Job’s

Documentazione del caso di studio Gruppo di lavoro:

- Carbonaro Vincenzo– 798281

- Giacò Simona – 798866

- Francesco De Giosa – 798712

AA 2025-2026

INTRODUZIONE

Per lo sviluppo del presente progetto è stato utilizzato il linguaggio di programmazione Python, con l'obiettivo di modellare e analizzare il problema del matching tra candidato e offerta di lavoro. Il dominio considerato riguarda la valutazione della compatibilità tra un profilo professionale e una specifica posizione lavorativa, integrando tecniche di apprendimento automatico, rappresentazione della conoscenza e ragionamento con vincoli.

Il dataset impiegato è di natura sintetica ed è stato strutturato in modo controllato per garantire coerenza tra le variabili numeriche e categoriali utilizzate nei diversi moduli del sistema. Le istanze rappresentano coppie candidato–offerta e includono sia feature descrittive sia una variabile target per il task di classificazione supervisionata.

La struttura del dataset comprende le seguenti feature:

skill_overlap: valore numerico compreso tra 0 e 1 che rappresenta il grado di sovrapposizione tra le competenze richieste dall'offerta e quelle possedute dal candidato.

years_experience: anni di esperienza del candidato nel settore di riferimento.

salary_offered: salario annuale proposto dall'offerta di lavoro.

distance_km: distanza geografica tra la sede lavorativa e la residenza del candidato.

seniority: livello richiesto per la posizione (junior, mid, senior).

contract_type: tipologia contrattuale prevista (internship, full_time, part_time, freelance).

remote: modalità di svolgimento del lavoro (yes/no).

match: variabile binaria che indica se la coppia candidato–offerta costituisce un buon match (1) oppure no (0).

Gli obiettivi del progetto sono:

predire la probabilità di compatibilità tra candidato e offerta mediante modelli di apprendimento supervisionato;

individuare segmenti omogenei di mercato e possibili anomalie tramite tecniche di clustering;

In particolare, il progetto mira a:

- modellare le dipendenze probabilistiche tra le variabili del dominio attraverso una rete bayesiana, al fine di stimare la probabilità condizionata $P(match = 1 \mid evidenze)$e analizzare in modo interpretabile le relazioni tra competenze, seniority, condizioni contrattuali e compatibilità complessiva;
- rappresentare esplicitamente la conoscenza del dominio mediante una Knowledge Base simbolica, in cui ruoli, requisiti e vincoli sono formalizzati attraverso fatti e regole, consentendo operazioni di inferenza e verifica di eleggibilità;
- costruire una short-list ottimale di offerte compatibili con un candidato, formalizzando il problema come un Constraint Satisfaction Problem (CSP) con vincoli hard (obbligatori) e soft (preferenze), risolto tramite tecniche di ricerca locale e metaeuristiche.

Il progetto si propone dunque di integrare modelli statistici, probabilistici e simbolici all'interno di un unico sistema coerente, dimostrando come approcci differenti possano cooperare nella risoluzione di un problema decisionale complesso, in linea con i principali temi affrontati nel corso di Ingegneria della Conoscenza.

ELENCO DI ARGOMENTI DI INTERESSE

Gli argomenti oggetto del corso di Ingegneria della Conoscenza che sono stati riproposti e applicati nel presente progetto sono i seguenti.

Apprendimento non supervisionato:
È stato utilizzato l'algoritmo K-Means, tecnica di hard-clustering in cui ogni istanza appartiene esclusivamente a un singolo cluster. L'obiettivo è stato quello di individuare segmenti omogenei di coppie candidato–offerta sulla base delle feature numeriche e categoriali

disponibili. Attraverso l'analisi della silhouette e della curva elbow è stato determinato il numero ottimale di cluster. L'approccio ha permesso di evidenziare differenti profili di mercato e di individuare possibili anomalie tramite l'analisi della distanza dai centroidi.

Apprendimento supervisionato:
Sono state adottate diverse tecniche di machine learning per affrontare il problema di classificazione binaria relativo alla variabile match. Dopo una fase di preprocessing (standardizzazione delle variabili numeriche e codifica one-hot delle variabili categoriali), sono stati addestrati e confrontati differenti modelli: K-Nearest Neighbors (KNN), Random Forest, Support Vector Machine (SVM), Decision Tree e Gradient Boosting. La selezione degli iperparametri è stata effettuata tramite GridSearch con cross-validation stratificata, e la valutazione è stata condotta mediante metriche quali ROC-AUC, Average Precision e GMAP, considerando medie e deviazioni standard su più run.

Ragionamento su conoscenza incerta (Modelli probabilistici):
È stata costruita una rete bayesiana discreta per modellare le dipendenze condizionali tra variabili del dominio (seniority, contract_type, remote, skill_overlap, distanza, ecc.). Attraverso l'inferenza probabilistica è stato possibile stimare la probabilità $P(match = 1 \mid evidenze)$, fornendo un modello interpretabile delle relazioni tra le caratteristiche del candidato e dell'offerta.

Rappresentazione e ragionamento simbolico:
È stata sviluppata una Knowledge Base contenente fatti e regole relative a ruoli, competenze richieste, tipologie contrattuali e vincoli di eleggibilità. La KB consente di eseguire query dichiarative per verificare l'ammissibilità di un candidato rispetto a un'offerta e per generare raccomandazioni sulla base di vincoli espliciti.

Risoluzione di un CSP:
La costruzione di una short-list di offerte compatibili con un candidato è stata formalizzata come un Constraint Satisfaction Problem con vincoli hard (es. salario minimo, distanza massima) e soft (es. diversità di seniority o tipologia contrattuale). Il problema è stato risolto mediante tecniche di ricerca locale, in particolare Random Walk e Simulated Annealing, analizzando la qualità delle soluzioni in termini di funzione di penalità e robustezza dei risultati.

Apprendimento Non Supervisionato

Nel contesto del mercato del lavoro, il processo di matching tra candidato e offerta non dipende unicamente da una decisione binaria di compatibilità, ma presenta spesso una struttura segmentata. Esistono infatti differenti "tipologie" di abbinamento, caratterizzate da combinazioni ricorrenti di esperienza, competenze, livello salariale e modalità di lavoro.

L'obiettivo dell'analisi non supervisionata è stato quindi quello di esplorare la struttura latente del mercato del lavoro rappresentato nel dataset, identificando gruppi omogenei di istanze candidato–offerta senza utilizzare la variabile target match. Questo consente di comprendere se esistano segmenti naturali (ad esempio posizioni senior ad alta retribuzione, offerte junior a bassa distanza, contratti flessibili con modalità remote, ecc.) e di individuare combinazioni atipiche.

Caricamento e Pulizia dei Dati

Il dataset utilizzato per l'analisi non supervisionata è stato caricato da file CSV tramite la libreria pandas.

Prima di procedere al clustering, è stata effettuata una fase preliminare di validazione e pulizia dei dati, necessaria per garantire coerenza e robustezza nell'addestramento del modello.

In particolare:

- è stata verificata la presenza di tutte le colonne richieste (skill_overlap, years_experience, salary_offered, distance_km, seniority, contract_type, remote);
- sono state eliminate le istanze contenenti valori mancanti nelle colonne considerate, tramite rimozione delle righe incomplete;
- non sono state effettuate imputazioni, al fine di evitare l'introduzione di bias artificiale nelle distanze utilizzate dal clustering.

La scelta di eliminare le istanze incomplete, anziché imputare valori stimati, è coerente con la natura dell'algoritmo K-Means, che si basa sul calcolo diretto delle distanze tra punti nello spazio delle feature. L'inserimento di valori imputati potrebbe alterare la struttura geometrica dei dati e influenzare in modo non controllato l'assegnazione ai cluster.

Questa fase garantisce che il modello operi esclusivamente su dati coerenti e completi, preservando l'integrità dell'analisi.

