UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO | DIPARTIMENTO DI INFORMATICA

Documentazione Caso di Studio Ingegneria della Conoscenza A.A. 2024/25

Gestione delle risorse idriche

Gruppo di lavoro

- Lorusso Michele, 776582, m.lorusso229@studenti.uniba.it

Repository Github

- Progetto

Sommario

Introduzione

Il software per la gestione delle risorse idriche mira a migliorare l'efficienza del sistema idrico attraverso l'utilizzo di tecnologie avanzate di intelligenza artificiale e analisi dei dati. L'obiettivo principale del software è quello di migliorare l’efficienza e la sostenibilità del sistema idrico, ottimizzando l’uso delle risorse, riducendo gli sprechi, la carenza idrica in modo da garantire un approvvigionamento di acqua in modo costante e sicuro.

Knowledge Base

Una Knowledge Base[1] è un insieme di conoscenze(regole e fatti) che possono essere utilizzate da programmi per prendere decisioni o rispondere a domande. La KB implementata riguarda un modello per la gestione delle risorse idriche in situazione di carenza idrica utilizzando la programmazione logica in Prolog[2] .

Fatti e predicati della KB

I predicati e i fatti utilizzati creati per la KB sono le seguenti(per generare clausole in Prolog sono stati utilizzati i dati di un ontologia in formato csv (risorse_idriche.csv[3]):

Il fatto **acqua_annuale(Nazione,AcquaAnnuale)** memorizza la quantità annuale d'acqua di una nazione

Il fatto **acqua_pro_capite(Nazione,AcquaProCapite)** memorizza l'acqua disponibile per ogni cittadino

Il fatto **popolazione(Nazione,Popolazione)** indica la popolazione di una nazione

---

```
carica_risorse:-
    csv_read_file('risorse_idriche.csv', Rows, [functor(row)]),
    forall(
        member(row(Nazione, AcquaAnnuale, AcquaProCapite, Popolazione), Rows),
        (   valid_data(Nazione, AcquaAnnuale, AcquaProCapite, Popolazione) ->
            assertz(acqua_annuale(Nazione, AcquaAnnuale)),
            assertz(acqua_pro_capite(Nazione, AcquaProCapite)),
            assertz(popolazione(Nazione, Popolazione))
        ;   write('Dati non validi per: '), write(Nazione), nl
        )
    ).
```

Il predicato **carica_risorse : - …**, ha lo scopo di creare i fatti prendendo in input un file csv e verificando la correttezza dell'intestazione prima di creare i fatti in modo dinamico

---

```
valid_data(_, AcquaAnnuale, AcquaProCapite, Popolazione):-
    number(AcquaAnnuale),
    number(AcquaProCapite),
    number(Popolazione).
```

Il predicato **valid_data(_, AcquaAnnuale, AcquaProCapite, Popolazione) :- number(AcquaAnnuale), number(AcquaProCapite), number(Popolazione).** , controlla che le righe con intestazioni AcquaAnnuale, AcquaProCapite e Popolazione sono dei numeri

---

```
acqua_totale_utilizzata(Nazione, AcquaTotale):-
    calcola_acqua(Nazione, AcquaTotale), !. % Usa il valore memorizzato
acqua_totale_utilizzata(Nazione, AcquaTotale):-
    acqua_annuale(Nazione, AcquaAnnuale),
    acqua_pro_capite(Nazione, AcquaProCapite),
    popolazione(Nazione, Popolazione),
    AcquaTotale is AcquaAnnuale + (AcquaProCapite * Popolazione),
    assertz(calcola_acqua(Nazione, AcquaTotale)). % Memorizza il risultato
```

Il predicato **acqua_totale_utilizzata (Nazione,AcquaTotale): - …** , calcola la quantità di acqua utilizzata da una nazione . Se il valore per una nazione è stato già calcolato l'algoritmo lo utilizza immediatamente, altrimenti lo calcola e lo memorizza in **calcola_acqua/2** per ottimizzare le successive richieste

---

```
previsione_domanda_futura(Nazione, Fattore, DomandaFutura):-
    calcola_acqua(Nazione, Fattore, DomandaFutura).
```

Il predicato **previsione_domanda_futura :- …**, prende in input una nazione e un fattore di rischio legato alla carenza idrica e calcola la domanda futura, cioè la quantità di acqua che una nazione userà effettivamente, richiamando calcola_acqua

---

```
calcola_acqua(Nazione, Fattore, Risultato):-
    acqua_totale_utilizzata(Nazione, AcquaTotale),
    Risultato is AcquaTotale * Fattore.
```

Il predicato **calcola_acqua:-**…, prende in input una nazione e un fattore di rischio e viene usato per stimare la domanda di risorse idriche e inserisce il risultato nella variabile Risultato.

```
puo_soddisfare_futura(Nazione, Fattore):-
    previsione_domanda_futura(Nazione, Fattore, DomandaFutura),
    acqua_totale_utilizzata(Nazione, AcquaTotale),
    AcquaTotale >= DomandaFutura.
```

Il predicato **puo_soddisfare_futura : - ….**, prende in input una nazione e un fattore di rischio e verifica se la quantità d'acqua attualmente disponibile di una nazione può soddisfare la domanda futura di acqua.

```
crisi_idrica(Nazione, SogliaCrisi):-
    acqua_totale_utilizzata(Nazione, AcquaTotale),
    AcquaTotale =< SogliaCrisi,
    write('Crisi idrica in '), write(Nazione),
    write('. Iniziare azioni di emergenza!'), nl.
```

Il predicato **crisi_idrica : - ….**, data una nazione verifica se l'acqua totale di una nazione supera una soglia critica, definita dalla variabile SogliaCrisi, che descrive se una nazione è in crisi idrica

---

```
raccomandazioni(Nazione, SogliaCrisi):-
    (   crisi_idrica(Nazione, SogliaCrisi) ->
        write('Suggerimenti per '), write(Nazione), write(': '), nl,
        write('- Ridurre gli sprechi.'), nl,
        write('- Investire in infrastrutture per la conservazione dell\'acqua.'), nl,
        write('- Promuovere l\'uso sostenibile delle risorse.'), nl
    ;   write('La gestione delle risorse di '), write(Nazione),
        write(' Ã¨ soddisfacente al momento.'),nl).
```

Il predicato **raccomandazioni : - ….**, data una nazione in input segnala se una nazione è in crisi idrica mostrando messaggi per pianificare un consumo delle risorse idriche più sostenibile

Apprendimento non Supervisionato

L'obiettivo dell'apprendimento non supervisionato[4] è quello di raggruppare elementi del dataset con caratteristiche simili in cluster omogenei. Quindi per il sistema progettato, è stato utilizzato per raggruppare le nazioni in base alla situazione idrica simile.

Data cleaning

Il dataset iniziale risorse_idriche.csv contiene i record relativi al consumo di risorse idriche ed è organizzato in questo modo:

	Nazione	AcquaAnnuale	AcquaProCapite	Popolazione
0	Afghanistan	20.280.000.000	2.843	19.542.982
1	Albania	1.311.000.000	1.196	3.003.387
2	Algeria	9.978.000.000	678.000	40.339.329
3	Angola	705.800.000	99.000	19.450.959
4	Antigua and Barbuda	11.500.000	359.000	87.674

Descrive il consumo acqua annuale, il consumo di acqua per abitante e la popolazione di ogni nazione.

Come primo passo abbiamo eliminato duplicati della colonna Nazione prendendo solo in considerazione valori unici. Inoltre si è creata una copia del dataset originale

utilizzabile ai fini dei due tipi di apprendimento di cui abbiamo apportato alcune modifiche: rimozione delle colonne testuali non utili ai fini dei modelli di apprendimento, rimozione dei valori mancanti e in ultimo non si tiene conto della colonna Stato_idrico (feature target di cui parlermo nell'apprendimento supervisionato). Inoltre vengono convertire le colonne prima in stringa eliminando qualsiasi carattere speciale e in seguito convertiti in intero, e infine normalizza tutte le colonne utilizzando lo scaling e in particolare il MinMaxScaler[5], cioè trasforma i valori in un intervallo fisso, e in questo caso è [0,1].

