un compromesso tra capacità esplicativa e accuratezza nella classificazione.

9 Sviluppi futuri

Possibili estensioni di questo progetto potrebbero riguardare l'integrazione di conoscenza proveniente da ontologie. Inoltre, a partire dai dati disponibili, si potrebbero definire problemi di soddisfacimento dei vincoli (CSP) per la pianificazione intelligente dell'irrigazione con l'obiettivo di ottimizzare il consumo idrico. Tali problemi potrebbero essere risolti mediante algoritmi di ricerca euristica, come A*.

10 Materiale

I dataset usati nel caso di studio, copia di questa documentazione e il codice sorgente sono disponibili al seguente link: GitHub. Il notebook Colab contenente il codice è disponibile anche al seguente link: NotebookCondiviso.

Riferimenti bibliografici

[1] D. L. Poole e A. K. Mackworth, *Artificial Intelligence*. Cambridge University Press, 2023, cap. 7. indirizzo: https://artint.info/3e/html/ArtInt3e.Ch7.html.

[2] D. L. Poole e A. K. Mackworth, *Artificial Intelligence*. Cambridge University Press, 2023, cap. 9. indirizzo: https://artint.info/3e/html/ArtInt3e.Ch9.html.

[3] D. L. Poole e A. K. Mackworth, *Artificial Intelligence*. Cambridge University Press, 2023, cap. 10. indirizzo: https://artint.info/3e/html/ArtInt3e.Ch10.html.

[4] "DecisionTreeClassifier. "indirizzo: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

[5] "RandomForestClassifier + AdaBoostClassifier. "indirizzo: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.html.

[6] "Repeated stratified K-Fold cross-validation. "indirizzo: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RepeatedStratifiedKFold.html.

[7] "GridSearch cross-validation. "indirizzo: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

[8] "HillClimbSearch. "indirizzo: https://pgmpy.org/structure_estimator/hill.html.

[9] "CategoricalNB. "indirizzo: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.CategoricalNB.html.

[10] D. L. Poole e A. K. Mackworth, *Artificial Intelligence*. Cambridge University Press, 2023, cap. 9.5.1. indirizzo: https://artint.info/3e/html/ArtInt3e.Ch9.S5.html.