Nota Metodologica

L'eliminazione delle istanze incomplete rappresenta una scelta metodologica consapevole.

Sebbene la rimozione dei dati mancanti possa ridurre la numerosità complessiva del campione, essa garantisce una maggiore stabilità nel calcolo delle distanze euclidee, elemento centrale nell'algoritmo K-Means.

Tecniche di imputazione non supervisionata, se applicate in questo contesto, avrebbero potuto alterare la struttura geometrica dello spazio delle feature, influenzando in modo artificiale la formazione dei cluster.

In particolare, l'inserimento di valori medi o stimati potrebbe:

- ridurre la varianza reale dei dati;
- avvicinare artificialmente punti che in realtà presentano caratteristiche differenti;
- generare cluster meno interpretabili dal punto di vista semantico.

Si è quindi preferito preservare l'integrità strutturale del dataset, privilegiando coerenza geometrica e interpretabilità rispetto alla massimizzazione della numerosità.

Questa scelta è coerente con l'obiettivo dell'analisi non supervisionata, che non mira alla predizione, ma alla comprensione delle strutture latenti nel dominio candidato–offerta.

Selezione delle Feature

Per il clustering sono state considerate tutte le variabili descrittive del problema, ad esclusione della variabile target:

Variabili numeriche:

- skill_overlap: misura della compatibilità tecnica tra competenze richieste e possedute;
- years_experience: anni di esperienza del candidato;
- salary_offered: retribuzione proposta dall'azienda;
- distance_km: distanza geografica tra candidato e sede lavorativa.

Variabili categoriali:

- seniority: livello richiesto (junior, mid, senior);
- contract_type: tipologia contrattuale (internship, full_time, part_time, freelance);
- remote: modalità di lavoro (yes/no).

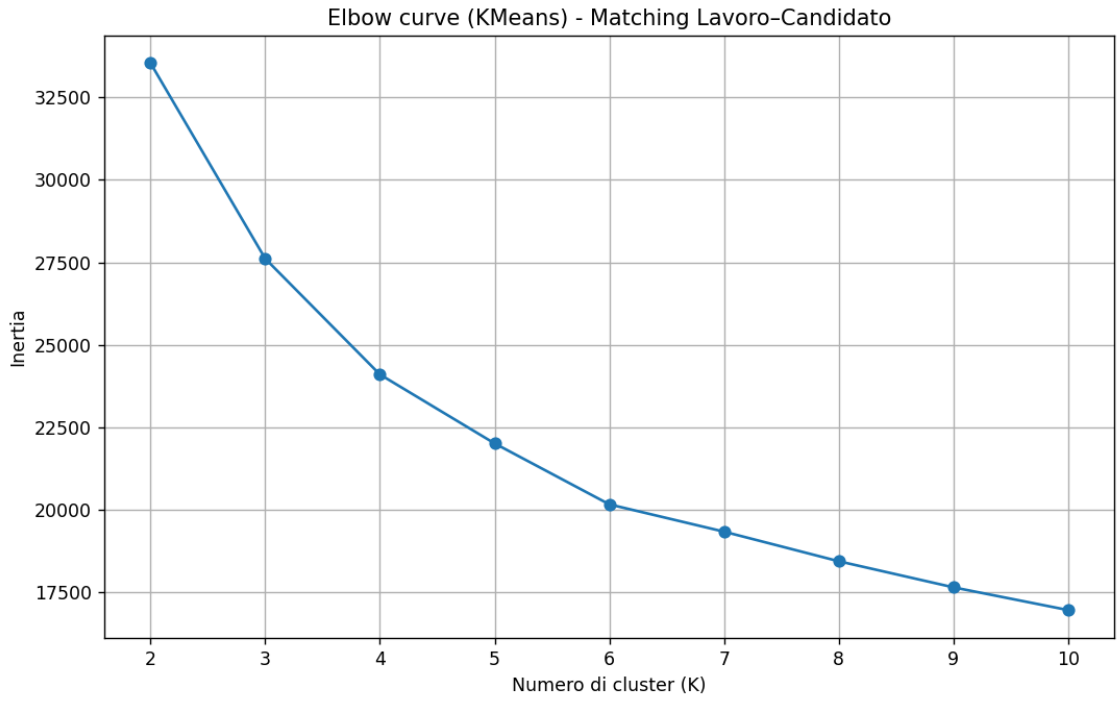
Queste variabili rappresentano dimensioni centrali del contesto lavorativo reale: compatibilità tecnica, esperienza professionale, condizioni economiche e vincoli logistici.

La scelta del numero di cluster K è stata effettuata in modo data-driven.

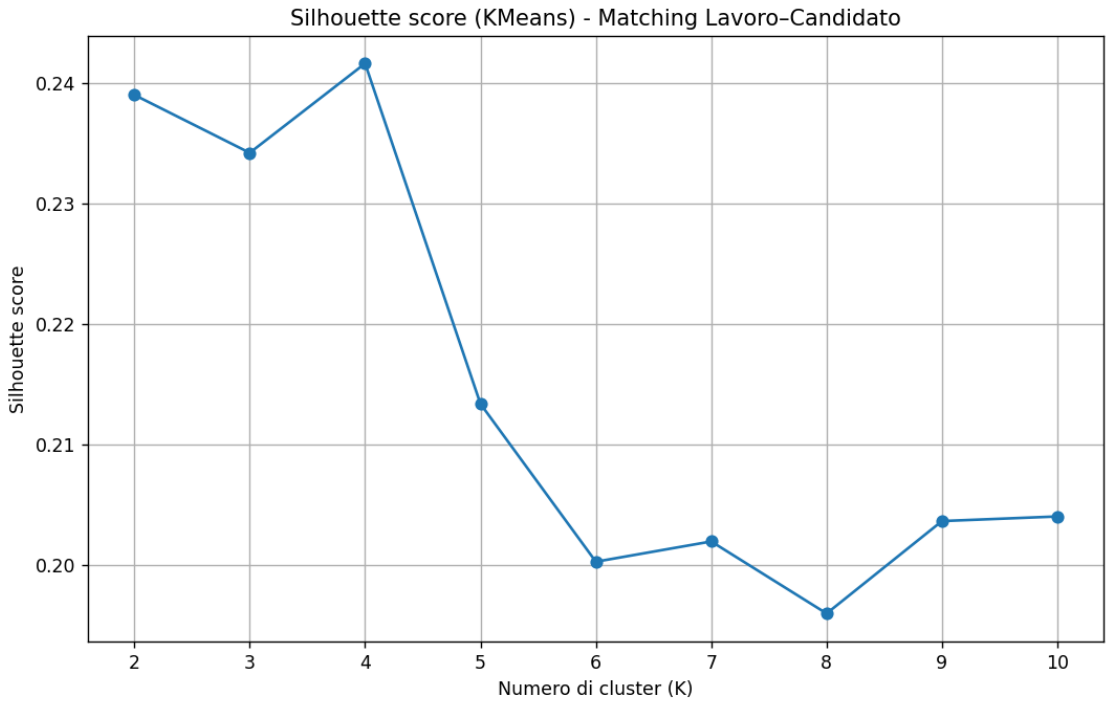
Sono stati testati diversi valori di K, valutando per ciascuno:

- l'**inertia** (somma delle distanze quadratiche dai centroidi);
- il **Silhouette Score**, utilizzato come criterio principale.

Il valore finale di K è stato scelto massimizzando la separabilità media tra cluster, bilanciando compattezza interna e separazione esterna.



La curva elbow mostra la riduzione dell'inertia all'aumentare del numero di cluster. Si osserva una diminuzione progressiva, senza un punto di flesso particolarmente marcato, rendendo non univoca la scelta di K basata esclusivamente su questo criterio.