Scelta modello

Sommario

Per effettuare il clustering è stato utilizzato il modello KMeans[4]. È un algoritmo di hard clustering[4] che permette di associare ogni punto dati in un cluster specifico rispetto al centroide di quel cluster.

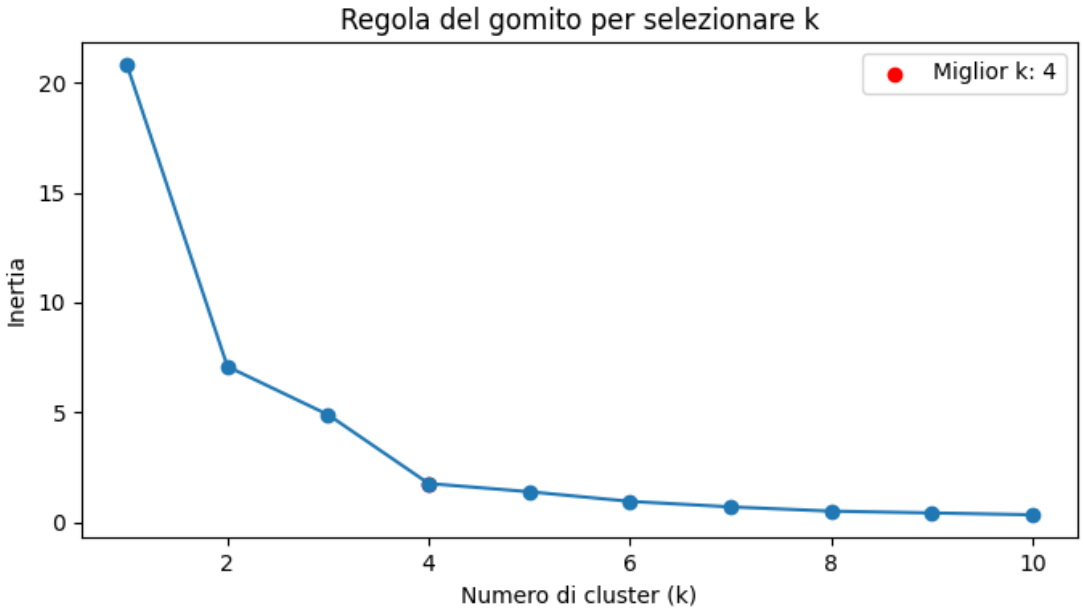
Strumenti utilizzati

Sono state utilizzante le librerie sklearn per il clustering, matplotlib e la numpy per la visualizzazione dei grafici, la libreria kneed per la regola del gomito e la libreria imblearn.over_sampling per l'oversampling

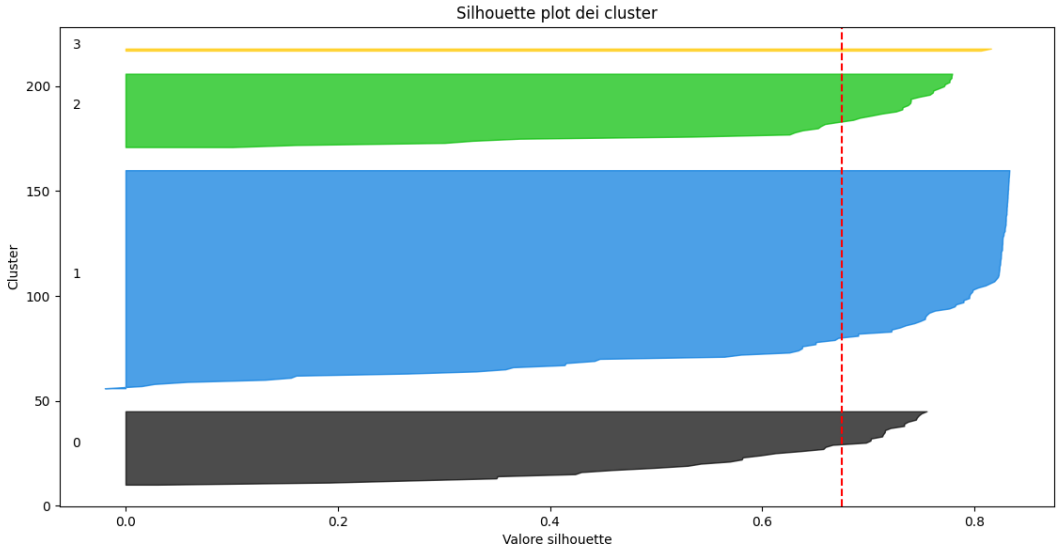
Decisioni di progetto

Il problema dell'algoritmo di KMeans è quello di ricercare il numero ottimale di cluster da assegnare all'algoritmo, in modo da minimizzare la distanza tra i punti dati e i cluster assegnati . Per trovare questo numero abbiamo utilizzato la regola del gomito, che utilizza una metrica di valutazione, l'inerzia, che aiuta a selezionare il numero ottimale di cluster, e viene calcolata come la distanza al quadrato di punto dati e il suo centroide e poi sommando tutte i quadrati per ogni punto dati nel cluster. Quindi viene eseguito l'algoritmo di Kmeans su range di valori che rappresenta i numeri dei possibili cluster, calcola l'inerzia per ogni cluster e poi viene mostrata la curva con l'identificazione del punto dove la diminuzione dell'inerzia e meno significativa. In questo caso abbiamo utilizzato un range di valori compreso da [1,10], un inizializzazione dei centroidi in modo casuale, e fatto eseguire l'algoritmo di Kmeans un numero di volte indicato dal parametro n_init (in questo caso 5) per

ogni cluster, per ottenere il clustering migliore:



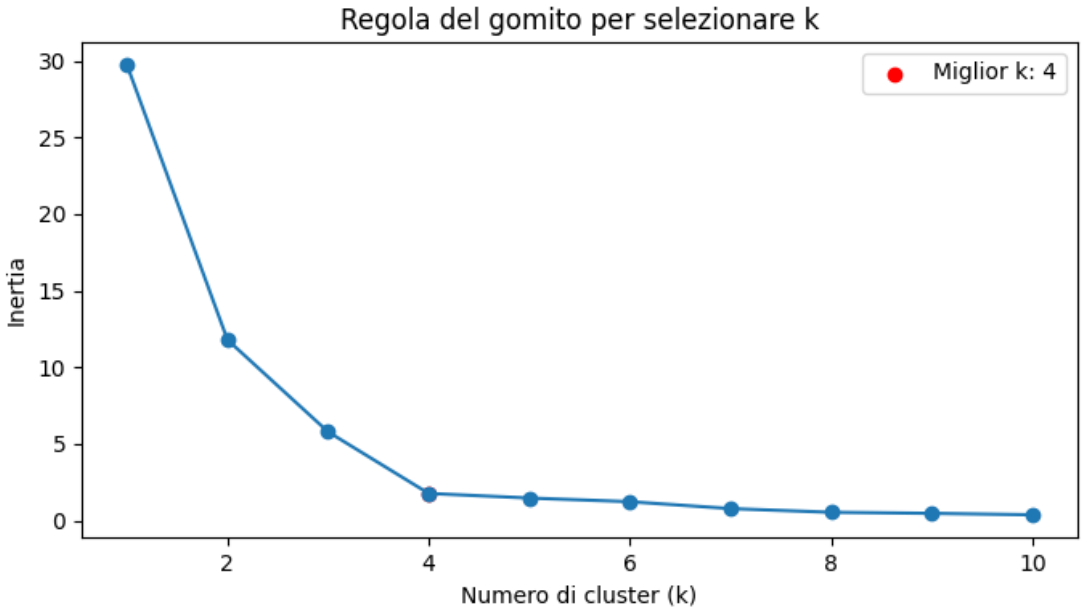
Il grafico evidenzia che il numero ottimale k di cluster è 4 ed è stato ri-eseguito in modo dinamico l'algoritmo di clustering con il numero k trovato. Il risultato del clustering è il seguente(che analizzaremo nella parte Valutazione):