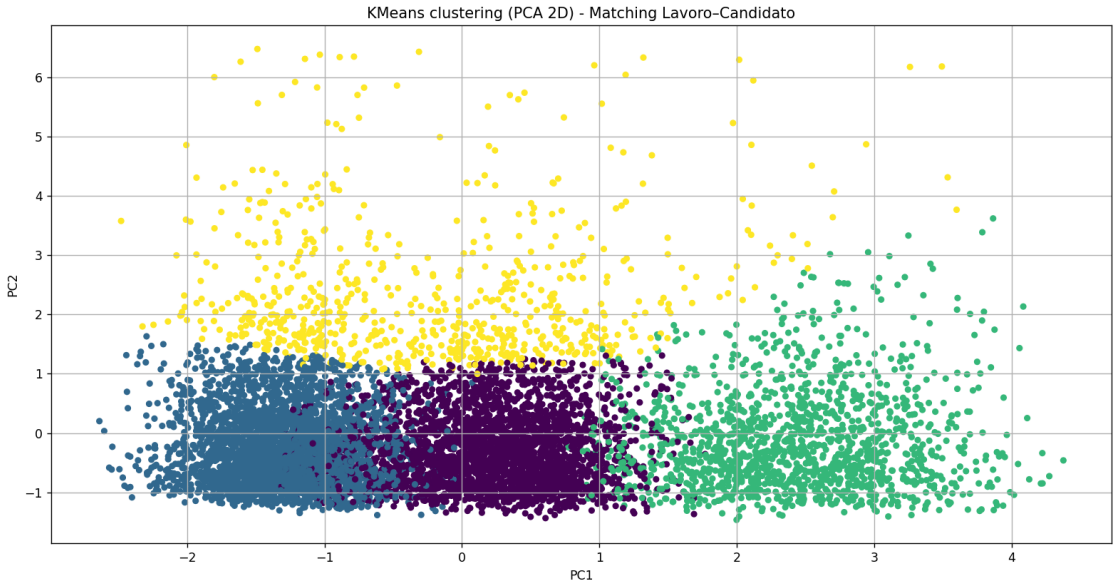


Il Silhouette Score misura la coerenza interna dei cluster rispetto alla loro separazione. Il valore massimo si ottiene per:

K=4

Pertanto, il modello finale è stato addestrato utilizzando quattro cluster.

Il valore del Silhouette Score risulta pari a circa 0.24. Sebbene non particolarmente elevato, tale valore è coerente con la natura del dominio lavorativo, in cui le variabili (esperienza, salario, distanza, compatibilità tecnica) variano in modo continuo piuttosto che generare segmentazioni nettamente separate.



Visualizzazione dei Cluster tramite PCA

Per facilitare l'interpretazione della segmentazione ottenuta con l'algoritmo K-Means, è stata applicata una **Principal Component Analysis (PCA)** ai dati preprocessati (standardizzazione delle variabili numeriche e codifica one-hot delle variabili categoriali).

La PCA consente di ridurre la dimensionalità dello spazio delle feature, proiettando le istanze sulle prime due componenti principali (PC1 e PC2), che rappresentano le direzioni di massima varianza dei dati.

Interpretazione del Grafico

Nel grafico:

1. **Assi**
    - L'asse delle ascisse (PC1) rappresenta la prima componente principale.
    - L'asse delle ordinate (PC2) rappresenta la seconda componente principale.

È importante sottolineare che PC1 e PC2 non corrispondono a singole feature originali (come salario o esperienza), ma a combinazioni lineari delle variabili preprocessate. La visualizzazione non deve quindi essere interpretata come una rappresentazione diretta delle feature originarie, ma come una proiezione sintetica dello spazio multidimensionale.

2. **Colori dei punti**
    - I punti sono colorati in base al cluster assegnato dall'algoritmo K-Means.
    - Ogni colore identifica un gruppo di istanze candidato–offerta con caratteristiche strutturalmente simili nello spazio delle feature.

Analisi della Separazione

Dalla visualizzazione si osserva una separazione parziale tra i cluster, principalmente lungo la prima componente principale (PC1), mentre lungo la seconda componente (PC2) si nota una maggiore sovrapposizione tra gruppi.

Questa distribuzione è coerente con il valore del Silhouette Score ottenuto (circa 0.24), che indica una segmentazione presente ma non fortemente marcata. Il dataset non mostra cluster completamente isolati, bensì gruppi con confini parzialmente sovrapposti.

Tale comportamento è coerente con la natura del dominio lavorativo, in cui le variabili (esperienza, compatibilità tecnica, salario, distanza) tendono a variare in modo graduale piuttosto che generare separazioni rigidamente definite.

Considerazioni Metodologiche

La PCA è stata utilizzata esclusivamente a fini di visualizzazione e non influisce sulla formazione dei cluster, che avviene nello spazio multidimensionale completo.

Il grafico rappresenta quindi uno strumento interpretativo, utile per comprendere la distribuzione dei cluster, ma non costituisce il criterio decisionale utilizzato per la scelta del numero ottimale di cluster.

Rilevazione dei Dati Anomali

Obiettivo e contesto

Nel dominio del matching Lavoro–Candidato, alcune coppie candidato–offerta possono risultare **atipiche** rispetto alla struttura generale dei dati: non necessariamente "sbagliate", ma potenzialmente interessanti perché rare (es. combinazioni estremamente sbilanciate tra compatibilità tecnica, distanza geografica e condizioni economiche).

In questo progetto l'analisi delle anomalie è stata utilizzata come estensione naturale del clustering: una volta ottenuti i cluster, si individuano i punti che risultano **molto distanti** dalla "configurazione tipica" del proprio cluster.

L'approccio implementato si basa sulla distanza euclidea tra ogni istanza e il **centroide del cluster più vicino**, calcolata nello stesso spazio in cui K-Means opera (cioè **dopo preprocessing**):

1. Si applica al dataset la trasformazione usata nel clustering:
    - standardizzazione delle feature numeriche (StandardScaler);
    - codifica one-hot delle feature categoriali (OneHotEncoder).
2. Si calcolano le distanze tra ciascun punto trasformato e tutti i centroidi del modello K-Means.
3. Per ogni istanza si considera la distanza minima dal centroide (cluster più vicino):

Per definire un criterio oggettivo e replicabile di anomalia, è stata fissata una soglia percentilica:

- si calcola il **95° percentile** della distribuzione delle distanze $d(x)$;
- si considerano anomalie le istanze con $d(x)$superiore a tale soglia.

La scelta del 95° percentile è un compromesso metodologico:

- percentili più bassi (es. 90°) produrrebbero un numero di outlier troppo elevato, riducendo la specificità del concetto di "anomalia";
- percentili più alti (es. 99°) individuerebbero solo casi estremi, rischiando di trascurare istanze rare ma informative.

Output e utilizzo

Le istanze individuate come anomalie vengono riportate in un file dedicato, includendo:

- cluster assegnato;
- distanza dal centroide;
- valori originali delle feature.

Questo consente un'analisi successiva mirata (ad esempio per verificare se tali combinazioni siano effettivamente anomale nel contesto lavorativo o se dipendano da particolarità/rumore nella generazione dei dati).

Apprendimento supervisionato

Introduzione

Nel nostro caso di studio, l'apprendimento supervisionato è stato utilizzato con lo scopo di predire se una coppia *(candidato, offerta)* costituisca un buon match, modellando il problema come **classificazione binaria** con target:

$$match \in \{0,1\}$$

dove:

- 1 indica un buon match,
- 0 indica un match non soddisfacente.

Il problema non è banale, poiché il concetto di "buon match" dipende dall'interazione tra più variabili eterogenee: competenze (skill overlap), esperienza, condizioni contrattuali, salario e

distanza geografica. Tali fattori possono interagire in modo non lineare, rendendo necessario l'utilizzo di modelli capaci di apprendere relazioni complesse.

L'approccio supervisionato è stato scelto perché consente di sfruttare esempi etichettati per apprendere una funzione decisionale capace di **generalizzare su nuove coppie candidato–offerta**, valutandone formalmente le prestazioni.

Per questa parte di progetto sono state utilizzate le seguenti librerie:

- **Pandas** e **NumPy** per la manipolazione e gestione dei dati
- **Scikit-learn (sklearn)** per pipeline, modelli di classificazione, cross-validation e ottimizzazione degli iperparametri
- **Matplotlib** per la visualizzazione delle curve di valutazione (ROC e learning curve)

L'intero processo è stato implementato in forma **end-to-end tramite una Pipeline sklearn**, integrando preprocessing e classificatore in un unico oggetto addestrabile.
Questa scelta:

- evita fenomeni di **data leakage**
- garantisce che le trasformazioni siano apprese esclusivamente sui dati di training
- rende l'esperimento completamente replicabile
- semplifica l'ottimizzazione tramite GridSearchCV

Caricamento e pulizia dei dati

Il dataset è stato caricato dal file:

datasets/jobs-matching_supervised.csv

Sono state considerate le colonne minime necessarie a garantire coerenza del task:

Variabili numeriche

- skill_overlap
- years_experience
- salary_offered
- distance_km

Variabili categoriche

- seniority
- contract_type

- remote

Target

- match

Per prevenire problemi dovuti a record incompleti:

- sono state rimosse le righe con valori mancanti nelle colonne sopra elencate
- il target match è stato convertito a interi e filtrato per mantenere solo valori validi {0,1}

La pulizia è stata volutamente **minima ma sufficiente**, poiché:

- preserva il maggior numero possibile di esempi
- evita imputazioni arbitrarie non giustificate dal dominio
- riduce il rischio di introdurre bias artificiali

Preprocessing dei dati

Le feature presentano scale molto differenti (es. salary_offered rispetto a skill_overlap) e natura mista (numerica e categorica).
Per questo motivo è stato adottato un preprocessing distinto per tipologia.

Trasformazione variabili numeriche

Le variabili numeriche sono state standardizzate mediante:

$$StandardScaler$$

Motivazione:

- rende confrontabili variabili con ordini di grandezza diversi
- migliora la stabilità numerica
- è essenziale per modelli sensibili alla scala (SVM, KNN)

Trasformazione variabili categoriche

Le variabili categoriche sono state codificate tramite:

$$OneHotEncoder(handle\_unknown = "ignore")$$

Motivazione:

- evita l'introduzione di un ordine artificiale tra categorie (che si avrebbe con LabelEncoder)

- consente la gestione robusta di categorie non viste in fase di training
- garantisce compatibilità con tutti i modelli del model zoo

Integrazione nella Pipeline

Il preprocessing è stato integrato in una **ColumnTransformer**, inserita in una Pipeline insieme al classificatore.

Questo garantisce che:

- scaler ed encoder siano stimati esclusivamente sui dati di training durante la cross-validation
- ogni trasformazione venga applicata in modo coerente durante training e test
- non vi sia contaminazione del test set

Suddivisione in training set e test set

Il dataset è stato suddiviso mediante train_test_split in:

- **70% training set**
- **30% test set**

con:

- stratify = y per preservare la distribuzione delle classi
- random_state fissato per garantire riproducibilità

La stratificazione è particolarmente importante in un problema binario, poiché consente di evitare squilibri accidentali tra training e test.

Oltre alla suddivisione train/test, la selezione dei modelli e degli iperparametri è stata effettuata tramite **validazione incrociata stratificata (5-fold)** sul training set.
Questa scelta consente di:

- ottenere una stima più stabile della capacità di generalizzazione
- ridurre la dipendenza da un singolo split
- evitare valutazioni basate su un unico run

Scelta dei modelli

Nel nostro caso di studio, l'apprendimento supervisionato è stato utilizzato con scopo di classificazione binaria.

Per effettuare tale classificazione, è stato adottato un **model zoo** composto da cinque modelli eterogenei:

- **Random Forest**
- **Support Vector Machine (SVM) con kernel RBF**
- **Decision Tree**
- **Gradient Boosting Classifier**
- **K-Nearest Neighbors (KNN)**

La scelta di modelli differenti è motivata dalla volontà di confrontare approcci con diversa capacità di modellare relazioni non lineari e diversa complessità strutturale:

- Modelli ad albero (RF, DT, GB) → robusti e interpretabili
- Modelli basati su margine (SVM) → efficaci in spazi ad alta dimensionalità
- Modelli basati su distanza (KNN) → sensibili alla struttura locale dello spazio

Non è stato selezionato un modello a priori: la scelta finale è stata guidata esclusivamente dalla valutazione empirica.

Scelta degli iperparametri

Per ottimizzare le prestazioni dei modelli è stato adottato il metodo della **Grid Search** in combinazione con **validazione incrociata stratificata**.

L'ottimizzazione è stata implementata tramite:

```
GridSearchCV(
  pipeline,
  param_grid=param_grid,
  scoring="roc_auc",
  cv=StratifiedKFold(n_splits=5, shuffle=True, random_state=42),
  n_jobs=-1
)
```

Motivazione della scelta

- La **Grid Search** consente un'esplorazione sistematica dello spazio degli iperparametri.
- La **StratifiedKFold** mantiene la distribuzione delle classi in ogni fold.
- L'ottimizzazione è stata effettuata sulla metrica **ROC-AUC**, scelta perché:
    - indipendente dalla soglia di classificazione
    - robusta in presenza di possibile sbilanciamento tra classi

- più informativa della sola accuracy

Il numero di split utilizzato è **5**, garantendo un buon compromesso tra stabilità della stima e costo computazionale.

Random Forest

Il modello Random Forest è stato configurato esplorando diversi parametri che influenzano la complessità e la capacità di generalizzazione del modello.

❖ **n_estimators**: numero di alberi nella foresta.
Un numero maggiore di alberi riduce la varianza del modello ma aumenta il costo computazionale.
Valori esplorati: **200, 400**

❖ **max_depth**: profondità massima degli alberi.
Limita la crescita dell'albero per controllare l'overfitting.
Valori esplorati: **None, 8, 16**

❖ **min_samples_split**: numero minimo di campioni necessari per dividere un nodo interno.
Parametro di regolarizzazione che riduce la complessità dell'albero.
Valori esplorati: **2, 5**

Codice – Griglia Random Forest

```
rf_param_grid = {
  "clf__n_estimators": [200, 400],
  "clf__max_depth": [None, 8, 16],
  "clf__min_samples_split": [2, 5],
}
```

Support Vector Machine (SVM – kernel RBF)

Per il modello SVM è stato utilizzato il kernel RBF, adatto a modellare relazioni non lineari tra le variabili del dominio.

❖ **C**: parametro di regolarizzazione.
Controlla il compromesso tra ampiezza del margine e penalizzazione degli errori.
Valori esplorati: **0.5, 1, 5**

❖ **gamma**: definisce l'influenza dei singoli punti di training.
Valori elevati rendono il modello più flessibile ma potenzialmente più soggetto a overfitting.
Valori esplorati: **"scale", 0.1, 0.01**

Codice – Griglia SVM

```
svm_param_grid = {
  "clf__C": [0.5, 1, 5],
  "clf__gamma": ["scale", 0.1, 0.01],
}
```

Decision Tree

Il Decision Tree è stato configurato per confrontare modelli con diversa profondità e diversa capacità espressiva.

❖ **max_depth** ∈ {None, 5, 10, 20}

❖ **min_samples_split** ∈ {2, 5, 10}

Codice – Griglia Decision Tree

```
dt_param_grid = {
  "clf__max_depth": [None, 5, 10, 20],
  "clf__min_samples_split": [2, 5, 10],
}
```

Gradient Boosting Classifier

Il modello Gradient Boosting costruisce alberi sequenziali che correggono gli errori dei precedenti.