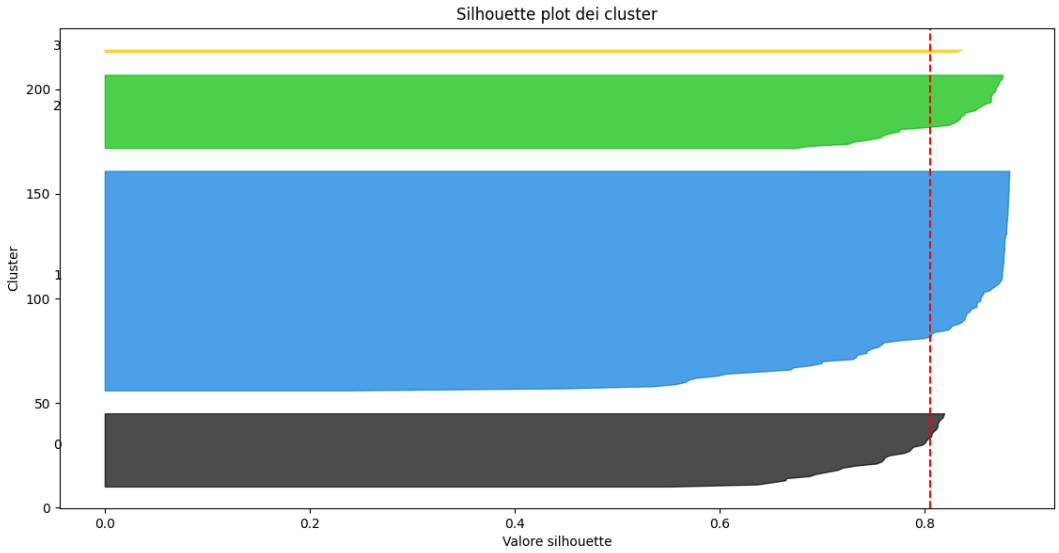


Questo grafico mostra la suddivisone in 4 cluster individuati in base all'obiettivo spiegato nella parte Sommario e inoltre questo grafico mostra anche una metrica che vedremo e descriveremo nella parte Valutazione.

Valutazione

In questa sezione analizziamo i cluster e in modo particolare la loro suddivisione, cioè se i punto dati risultano ben assegnati al proprio cluster di appartenenza rispetto ai cluster vicini, e per questo è stata utilizzata la metrica silhouette score[6]. Sono stati dati come parametri di input il dataset originale scalato e i cluster assegnati dal kMeans, e si calcola utilizzando due misure: coesione intra_cluster, cioè calcolando la distanza media tra punto e tutti gli altri punti del cluster, e la separazione inter_cluster , la distanza media tra il punto e i punti di un cluster più vicino. Il valore risultante, eseguito sui cluster trovati, è di 0.675 e indica una buona struttura del clustering anche se possono esserci dei punti in prossimità dei confini. Infatti come si può veder dal grafico sopra, nel cluster 1 sono presenti dei punti con valori negativi di silhouette score è questo indica una struttura non ben distinta. Per ovviare a questo problema è stato utilizzato l'oversampling[7] con tecnica SMOTE. Questa prende le osservazioni più vicine tra quelle dei cluster minoritari, e tramite combinazioni lineari genera dati sintetici. Questo ha permesso di migliorare la separazione dei cluster e ha aggiornato anche il grafico che mostra il numero ottimale di cluster da assegnare al KMeans come possiamo vedere dai seguenti grafici:





Possiamo notare dal primo grafico come il numero di cluster rimane invariato, e dal secondo grafico come il silhouette score sia migliorato (0.807) concludendo che i cluster risultano ben distinti tra di loro senza punti in prossimità dei confini di altri cluster.

Apprendimento supervisionato

Sommario

Per predire lo stato idrico di una nazione, cioè se è presente una crisi idrica e quindi scarsità d'acqua o non vi è ancora emergenza, in base alla disponibilità annuale e alla densità demografica sono stati utilizzati 4 modelli di classificazione(binaria), perché sono modelli che vengono addestrati, su dati etichettati ,in classi.

Il primo modello **LogisticRegression**[8] predice la probabilità di appartenenza a una classe.

Il secondo modello **DecisionTreeClassifier**[9], usa una serie di decisioni per classificare i dati.

Il terzo modello Random **ForestClassifier**[10], combina più alberi decisionali per migliorare la precisione.

Il quarto modello **K-NearestNeighbors**[11], classifica i dati in base alla vicinanza alle classi dei dati più vicini

Strumenti utilizzati

Sono state utilizzante le librerie sklearn per il clustering , matplotlib e la numpy per la visualizzazione dei grafici

Decisioni di progetto

Prima di impostare i modelli , abbiamo individuato la variabile target calcolata sulla variabile AcquaProCapite , prima calcolando la mediana[12] sui valori di AcquaProCapite e in seguito abbiamo inserito una condizione su quale valore deve assumere la variabile target Stato_idrico: 1(situazione di crisi_idrica) se il valore in AcquaProCapite è minore della mediana, 0 altrimenti.

In seguito sono stati selezionati gli iper-parametri dei modelli di apprendimento supervisionato[13] scelti. La loro impostazione è molto importate perché verranno utilizzati durante l'addestramento dei modelli e la loro scelta influisce sulla complessità e sulle prestazioni dei modelli. È stata utilizzata la tecnica della GridSearch con StratifiedKFold Validation. In questo approccio vengono definite le griglie dei possibili valori per ogni iper-parametro e si esplorano per ricercare la miglior combinazione possibile.

Per Il modello **LogisticRegression**:

- "*C*": specifica quanto è forte la regolarizzazione, un valore piccolo indica una forte regolarizzazione;
- "*penalty*": specifica una penalità applicata al modello, per ridurre la complessità e l'overfitting. I possibili valore che può assumere sono l1(norma 1) e l2(norma 2, utile per prevenire l'overfitting);
- "*solver*": utilizzato per ottimizzare il modello. Il valore di default è 'lbfgs' (si dimostra efficiente con penalty: l2)

Per il modello **DecisionTreeClassifier**:

- "*max-depth*": indica l'altezza dell'albero;
- "*min_samples_split*": il numero minimo di esempi necessari affinché possa essere inserito un criterio di split;
- "*min_samples_leaf*": il numero minimo di esempi per poter creare una foglia

Per il modello **RandomForestClassifier**:

- "*n_estimators*": numero di alberi nella foresta;
- "*max_depth*": indica l'altezza della foresta;
- "*min_samples_split*": il numero minimo di esempi necessari affinché possa essere inserito un criterio di split

Per il modello **K-NN**:

- "*n_neighbors*": indica il numero di vicini che verranno considerati per determinare la classe di un campione;
- "*weights*": specificano come pesare il contributo dei vicini. Può assumere due valori: **uniform**, ogni vicino contribuisce in modo equo, indipendentemente dalla distanza e **distance**, i vicini più vicini pesano di più rispetto a quello più lontani
- "*metric*": definisce il tipo di distanza da utilizzare per misurare la similarità tra campioni

Dopo aver introdotto questi parametri abbiamo eseguito la GridSearch con StratifiedKFold Validation (k=5) con i seguenti valori:

```
"LogisticRegression": {
    "model": LogisticRegression(max_iter=1000),
    "param_grid": {
        "C": [0.01, 0.1, 1, 10, 100],
        "penalty": ["l2"],
        "solver": ["lbfgs"]
    },
},
"DecisionTree": {
    "model": DecisionTreeClassifier(random_state=42),
    "param_grid": {
        "max_depth": [None, 5, 10, 20],
        "min_samples_split": [2, 5, 10],
        "min_samples_leaf": [1, 2, 4]
    },
},
"RandomForest": {
    "model": RandomForestClassifier(random_state=42),
    "param_grid": {
        "n_estimators": [50, 100, 200],
        "max_depth": [None, 5, 10, 20],
        "min_samples_split": [2, 5, 10]
    },
},
"KNN": {
    "model": KNeighborsClassifier(),
    "param_grid": {
        "n_neighbors": [3, 5, 7, 9],
        "weights": ["uniform", "distance"],
        "metric": ["euclidean", "manhattan"]
    },
```

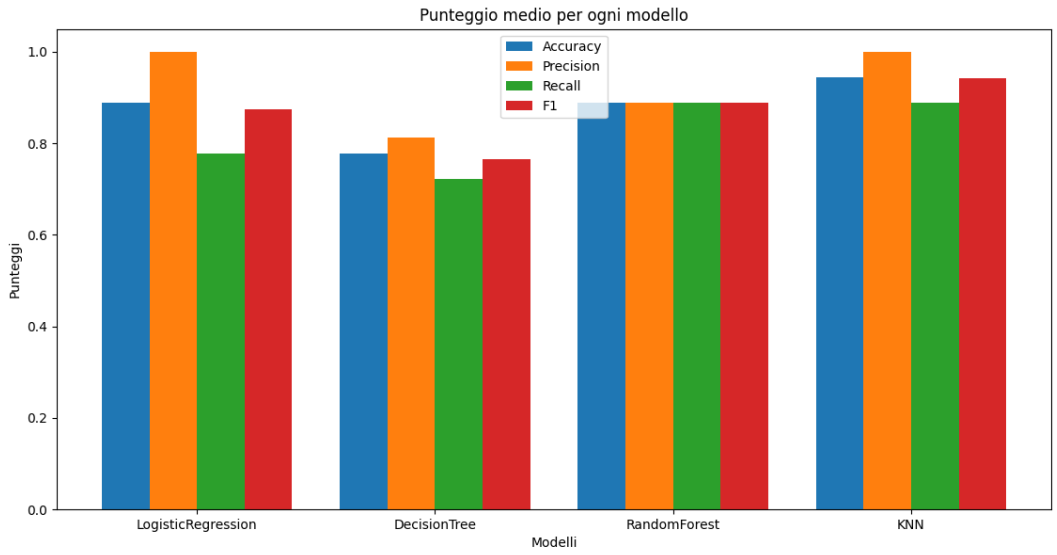
I risultati della GridSearch sono:

```
LogisticRegression: {'C': 100, 'penalty': 'l2', 'solver': 'lbfgs'}

DecisionTree: {'max_depth': None, 'min_samples_leaf': 4, 'min_samples_split': 10}

RandomForest: {'max_depth': None, 'min_samples_split': 5, 'n_estimators': 200}

KNN: {'metric': 'euclidean', 'n_neighbors': 9, 'weights': 'distance'}
```

Valutazione

Per poter testare le prestazioni dei modelli e ridurre l'overfitting[14] si è utilizzato la StratifiedKFold Validation, un modello di cross validation[15] che suddivide il set di dati totali in k parti uguali, dove a rotazione tutte le partizioni verranno usate come training set tranne una che verrà usata come validazione, quindi i modelli vengono addestrati k volte. In base alla scelta del valore k abbiamo utilizzato una serie di metriche per poter valutare il modello specifico. La scelta del valore k è stata fatta sulla base del consumo di acqua relativo a un mese e in questo caso abbiamo scelto il valore k=5 che indica il numero massimo di settimane in un mese. Quindi si è deciso di dividere il set di dati in 5 fold in modo tale da utilizzare a rotazione una settimana al mese come validazione(su cui non è stata addestrata).

Per valutare ogni modello specifico abbiamo utilizzato, come detto precedentemente, delle metriche come: Accuracy,Precision, Recall, F1[16]

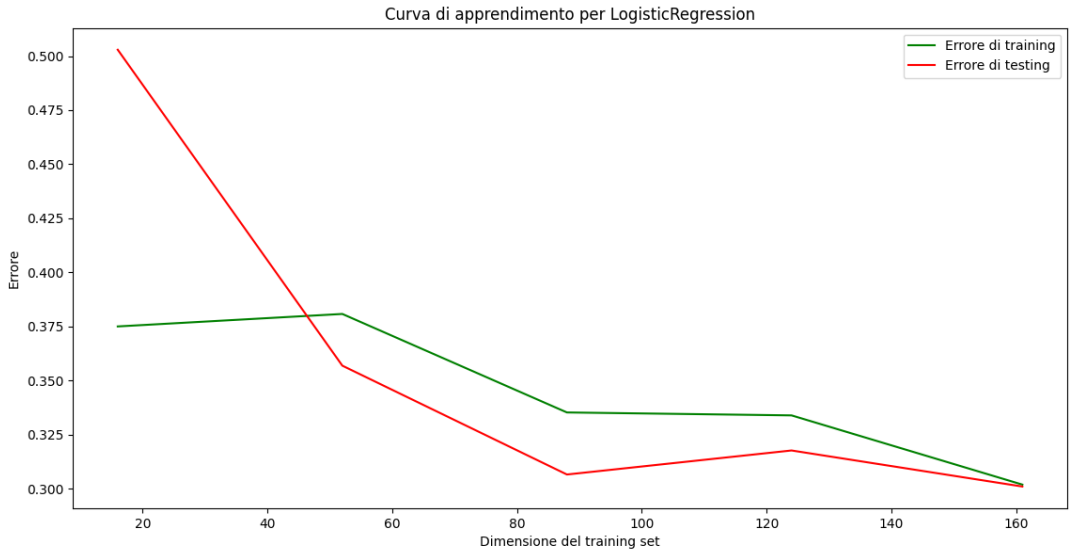


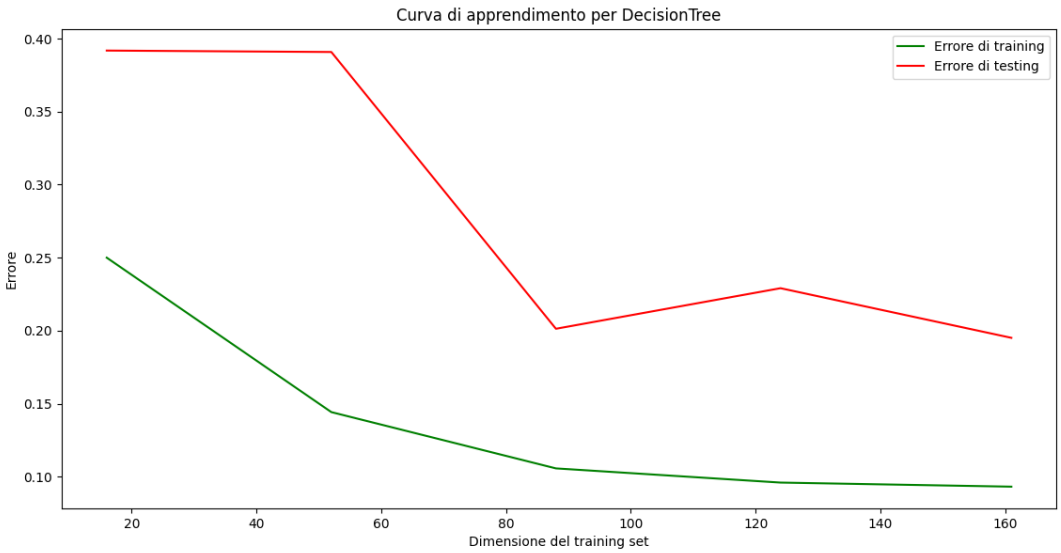
I risultati hanno evidenziato, come possiamo vedere dal grafico sopra, il K-NN e la Logistic Regression sono i modelli con le performance più equilibrate ottimizzando sia la precision(garantendo che ogni predizione positiva sia corretta) e sia la