❖ **n_estimators** ∈ {150, 300}

❖ **learning_rate** ∈ {0.05, 0.1}

❖ **max_depth** ∈ {2, 3}

Il parametro learning_rate controlla quanto ogni albero contribuisce alla correzione dell'errore residuo.

Codice – Griglia Gradient Boosting

```
gb_param_grid = {
  "clf__n_estimators": [150, 300],
  "clf__learning_rate": [0.05, 0.1],
  "clf__max_depth": [2, 3],
}
```

K-Nearest Neighbors (KNN)

Il modello KNN classifica un punto sulla base dei suoi vicini più prossimi nello spazio delle feature.

❖ **n_neighbors** ∈ {5, 11, 21}

❖ **weights** ∈ {"uniform", "distance"}

weights="distance" assegna maggiore importanza ai vicini più vicini.

Codice – Griglia KNN

```
knn_param_grid = {
  "clf__n_neighbors": [5, 11, 21],
  "clf__weights": ["uniform", "distance"],
}
```

Addestramento con GridSearch e Cross-Validation

L'ottimizzazione è stata effettuata tramite GridSearchCV con validazione incrociata stratificata a 5 fold, massimizzando la metrica ROC-AUC.

Codice – GridSearchCV

```
from sklearn.model_selection import GridSearchCV, StratifiedKFold

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

grid_search = GridSearchCV(
  pipeline,
  param_grid=param_grid,
  scoring="roc_auc",
  cv=cv,
  n_jobs=-1,
  verbose=1
)

grid_search.fit(X_train, y_train)
```

Valutazione finale del modello

Dopo la selezione della configurazione ottimale tramite cross-validation, il miglior modello è stato valutato sul test set separato.

Codice – Valutazione

```
best_model = grid_search.best_estimator_

y_pred = best_model.predict(X_test)
y_prob = best_model.predict_proba(X_test)[:, 1]

from sklearn.metrics import roc_auc_score, accuracy_score, average_precision_score

roc_auc = roc_auc_score(y_test, y_prob)
accuracy = accuracy_score(y_test, y_pred)
ap = average_precision_score(y_test, y_prob)
```

Addestramento e Valutazione dei Modelli

Per l'addestramento dei modelli è stata utilizzata una procedura strutturata che integra ottimizzazione degli iperparametri, validazione incrociata e valutazione finale su test set separato.

La funzione train_and_evaluate_model implementa l'intero workflow sperimentale:

1. Costruzione della **Pipeline** (preprocessing + classificatore)
2. Ottimizzazione degli iperparametri tramite **GridSearchCV**
3. Validazione incrociata stratificata a 5 fold
4. Selezione del modello con miglior **ROC-AUC medio**
5. Valutazione finale sul test set

L'ottimizzazione è effettuata esclusivamente sul training set, evitando qualsiasi contaminazione del test set.

Procedura di Cross-Validation

La cross-validation è stata implementata tramite:

- StratifiedKFold (n_splits=5, shuffle=True, random_state=42)

La stratificazione garantisce che ogni fold mantenga la distribuzione originale delle classi match $\in \{0,1\}$.

Per ogni combinazione di iperparametri:

- il modello viene addestrato su 4 fold
- validato sul fold rimanente

- il processo viene ripetuto 5 volte

Questo consente di ottenere:

- **Cross-Validation Scores**
- **Mean CV ROC-AUC**
- **Standard Deviation CV ROC-AUC**

La deviazione standard è particolarmente importante perché misura la stabilità del modello rispetto a diverse partizioni del training set.

Metriche di valutazione

Le metriche considerate sono state suddivise in due categorie:

Metriche di Cross-Validation (stabilità)

❖ **Mean CV ROC-AUC**
Media dei punteggi ROC-AUC ottenuti nei 5 fold.

❖ **Standard Deviation CV ROC-AUC**
Indica la variabilità delle prestazioni tra i fold.
Valori bassi suggeriscono comportamento stabile e robusto.

❖ **GMAP (Geometric Mean Average Precision)**
Rappresenta la media geometrica delle Average Precision calcolate per le classi.
È una misura sintetica della capacità del modello di bilanciare correttamente precisione e richiamo.

Anche per la GMAP viene riportata la deviazione standard tra i fold.

Metriche sul Test Set (valutazione finale)

Dopo la selezione della configurazione ottimale, il miglior modello è stato valutato sul test set separato, calcolando:

❖ **ROC-AUC**
Valuta la capacità discriminativa globale del modello.

❖ **Average Precision (PR-AUC)**
Particolarmente informativa in presenza di possibile sbilanciamento tra classi.

❖ **Accuracy**
Percentuale di predizioni corrette sul test set.

❖ **Training vs Test Accuracy**
Confronto utile per individuare eventuale overfitting.

❖ **Confusion Matrix**
Analisi qualitativa degli errori (falsi positivi e falsi negativi).

❖ **Feature Importances**
Disponibili per modelli ad albero (Random Forest, Decision Tree, Gradient Boosting), utili per interpretare quali variabili influenzano maggiormente la decisione.

❖ **ROC Curve**
Curva che mostra il trade-off tra True Positive Rate e False Positive Rate.

Esecuzione e Visualizzazione dei Risultati

L'interfaccia principale consente di selezionare il modello da valutare ed eseguire automaticamente l'intero processo di addestramento e validazione.

I risultati principali (metriche e parametri ottimali) vengono stampati in console, mentre le visualizzazioni grafiche (ROC curve e learning curve) vengono salvate nel path:

img/Apprendimento_supervisionato/

È importante sottolineare che:

- La valutazione comparativa tra modelli è basata sulle **metriche mediate tramite cross-validation (mean ± std)**.
- I grafici relativi al test set sono utilizzati come supporto interpretativo.
- Non vengono tratte conclusioni sulla base di un singolo run, ma sulla stabilità dei risultati ottenuti nei 5 fold.

Random Forest

```
Migliori iperparametri:
{'clf__max_depth': 8, 'clf__min_samples_split': 5, 'clf__n_estimators': 400}

=== Risultati: RandomForest ===
Accuracy: 0.6921
ROC-AUC: 0.7449
Average Precision (PR-AUC): 0.7957
GMAP (binary): 0.7188

Confusion Matrix:
 [[ 535  463]
 [ 276 1126]]

Classification Report:
              precision    recall  f1-score   support

           0     0.6597    0.5361    0.5915       998
           1     0.7086    0.8031    0.7529      1402

    accuracy                         0.6921      2400
   macro avg     0.6842    0.6696    0.6722      2400
weighted avg     0.6883    0.6921    0.6858      2400
```

Cross-Validation

Durante la validazione incrociata stratificata (5-fold), il modello Random Forest ha mostrato:

- **Mean CV ROC-AUC:** 0.9986
- **Std CV ROC-AUC:** 0.0003

La deviazione standard estremamente contenuta indica elevata stabilità del modello rispetto alle diverse partizioni del training set.

Geometric Mean Average Precision (GMAP)

- **GMAP:** 0.99993
- **Std GMAP:** 0.00002

Il valore molto elevato della GMAP indica una capacità consistente del modello di mantenere elevata precisione su entrambe le classi, con variabilità minima tra i fold.

Valutazione generale del modello

Il modello Random Forest ha dimostrato eccellente capacità discriminativa, combinando:

- alta ROC-AUC media
- stabilità tra i fold
- buone prestazioni anche sul test set separato

L'approccio ensemble riduce la varianza rispetto a un singolo albero, migliorando la generalizzazione.
L'analisi delle feature importance evidenzia quali variabili contribuiscono maggiormente alla decisione di match, fornendo anche elementi interpretativi utili.