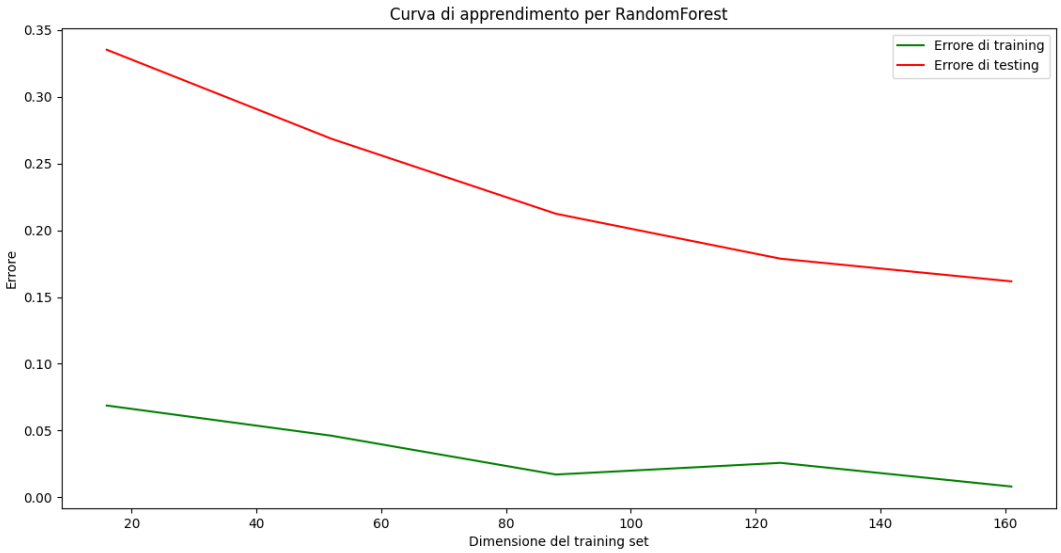
recall(prendendo la maggior parte dei casi positivi) con l'accuratezza più elevata e i migliori F1 score. Inoltre possiamo vedere che il modello Random Forest presenta valori uguali per le metriche e queste può essere segno di overfitting; e il modello Decision Tree mostra basse performance con recall e precision basse rispetto ai modelli precedenti. Per verificare in modo più ottimale le performance, e quindi verificare il modello con meno overfitting, si è analizzata la varianza, la deviazione standard[17] e le curve di apprendimento per ogni modello. Queste metriche misurano la dispersione dell'errore; un test error elevato indica che il modello è sensibile alla variazione dei dati, e questo significa possibile overfitting.

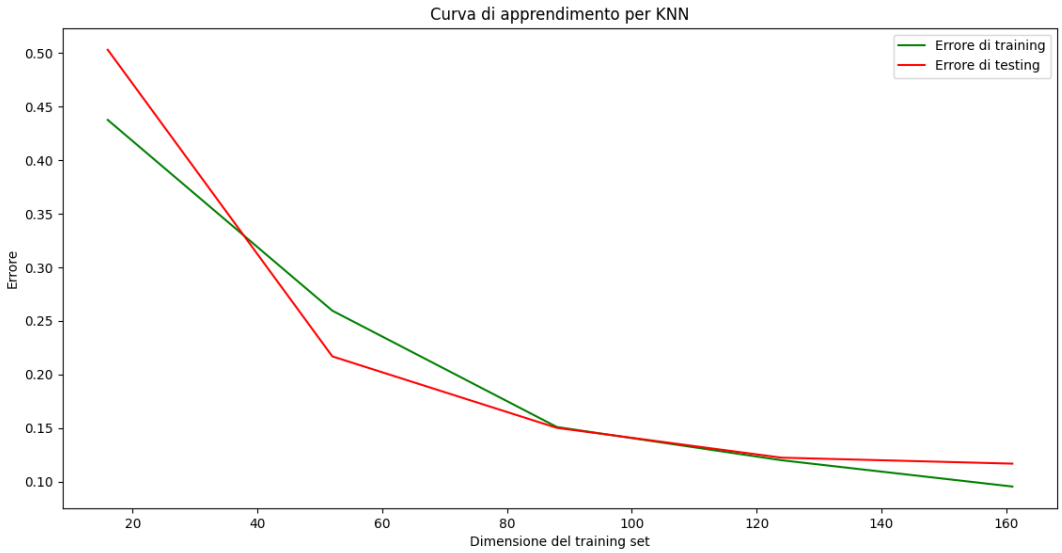
Modello	**Tipo di errore**	**Deviazione standard**	**Varianza**
Logistic Regression	Train	0.006334185731171173	4.0121908876972494e-05
Logistic Regression	Test	0.08161368057775602	0.006660792857447989
Decision Tree	Train	0.018425337856138282	0.0003394930751128425
Decision Tree	Test	0.08635184195089901	0.007456640608313042
Random Forest	Train	0.0039770957996477395	1.5817290999575695e-05
Random Forest	Test	0.08731363228720905	0.007623670383185954
K-NN	Train	0.00691647746935404	4.783766058408206e-05
K-NN	Test	0.09090373150699978	0.008263488401896705

Curve di apprendimento




Curva di apprendimento per DecisionTree
Errore di training
Errore di testing
Errore
Dimensione del training set
0.40
0.35
0.30
0.25
0.20
0.15
0.10
20
40
60
80
100
120
140
160


Curva di apprendimento per RandomForest
Errore di training
Errore di testing
Errore
Dimensione del training set
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
20
40
60
80
100
120
140
160



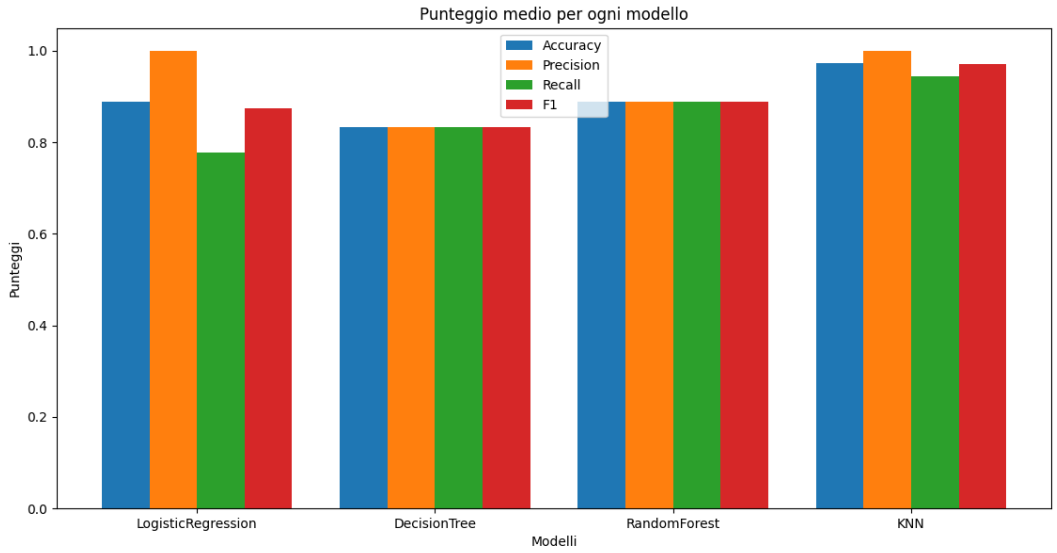
Analisi dei risultati dei modelli:

- **LogisticRegression**: i valori della varianza e deviazione standard sono bassi; la varianza del test error non è elevata rispetto a quella del train error. Inoltre, dal grafico possiamo vedere come il test error e il train error convergono con un errore basso, con il test error che parte da un errore elevato. Possiamo dire che questo modello non presenta overfitting
- **DecisionTree**: anche in questo caso la varianza e la deviazione standard hanno valori bassi. La differenza tra i due valori di varianza di test e train è abbastanza elevata è questo vuol dire presenza di overfitting. Dal grafico possiamo vedere che la curva parte con un errore elevato ma diminuisce non in modo netto. Quini potrebbe esserci overfitting.
- **RandomForest**: in questo caso la differenza tra i valori della varianza e i valori della deviazione standard è elevata. Anche se i valori sono bassi, è presente una distanza tra la varianza del train error e del test error. Tuttavia questo suggerisce la presenza di overfitting come si può notare dal grafico.
- **K-NN**: in questo modello la differenza tra le due varianza non è elevato. Dal grafico possiamo notare che la curva parte da un errore elevato fino a diminuire rapidamente con l'aumentare degli esempi, e quindi non ci sarebbe presenza di overfitting.

Quindi per diminuire l'overfitting sui modelli, abbiamo eseguito un oversampling sui dati di input e non sul target per evitare il data leaking[18], cioè che il modello train

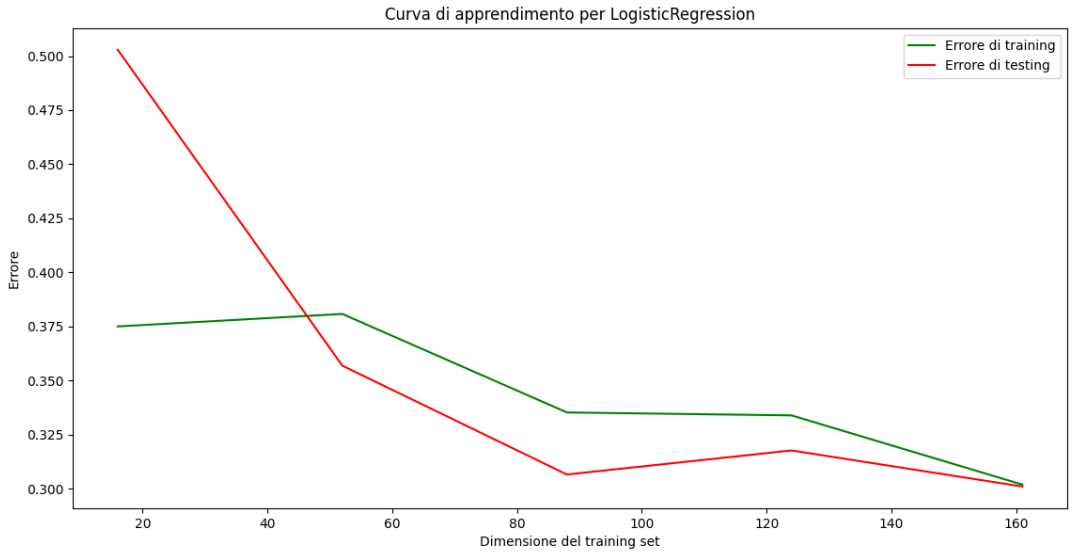
acquisisce alcune informazioni che andrebbero così ad influire sulla performance del modello, e quindi causerebbe la presenza di overfitting.

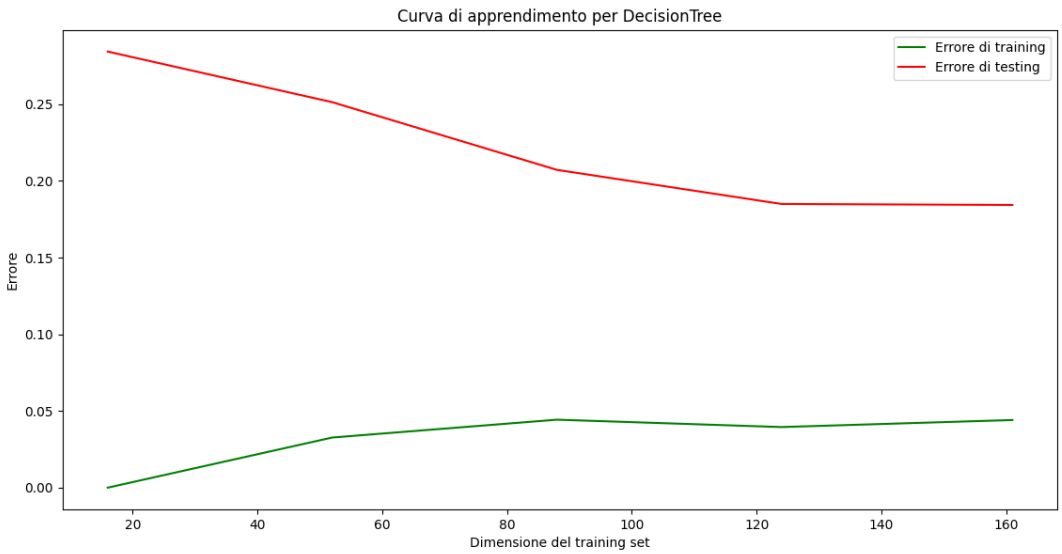
Metriche dopo l'oversampling (non sull'intero dataset)

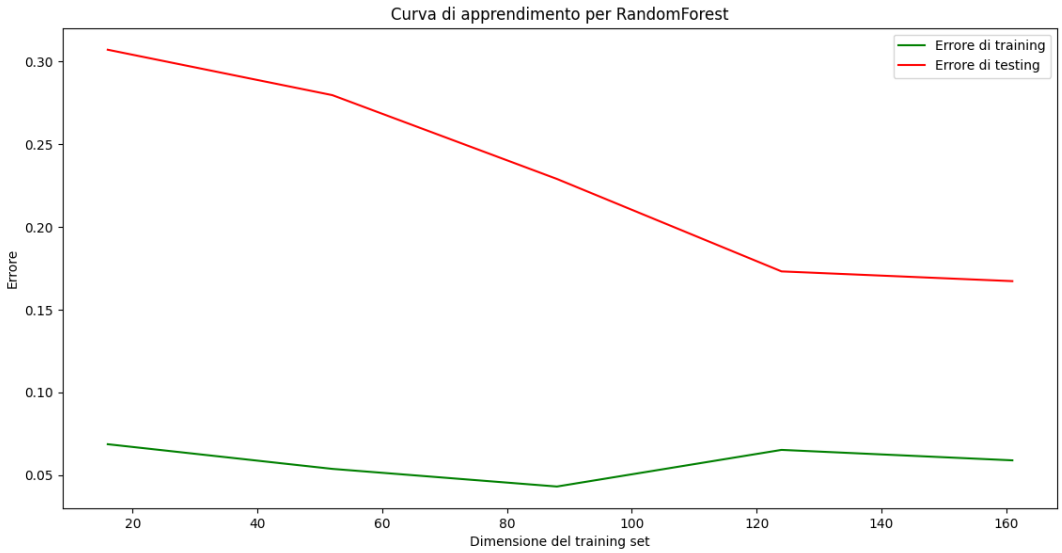


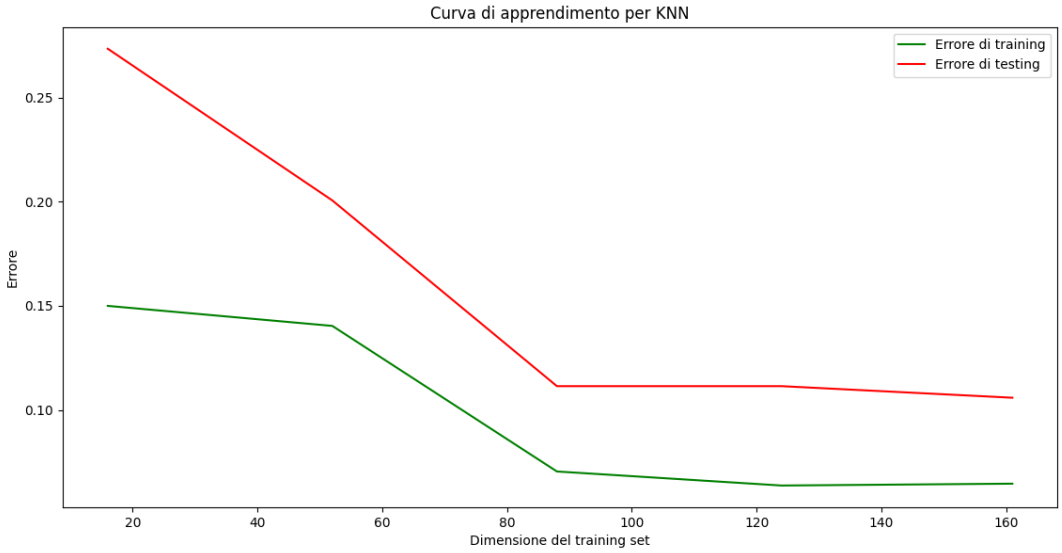
Modello	**Tipo di errore**	**Deviazione standard**	**Varianza**
Logistic Regression	Train	0.006334185731171173	4.0121908876972494e-05
Logistic Regression	Test	0.08161368057775602	0.006660792857447989
Decision Tree	Train	0.012250362064171447	0.00015007137070329092
Decision Tree	Test	0.10544167150188309	0.011117946089111024
Random Forest	Train	0.016491823661306533	0.00027198024767563
Random Forest	Test	0.07812850017790371	0.006104062540048699
K-NN	Train	0.004968944099378891	2.4690405462752297e-05
K-NN	Test	0.07616833176296091	0.005801614763552479

Curve di apprendimento dopo oversampling (non sull’intero dataset)









Dai grafici, riscontriamo una riduzione dell'overfitting in due modelli grazie all'uso dell'oversampling. Le metriche hanno confermato la presenza di overfitting nei modelli Decision Tree e Random Forest, a differenza dei modelli Logistic Regression e K-NN che presentano una buona performance e basso overfitting.