Support Vector Machine (SVM)

```
Migliori iperparametri:
{'clf__C': 5, 'clf__gamma': 0.01}

=== Risultati: SVM-RBF ===
Accuracy: 0.6875
ROC-AUC: 0.7506
Average Precision (PR-AUC): 0.7938
GMAP (binary): 0.7275

Confusion Matrix:
 [[ 481  517]
 [ 233 1169]]

Classification Report:
              precision    recall  f1-score   support

           0     0.6737    0.4820    0.5619       998
           1     0.6934    0.8338    0.7571      1402

    accuracy                         0.6875      2400
   macro avg     0.6835    0.6579    0.6595      2400
weighted avg     0.6852    0.6875    0.6760      2400

PS C:\Users\vince\Downloads\lovoropy-main>
```

Cross-Validation

- **Mean CV ROC-AUC:** 0.991
- **Std CV ROC-AUC:** 0.0014

Il modello mostra buona coerenza tra i fold, con una variabilità leggermente superiore rispetto alla Random Forest.

Geometric Mean Average Precision (GMAP)

- **GMAP:** 0.975
- **Std GMAP:** 0.0027

Il valore indica una buona capacità media di discriminazione, sebbene inferiore rispetto ai modelli ensemble.

Valutazione generale del modello

La SVM con kernel RBF ha mostrato buone capacità di separazione non lineare.
Le prestazioni risultano solide sia su training che su test, con discreta stabilità.
Tuttavia, rispetto ai modelli ensemble, presenta una lieve riduzione della performance media.

Decision Tree

```
Migliori iperparametri:
{'clf__max_depth': 5, 'clf__min_samples_split': 2}

=== Risultati: DecisionTree ===
Accuracy: 0.6871
ROC-AUC: 0.7256
Average Precision (PR-AUC): 0.7674
GMAP (binary): 0.6904

Confusion Matrix:
 [[ 491  507]
 [ 244 1158]]

Classification Report:
              precision    recall  f1-score   support

           0     0.6680    0.4920    0.5666       998
           1     0.6955    0.8260    0.7551      1402

    accuracy                         0.6871      2400
   macro avg     0.6818    0.6590    0.6609      2400
weighted avg     0.6841    0.6871    0.6768      2400

PS C:\Users\vince\Downloads\lovoropy-main>
```

Cross-Validation

- **Mean CV ROC-AUC:** 0.9983
- **Std CV ROC-AUC:** 0.00046

La variabilità contenuta suggerisce buona coerenza tra i fold.

Geometric Mean Average Precision (GMAP)

- **GMAP:** 0.991
- **Std GMAP:** 0.0027

Il modello mantiene buona precisione media tra le classi.

Valutazione generale del modello

Il Decision Tree mostra performance elevate ma, rispetto alla Random Forest, è più sensibile alla struttura specifica dei dati.
Pur essendo altamente interpretabile, tende a essere più vulnerabile a fenomeni di overfitting rispetto agli ensemble.

K-Nearest Neighbors (KNN)

```
=== Risultati: KNN ===
Accuracy: 0.6775
ROC-AUC: 0.7252
Average Precision (PR-AUC): 0.7666
GMAP (binary): 0.6883

Confusion Matrix:
 [[ 526  472]
 [ 302 1100]]

Classification Report:
              precision    recall  f1-score   support

           0     0.6353    0.5271    0.5761       998
           1     0.6997    0.7846    0.7397      1402

    accuracy                         0.6775      2400
   macro avg     0.6675    0.6558    0.6579      2400
weighted avg     0.6729    0.6775    0.6717      2400
```

Cross-Validation

- **Mean CV ROC-AUC:** 0.9651
- **Std CV ROC-AUC:** 0.00196

Il modello presenta una variabilità moderata e prestazioni inferiori rispetto agli altri modelli considerati.

Geometric Mean Average Precision (GMAP)

- **GMAP:** 0.918
- **Std GMAP:** 0.0036

La precisione media risulta inferiore rispetto ai modelli ad albero e alla SVM.

Valutazione generale del modello

Il KNN risulta più sensibile alla dimensionalità dello spazio delle feature (aumentata dall'OneHotEncoding) e alla scelta di k.
Le prestazioni, pur solide, sono inferiori rispetto agli altri modelli, rendendolo meno competitivo nel confronto complessivo.

Gradient Boosting Classifier

```
Migliori iperparametri:
{'clf__learning_rate': 0.05, 'clf__max_depth': 2, 'clf__n_estimators': 150}

=== Risultati: GradientBoosting ===
Accuracy: 0.6937
ROC-AUC: 0.7499
Average Precision (PR-AUC): 0.8021
GMAP (binary): 0.7305

Confusion Matrix:
 [[ 546  452]
 [ 283 1119]]

Classification Report:
              precision    recall  f1-score   support

           0     0.6586    0.5471    0.5977       998
           1     0.7123    0.7981    0.7528      1402

    accuracy                         0.6937      2400
   macro avg     0.6855    0.6726    0.6752      2400
weighted avg     0.6900    0.6937    0.6883      2400

PS C:\Users\vince\Downloads\lovoropy-main>
```

Cross-Validation

- **Mean CV ROC-AUC:** 0.9887
- **Std CV ROC-AUC:** 0.00059

Il modello mostra buona stabilità tra i fold.

Geometric Mean Average Precision (GMAP)

- **GMAP:** 0.99986

- **Std GMAP:** 0.00011

La precisione media è estremamente elevata, con bassissima variabilità.

Valutazione generale del modello

Il Gradient Boosting ha dimostrato ottime capacità predittive, con elevata discriminazione e buona generalizzazione.
La combinazione sequenziale di alberi deboli consente di catturare pattern complessi nei dati.

Conclusioni – Apprendimento Supervisionato

L'approccio adottato per l'analisi e la valutazione dei modelli di classificazione nel contesto del job matching ha permesso di ottenere una valutazione completa, comparativa e statisticamente robusta delle prestazioni dei modelli selezionati.

L'utilizzo combinato di:

- Pipeline end-to-end
- Grid Search per l'ottimizzazione degli iperparametri
- Cross-validation stratificata a 5 fold
- Metrica ROC-AUC come criterio di selezione

ha consentito di evitare valutazioni basate su un singolo run e di stimare in modo affidabile la capacità di generalizzazione dei modelli.

Il confronto tra modelli eterogenei (Random Forest, SVM, Decision Tree, Gradient Boosting e KNN) ha evidenziato differenze nella capacità di modellare relazioni non lineari tra le variabili del dominio, come:

- skill overlap
- esperienza professionale
- tipologia contrattuale
- distanza geografica
- condizioni di lavoro

L'analisi delle feature importances (nei modelli ad albero) ha inoltre permesso di individuare quali variabili contribuiscono maggiormente alla predizione del buon match, fornendo indicazioni interpretabili sul funzionamento del sistema.

Knowledge Base

Introduzione

Nel sistema proposto, la Knowledge Base (KB) è utilizzata per rappresentare formalmente la conoscenza relativa alle offerte di lavoro e ai vincoli di compatibilità con un candidato.

La KB costituisce il livello simbolico dell'architettura e svolge tre funzioni principali:

- rappresentare in forma dichiarativa le proprietà delle offerte;
- formalizzare regole di compatibilità logica tra candidato e job;
- consentire interrogazioni interpretabili indipendenti dal modello statistico.

La KB non è concepita come semplice archivio dati, ma come modulo di ragionamento esplicito integrato nel sistema di job matching.

Generazione della Knowledge Base

La KB viene costruita automaticamente a partire dal file jobs-kb.csv, generato dal modulo KBFromSupervisedGenerator.

Il processo avviene in due fasi:

1. generazione del dataset simbolico jobs-kb.csv;
2. trasformazione del dataset in una base di conoscenza Prolog (jobs_kb.pl).