Constraint Satisfaction Problem (CSP)

Sommario

L'obiettivo dell'uso del CSP[19] è di risolvere un problema del seguente sistema: la carenza idrica. L'obiettivo di questo sistema è ottimizzare l'uso delle risorse idriche durante tutti i periodi di scarsità d'acqua. Quindi il modo per gestire le risorse è quello di riuscire a gestire l'acqua delle nazioni tramite l'utilizzo del CSP.

Strumenti utilizzati

Per la realizzazione e l'utilizzo del CSP sono state utilizzate le librerie AIPython[20] messe a disposizione del libro di testo adottato. È stato deciso di optare per una variante dell'algoritmo del CSP che implementa la ricerca branch and bound[21]. Questa ricerca richiede un solo parametro:

- **Bound** : è il limite entro il quale si cercherà la soluzione ottimale

A seguito di diversi test è stato scelto il seguente valore: bound = 50

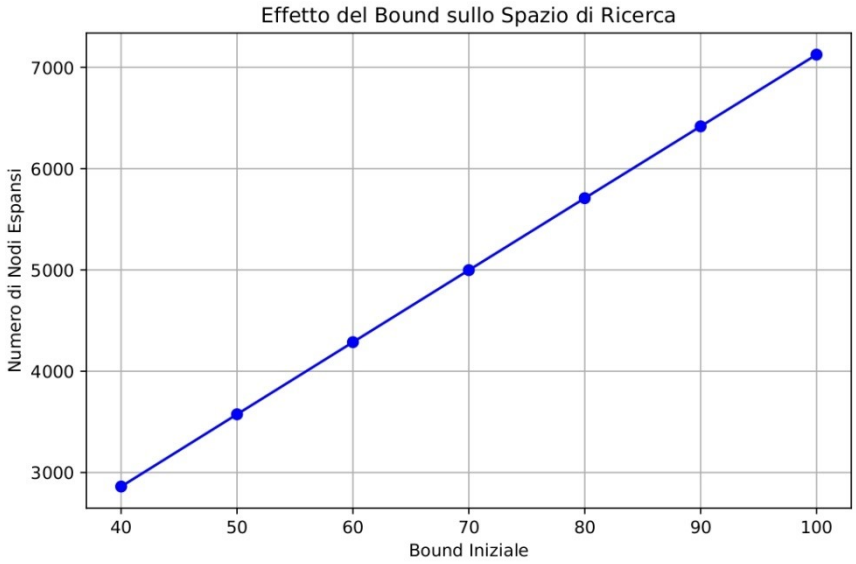
Decisioni di progetto

Sono state scelte come variabili le nazioni(i nodi) e come loro dominio un range di numeri che indicano la loro disponibilità idrica. Il CSP esegue possibili assegnazioni per ciascuna variabile scegliendo un solo valore dal dominio rispettando determinati vincoli. Per assegnazione di un solo valore a una variabile si intende la disponibilità d'acqua di una nazione. Abbiamo definito dei vincoli detti vincoli soft(Soft Constraint),con una funzione che definisce un vincolo che assegna penalità, utilizzando un coefficiente di penalità maggiore, in caso di risorse insufficienti , e un'altra funzione che definisce un vincolo che assegna una penalità minore in caso di uso minimo di acqua data una soglia minima di utilizzo(la media del dominio).

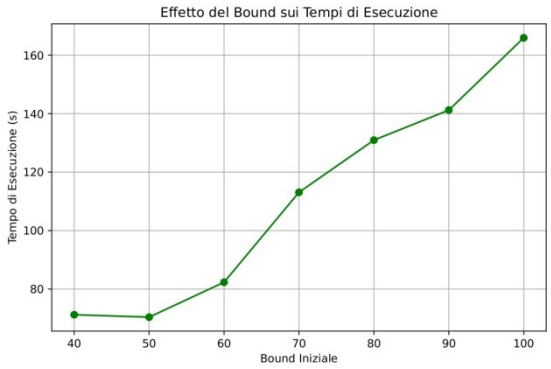
Valutazione

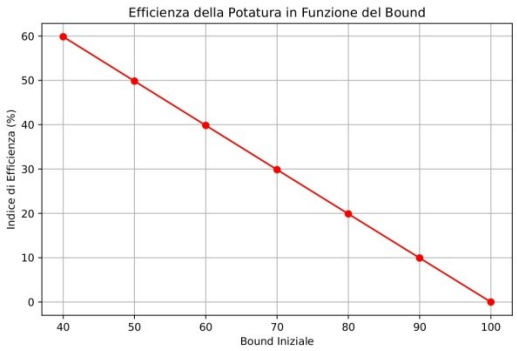
In questo caso abbiamo valutato l'efficienza dell'esecuzione dell'algoritmo branch and bound utilizzato per risolvere il CSP e in particolare della scelta del valore ottimale del parametro bound citato precedentemente. Per primo, abbiamo impostato un range di possibili valori per il parametro e in seguito analizzato il bound sul numero di nodi espansi (spazio di ricerca), sul tempo di esecuzione e in

base all'efficienza della potatura dei percorsi non utili alla risoluzione del problema del CSP.



Dall'analisi effettuata abbiamo osservato una crescita esponenziale dei nodi espansi all'aumentare dei valori del parametro bound. La scelta del valore del parametro bound, inoltre si è basata sull'analisi di altri due fattori:





Dal primo grafico, possiamo osservare che il tempo di esecuzione dell'algoritmo branch and bound ha un andamento esponenziale all'aumentare del bound; invece dal secondo grafico possiamo descrivere l'indice di efficienza riguardo la potatura dei percorsi, cioè sulla riduzione dei nodi espansi, rispetto a una base in cui il bound inziale è alto(minor riduzione dei nodi espansi).

Da questa analisi abbiamo scelto bound=50, dato che con tale valore i nodi espansi non sono elevati, il tempo di esecuzione è basso e quindi risulta efficiente e in ultima

analisi ha un indice di efficienza elevato, cioè si effettua una potatura efficiente eliminando una gran parte di nodi che non avrebbero portato alla soluzione ottimale del CSP.

Ragionamento probabilistico

Sommario

Il ragionamento probabilistico[22] è un tipo di ragionamento che utilizza la probabilità è in particolare l'indipendenza e dipendenza di variabili aleatorie e la regola di Bayes. Grazie a questo ragionamento possiamo utilizzare le reti di credenza(o reti bayesiane). Quindi, l'obiettivo del sistema è stato quello di progettare una rete bayesiana in grado di osservare le dipendenze probabilistiche tra le variabili.

Strumenti utilizzati

Per la realizzazione della rete bayesiana abbiamo utilizzato pgmpy per la creazione della rete bayesiana e networkx per la visualizzazione della rete bayesiana

Decisioni di progetto

Per costruire una rete bayesiana, c'è il bisogno di identificare le variabili e le dipendenze tra di esse. In questo caso le variabili sono : AcquaAnnuale, Popolazione, AcquaProCapite e Stato_idrico. Prima di apprendere la struttura si è deciso di discretizzare il dataset mediante KBinsDiscretizer il quale trasforma i valori continui in discreti.