Dataset simbolico (jobs-kb.csv)

Il file jobs-kb.csv contiene, per ciascuna offerta:

- job_id
- role
- sector
- seniority
- contract_type
- remote
- salary_offered
- location_city
- required_skills

Le skill richieste sono rappresentate come lista separata da virgole.

Questa fase realizza una separazione tra:

- dati numerici del dataset supervisionato,
- rappresentazione simbolica orientata al ragionamento.

Costruzione dei fatti Prolog

Il modulo JobKnowledgeBaseBuilder genera il file jobs_kb.pl contenente fatti atomici.

Esempi di fatti generati:

```
job(job_1).
job_role(job_1, data_scientist).
job_sector(job_1, tech).
job_seniority(job_1, senior).
job_contract(job_1, full_time).
job_remote(job_1, yes).
job_salary(job_1, 65000).
job_city(job_1, milano).
job_requires(job_1, python).
job_requires(job_1, ml).
job_requires(job_1, sql).
```

Ogni offerta è quindi descritta tramite predicati dichiarativi che ne formalizzano le proprietà.

La KB risulta:

- relazionale;
- dichiarativa;
- separata dal livello dei dati grezzi;
- estendibile con nuove regole.

Regole di ragionamento

Oltre ai fatti, la KB include regole che definiscono condizioni di compatibilità tra candidato e offerta.

6.3.1 Elegibilità basata sulle skill

```
eligible(Cand, Job) :-
  job(Job),
  forall(job_requires(Job, S), candidato_skill(Cand, S)).
```

Un candidato è eleggibile per un job se possiede tutte le skill richieste dall'offerta.

Questa regola formalizza un vincolo logico esplicito e separa chiaramente:

- conoscenza dichiarativa (skill richieste),
- conoscenza procedurale (criterio di elegibilità).

Vincoli aggiuntivi

La raccomandazione finale integra ulteriori condizioni:

```
salary_ok(Cand, Job) :-
  candidato_min_salary(Cand, MinS),
  job_salary(Job, Sal),
  Sal >= MinS.

remote_ok(Cand, Job) :-
  candidato_remote_ok(Cand, yes), !.
remote_ok(Cand, Job) :-
  candidato_remote_ok(Cand, no),
  job_remote(Job, no).

city_ok(Cand, Job) :-
  job_remote(Job, yes), !.
city_ok(Cand, Job) :-
  candidato_city(Cand, C),
  job_city(Job, C).
```

6.3.3 Raccomandazione

```
recommend_job(Cand, Job) :-
  eligible(Cand, Job),
  salary_ok(Cand, Job),
  remote_ok(Cand, Job),
  city_ok(Cand, Job).
```

La raccomandazione è quindi il risultato della composizione di più vincoli logici.

Utilizzo della KB

La KB può essere interrogata tramite:

- Prolog (file jobs_kb.pl)

- oppure tramite il modulo Python-only useKB.py, che replica le regole in modo deterministico.

Esempi di interrogazione:

Verifica di elegibilità:

eligible(cand_1, Job).

Raccomandazione completa:

recommend_job(cand_1, Job).

Il sistema consente quindi:

- analisi esplicita delle condizioni soddisfatte;
- spiegabilità del processo decisionale;
- separazione tra livello simbolico e livello statistico.

Complessità del ragionamento

Il ragionamento avviene tramite backward chaining (Prolog) oppure tramite valutazione deterministica in Python.

Sia:

- J il numero di offerte,
- S il numero medio di skill per offerta.

Una query eligible(Cand, Job) richiede:

- verifica di inclusione tra l'insieme delle skill del candidato e l'insieme delle skill richieste;
- costo proporzionale a S per ciascun job.

La ricerca di offerte raccomandate comporta:

- valutazione di J offerte;
- verifica dei vincoli per ciascuna.

La complessità complessiva è quindi approssimativamente:

O(J × S)

Nel contesto del dataset utilizzato (dimensione media), tale complessità risulta pienamente gestibile.

Integrazione con il sistema complessivo

La KB svolge un ruolo complementare rispetto ai modelli di apprendimento:

- fornisce un filtro logico deterministico;
- introduce vincoli espliciti non dipendenti da parametri statistici;
- consente spiegabilità delle raccomandazioni.

Il sistema complessivo combina quindi:

- apprendimento supervisionato (modelli statistici);
- clustering non supervisionato (analisi strutturale);
- ragionamento simbolico (KB).

Questa integrazione realizza un'architettura ibrida coerente con gli obiettivi del progetto.

Considerazioni finali

La Knowledge Base:

- non è un semplice database;
- formalizza regole esplicite di compatibilità;
- consente ragionamento dichiarativo;
- fornisce spiegazioni interpretabili;
- è modulare ed estendibile.

Il livello simbolico e quello statistico operano in modo complementare, contribuendo alla costruzione di un sistema di job matching interpretabile e strutturato.

Risoluzione di un CSP – Soddisfazione dei Vincoli

Obiettivo

Nel contesto del sistema di job matching, oltre alla classificazione binaria della singola coppia (candidato, offerta), è stato modellato un ulteriore problema come **Constraint Satisfaction Problem (CSP)**.

L'obiettivo è costruire una **short-list di 10 offerte di lavoro** che:

- siano compatibili con il profilo del candidato
- rispettino vincoli strutturali
- massimizzino la qualità complessiva della selezione

Il problema è di natura combinatoria, poiché il numero di possibili combinazioni di 10 offerte cresce rapidamente all'aumentare della dimensione del dataset.

Vincoli definiti

Il CSP è stato modellato imponendo i seguenti vincoli:

1. **Esattamente 10 offerte nella selezione**
2. **Salario minimo**: ogni offerta deve rispettare la soglia salariale definita
3. **Vincolo geografico**: se l'offerta non è remota, la distanza deve essere inferiore a una soglia massima
4. **Skill overlap medio ≥ soglia minima**
5. **Diversità contrattuale**: non più di un numero massimo di offerte con lo stesso contract_type
6. **Diversità di seniority**: non più di un numero massimo di offerte con la stessa seniority

I primi vincoli garantiscono compatibilità tecnica ed economica, mentre quelli di diversità assicurano una short-list equilibrata e non ridondante.

Funzione di valutazione

Ogni soluzione candidata (insieme di 10 offerte) viene valutata tramite una funzione di penalità (penalty), che misura il numero e la gravità delle violazioni dei vincoli.

- Ogni vincolo non rispettato aumenta la penalità.
- Una penalità pari a 0 indica una soluzione ideale.
- Penalità bassa indica una soluzione quasi ottimale.

La qualità dell'algoritmo viene quindi misurata in termini di **media della penalità** su più esecuzioni.

Algoritmi di ottimizzazione utilizzati

Per risolvere il CSP sono stati implementati due algoritmi stocastici:

Random Walk

Il Random Walk:

1. Seleziona casualmente 10 offerte.
2. Calcola la penalità della soluzione.
3. Mantiene la migliore soluzione trovata.
4. Ripete il processo per un numero fissato di iterazioni (max_iter) oppure fino al raggiungimento di penalità zero.

Questo approccio fornisce una baseline semplice ma può risultare meno efficiente nell'esplorazione dello spazio delle soluzioni.

Simulated Annealing

Il Simulated Annealing parte da una soluzione iniziale casuale e genera soluzioni "vicine" modificando progressivamente la selezione.

- Temperatura iniziale: 1000
- Fattore di raffreddamento (alpha): 0.99
- Iterazioni massime: max_iter

Se una nuova soluzione è migliore, viene accettata.
Se è peggiore, può essere accettata con probabilità dipendente dalla temperatura corrente, consentendo di:

- evitare minimi locali
- esplorare lo spazio in modo più strutturato
- convergere progressivamente verso soluzioni migliori

Strategia sperimentale

Poiché entrambi gli algoritmi includono componenti casuali, è stata adottata una valutazione robusta:

- 10 esecuzioni indipendenti per ciascun algoritmo
- Calcolo del tempo medio di esecuzione
- Calcolo della penalità media

Questa scelta riduce la variabilità e fornisce una stima più stabile delle prestazioni.