Il passo successivo riguarda l'apprendimento della struttura di cui abbiamo fatto uso della tecnica nota come HillClimbSearch[23], che stima di trovare un modello con punteggio ottimale, in base al metodo di punteggio dato(in questo caso K2). Dopo aver appreso la struttura le CPD risultanti dal modello sono state le seguenti:

CPD per la variabile 'AcquaAnnuale':

AcquaAnnuale(0.0)	0.927374
AcquaAnnuale(1.0)	0.0446927
AcquaAnnuale(2.0)	0.00558659
AcquaAnnuale(3.0)	0.00558659
AcquaAnnuale(7.0)	0.00558659
AcquaAnnuale(9.0)	0.00558659
AcquaAnnuale(11.0)	0.00558659

CPD per la variabile 'Popolazione':

AcquaAnnuale	...	AcquaAnnuale(11.0)
AcquaProCapite	...	AcquaProCapite(11.0)
Stato_idrico	...	Stato_idrico(11.0)
Popolazione(0.0)	...	0.2
Popolazione(1.0)	...	0.2
Popolazione(2.0)	...	0.2
Popolazione(10.0)	...	0.2
Popolazione(11.0)	...	0.2

CPD per la variabile 'AcquaProCapite':

AcquaProCapite(0.0)	0.513966
AcquaProCapite(1.0)	0.0558659
AcquaProCapite(2.0)	0.0726257
AcquaProCapite(3.0)	0.0558659
AcquaProCapite(4.0)	0.0502793
AcquaProCapite(5.0)	0.0446927
AcquaProCapite(6.0)	0.0111732
AcquaProCapite(7.0)	0.0223464
AcquaProCapite(8.0)	0.0558659
AcquaProCapite(9.0)	0.0502793
AcquaProCapite(10.0)	0.0446927
AcquaProCapite(11.0)	0.0223464

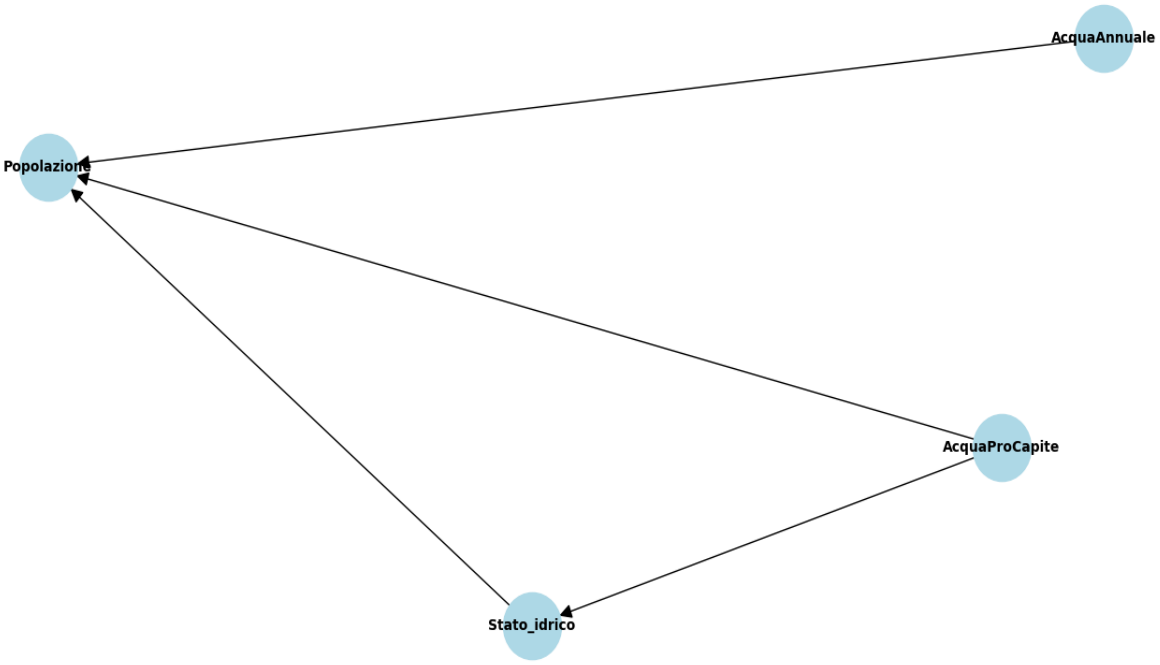
CPD per la variabile 'Stato_idrico':

AcquaProCapite	...	AcquaProCapite(11.0)
Stato_idrico(0.0)	...	1.0
Stato_idrico(11.0)	...	0.0

Da queste CPD risulta che AcquaAnnuale e AcquaProCapite sono variabili indipendenti, invece le variabili Popolazione e Stato_idrico sono dipendenti da:

- Popolazione : AcquaAnnuale, AcquaProCapite,Stato_idrico (P(11.0|AcquaAnnuale=11.0, AcquaProCapite=11.0, Stato_idrico=11.0) = 0.2)
- Stato_idrico : AcquaProCapite

Questo è la rete bayesiana risultante:



Ora osserviamo, grazie al modello precedente, la predizione effettuato su Stato_idrico(variabile target) date le variabili AcquaAnnuale, AcquaProCapite, Popolazione(osservazioni):

Stato_idrico	phi(Stato_idrico)
Stato_idrico(0.0)	0.0326
Stato_idrico(11.0)	0.9674

Valutazione

L'osservazione della rete bayesiana risultante è dovuta alla scelta nel modo in cui discretizzare le variabili e quindi sulla scelta del valore per il parametro n_bins (che divide l'intervallo di valori di ciascun variabile in un numero prestabilito di intervalli). Per la scelta del valore si sono utilizzate due metriche che verificano l'efficienza del modello: la log_likelihood[24] e il BIC[25].

Si è analizzato un range di valori da assegnare al parametro n_bins:

Valore	**Log_likelihood**	**BIC**
6	-309.78074684218905	-374.6230694151984
7	-331.28846291298396	-409.09925000059513
8	-340.7751077853428	-423.7732806787948
9	-364.3860098329218	-1261.8037542433722
10	-391.0241392950926	-1747.52552752245
11	-403.4240484526591	-2071.168585030462
12	-412.91113708873274	-2231.0898620359176

Dopo questa analisi si è scelto il valore 12, perché osserviamo un valore sia per il BIC che per la log_likelihood (risp. -2231.08986, -412.91113) molto piccolo e questo significa un modello più robusto e efficiente.

Conclusione

Il software può essere espanso integrandolo con l'introduzione di sensori che rilevano la quantità di acqua disponibile quotidianamente in tempo reale, e aiutare molto nella ricerca di soluzioni per ridurre gli sprechi e gestire in modo efficiente le risorse idriche. Potrebbe essere anche espanso in relazione con sistemi in grado di monitorare sia la pressione o eventuali perdite nella rete idrica, sia la qualità dell'acqua così da intervenire nella riparazione di guasti alla rete e successivamente nella depurazione dell'acqua.

Riferimenti bibliografici/sitografici

[1] https://artint.info/3e/html/ArtInt3e.Ch1.S6.html

[2] https://it.wikipedia.org/wiki/Prolog

[3] https://www.kaggle.com/datasets/shuvokumarbasak4004/global-water-usage-statistics

[4] https://artint.info/3e/html/ArtInt3e.Ch10.S3.html

[5] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

[6] https://en.wikipedia.org/wiki/Silhouette_(clustering)

[7] https://en.wikipedia.org/wiki/Oversampling_and_undersampling_in_data_analysis

[8] https://artint.info/3e/html/ArtInt3e.Ch7.S3.html

[9] https://artint.info/3e/html/ArtInt3e.Ch7.S3.html

[10] https://artint.info/3e/html/ArtInt3e.Ch7.S5.html

[11] https://it.wikipedia.org/wiki/K-nearest_neighbors

[12] https://it.wikipedia.org/wiki/Mediana_(statistica)

[13] https://artint.info/3e/html/ArtInt3e.Ch7.S2.html

[14] https://artint.info/3e/html/ArtInt3e.Ch7.S4.html

[15] https://artint.info/3e/html/ArtInt3e.Ch7.S4.html

[16] https://artint.info/3e/html/ArtInt3e.Ch7.S2.html

[17] https://artint.info/3e/html/ArtInt3e.Ch7.S4.html

[18] https://en.wikipedia.org/wiki/Leakage_(machine_learning)

[19] https://artint.info/3e/html/ArtInt3e.Ch4.S1.html

[20] https://artint.info/AIPython/

[21] https://artint.info/3e/html/ArtInt3e.Ch3.S6.html

[22] https://artint.info/3e/html/ArtInt3e.Ch9.S1.html

[23] https://artint.info/3e/html/ArtInt3e.Ch4.S6.html

[24] https://artint.info/3e/html/ArtInt3e.Ch7.S2.html

[25] https://artint.info/3e/html/ArtInt3e.Ch10.S2.html