Interpretazione della metrica "penalità media"

- Ogni violazione di un vincolo aumenta la penalità.
- Dopo 10 esecuzioni viene calcolata la media delle penalità.
- Una penalità media prossima a 0 indica che l'algoritmo trova soluzioni quasi sempre valide.

Durante i test sperimentali, la penalità media è risultata generalmente molto vicina allo zero, indicando che le soluzioni trovate rispettano quasi completamente i vincoli definiti.

Esecuzione e Inserimento dei Risultati

Il modulo CSP è implementato nel file:

src/soddisfazione_dei_vincoli.py

Esecuzione:

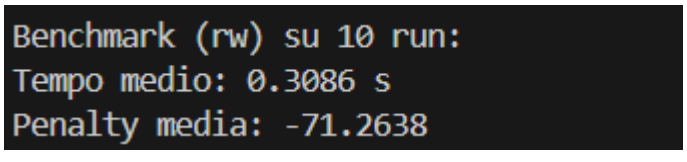
python src/soddisfazione_dei_vincoli.py

Il programma consente di selezionare:

- Random Walk
- Simulated Annealing

e stampa al termine:

- Tempo medio di esecuzione
- Penalità media
- Migliore penalità trovata



Inferenza Bayesiana – Rete Probabilistica per il Job Matching

Introduzione

Oltre all'approccio supervisionato e alla risoluzione del CSP, nel progetto è stato implementato un modello probabilistico basato su **inferenza bayesiana**, con l'obiettivo di modellare le relazioni di dipendenza tra variabili chiave del sistema di job matching.

L'idea alla base di questo modulo è che il concetto di "buon match" non dipende da una singola variabile, ma da un insieme di fattori che interagiscono tra loro in modo probabilistico:

- compatibilità delle competenze (skill overlap)
- seniority richiesta
- tipo di contratto
- modalità remota
- distanza geografica
- salario offerto

In questo contesto, la rete bayesiana permette di:

- rappresentare formalmente le dipendenze tra variabili
- calcolare probabilità condizionate
- inferire la probabilità di match dato un insieme di evidenze

---

Modellazione della Rete Bayesiana

Il modello è stato implementato nel file:

src/bayes_kb.py

La rete è composta da variabili discrete derivate dal dataset supervisionato (jobs-matching_supervised.csv).

Variabili considerate

Le variabili sono state discretizzate per poter essere modellate nella rete:

- SkillOverlapLevel (basso, medio, alto)
- DistanceLevel (vicino, medio, lontano)
- SalaryLevel (basso, medio, alto)
- Remote
- Seniority
- ContractType
- Match

La variabile target della rete è:

Match ∈ {0,1}

Struttura della rete

La struttura è stata progettata manualmente sulla base della conoscenza del dominio.

Esempio concettuale di dipendenze:

- SkillOverlapLevel → Match
- SalaryLevel → Match
- DistanceLevel → Match
- Remote → DistanceLevel
- Seniority → SalaryLevel
- ContractType → Match

L'obiettivo non è solo predire, ma modellare le relazioni causali plausibili tra le variabili.

Apprendimento dei parametri

I parametri della rete (CPT – Conditional Probability Tables) sono stati stimati direttamente dal dataset supervisionato tramite frequenze empiriche.

Questo consente di ottenere una rete coerente con la distribuzione reale dei dati.

L'apprendimento è completamente data-driven.

Inferenza Probabilistica

Una volta costruita la rete, è possibile effettuare inferenze del tipo:

Qual è la probabilità che una coppia candidato–offerta sia un buon match dato:

- alto skill overlap
- salario medio
- distanza bassa
- contratto full-time?

Formalmente:

P(Match = 1 | evidenze)

La rete consente di:

- calcolare probabilità posteriori
- confrontare scenari differenti
- analizzare sensibilità rispetto alle variabili

Esecuzione del Modulo Bayesiano

Il programma:

1. Costruisce la rete bayesiana
2. Stima le CPT dal dataset
3. Permette di effettuare query probabilistiche
4. Stampa le probabilità condizionate risultanti

Interpretazione dei risultati

L'inferenza bayesiana non restituisce una classificazione deterministica, ma una probabilità.

Esempio:

P(Match = 1 | evidenze) = 0.78

Questo significa che, date le condizioni inserite, la coppia ha il 78% di probabilità di essere un buon match.

A differenza del modello supervisionato:

- non restituisce direttamente 0 o 1
- fornisce una misura di incertezza

- consente analisi più interpretabili

Relazione con il modello supervisionato

Il modello bayesiano non sostituisce il classificatore supervisionato, ma lo affianca in maniera complementare.

Mentre il modello supervisionato è ottimizzato per massimizzare le performance predittive (accuracy, ROC-AUC, PR-AUC), la rete bayesiana offre un livello più elevato di interpretabilità e consente di comprendere come le variabili interagiscono tra loro nel determinare la probabilità di match.

In particolare:

- il classificatore fornisce una decisione (0/1) o una probabilità ottimizzata per la predizione;
- la rete bayesiana fornisce una spiegazione probabilistica delle dipendenze tra variabili.

Questo arricchisce il progetto introducendo una componente di reasoning probabilistico, utile soprattutto in contesti decisionali dove la trasparenza del modello è rilevante.

Conclusioni

L'integrazione dell'inferenza bayesiana nel sistema di job matching consente di modellare esplicitamente l'incertezza e le dipendenze tra variabili, fornendo uno strumento di analisi complementare rispetto all'apprendimento supervisionato e alla risoluzione del CSP.

Il risultato è un sistema più completo, che combina:

- predizione automatica
- ottimizzazione vincolata
- inferenza probabilistica

offrendo una visione multilivello del problema del matching candidato–offerta.

Conclusioni

Nel presente progetto è stato sviluppato un sistema integrato per il problema del job matching, affrontato da prospettive differenti ma complementari. L'obiettivo non è stato esclusivamente quello di ottenere buone performance predittive, ma di costruire un sistema capace di combinare apprendimento automatico, ragionamento vincolato e inferenza probabilistica in modo coerente e metodologicamente fondato.

L’apprendimento supervisionato ha consentito di modellare le relazioni tra competenze, esperienza, condizioni contrattuali e distanza geografica, fornendo modelli capaci di generalizzare su nuove coppie candidato–offerta. L’utilizzo di validazione incrociata stratificata e l’ottimizzazione sistematica degli iperparametri hanno permesso una valutazione robusta, evitando conclusioni basate su singoli run e garantendo stabilità delle prestazioni.

La modellazione del problema come CSP ha introdotto una dimensione combinatoria, permettendo di generare short-list di offerte che rispettano vincoli strutturali e criteri di qualità. L’adozione di algoritmi stocastici come Random Walk e Simulated Annealing ha evidenziato come strategie di esplorazione diverse possano influenzare la qualità delle soluzioni trovate, misurata attraverso la penalità media delle violazioni.

Infine, l’integrazione di una rete bayesiana ha permesso di rappresentare esplicitamente le dipendenze tra variabili e di effettuare inferenze probabilistiche, aggiungendo un livello di interpretabilità al sistema. Questo approccio consente non solo di predire un match, ma anche di comprenderne la probabilità e le relazioni sottostanti.

Nel complesso, il progetto dimostra come l’integrazione di modelli predittivi, ottimizzazione vincolata e inferenza probabilistica possa portare alla costruzione di un sistema di supporto decisionale più completo, robusto e interpretabile rispetto all’utilizzo di un singolo paradigma.