IBM Attrition Prediction

Eleonora Ghizzota, 685939
Ingegneria della Conoscenza, A.A 2021-2022
Università degli Studi di Bari "Aldo Moro", Informatica

1 Introduzione

IBM L'*International Business Machines Corporation*, costituita nel 1911, ma attiva già dal 1888, è un'azienda statunitense veterana nel settore informatico, attiva in 170 paesi. Produce e commercializza hardware e software, offrendo infrastrutture, servizi di hosting, cloud computing, IA, quantum computing e consulenza nel settore informatico e strategico; per 27 anni consecutivi, fino al 2020, ha detenuto inoltre il record per il numero di brevetti statunitensi generati da un'azienda, rendendosi un'organizzazione di grande rilevanza anche nel campo della ricerca scientifica. Tra le sue invenzioni più diffuse abbiamo il primo PC con sistema operativo MS-DOS, il disco floppy, il database relazionale, la scheda madre e il codice a barre.

Risorse umane L'espressione risorse umane, coniata da Raymond Miles negli anni '60, indica il personale che presta la propria attività lavorativa in ente con il proprio *capitale umano*. Con l'introduzione delle nuove teorie che si allontanavano dal taylorismo, i dipendenti si elevarono da beni strumentali a risorse e le aziende iniziarono ad *investire* sui dipendenti per ottimizzarne le prestazioni e la qualità. Per ogni grande azienda, come anche IBM, è di vitale importanza il ruolo dell'addetto alle *risorse umane*, colui che gestisce il personale impiegato. I suoi compiti sono quelli di selezionare persone che siano capaci di fare la differenza nelle aree di cui è composta l'azienda, reclutare risorse talentuose e farne emergere le qualità, assumere figure con propensioni specifiche e inserirle nella giusta area dell'azienda, applicare politiche retributive mirate al fine di favorire la meritocrazia.

Attrition Traducibile come *tasso di abbandono*, l'attrition è uno dei rischi maggiori per ogni azienda ove il personale qualificato è la sua risorsa fondamentale. Riuscire a prevedere il tasso di abbandono di un impiegato aiuterebbe il dipartimento delle risorse umane a prendere scelte più oculate, al fine di ridurre lo spreco di risorse, cercando di investire su impiegati con un basso tasso di abbandono.

Obiettivo Pensato come uno strumento che possa aiutare gli addetti alle risorse umane, l'obiettivo di questo lavoro è di costruire una *rete bayesiana* capace di effettuare predizioni probabilistiche sul tasso di abbandono sia di candidati, sia di impiegati, a seconda dei dati che si hanno a disposizione. Tali predizioni dovrebbero aiutare gli addetti alle risorse umane a selezionare personale su cui investire a lungo termine. Si dà inoltre la possibilità di interrogare una *base di conoscenza* per ottenere informazioni e statistiche utili, al fine di monitorare l'ambiente di lavoro.

2 Dataset

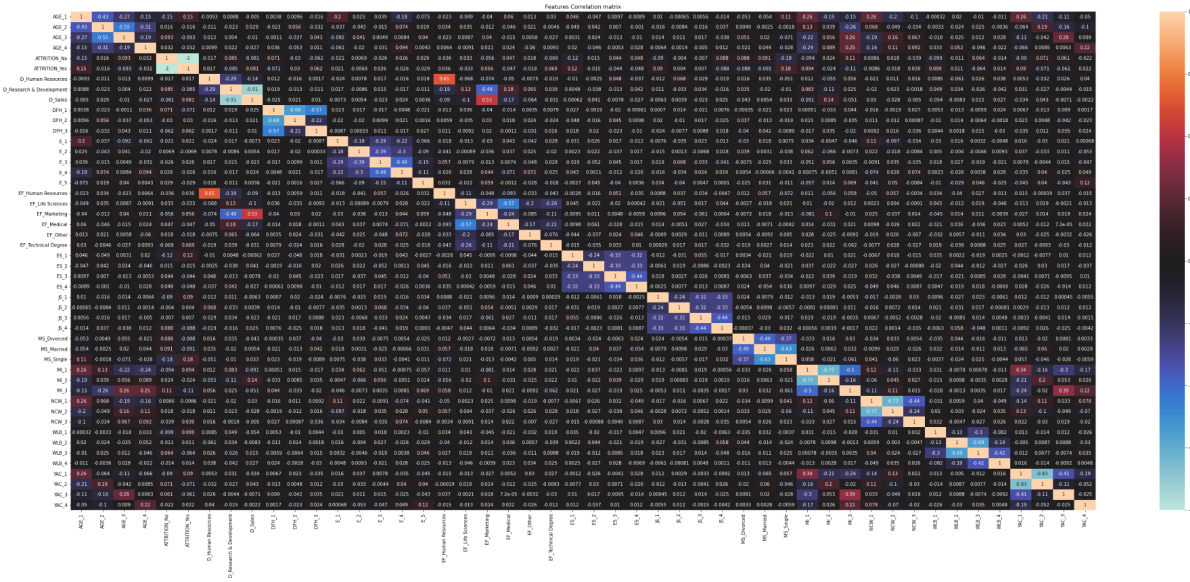
Il dataset utilizzato, contenente dati raccolti da Watson Analytics, è kaggle.com/yasserh/ibm-attrition-dataset, composto da 13 feature:

- `Age: [18, 60]`, età dell'impiegato
- `Attrition: ['Yes, 'No']`, stato di abbandono dell'impiegato
- `Department: ['Research & Development', 'Sales', 'Human Resources']`, dipartimento in cui lavora l'impiegato
- `DistanceFromHome: [1, 29]`, distanza tra l'abitazione e il posto di lavoro
- `Education: [1: Below College, 2: College, 3: Bachelor, 4: Master, 5: Doctor]`, livello di istruzione dell'impiegato
- `EducationField: ['Life Sciences', 'Medical', 'Human Resources', 'Technical Degree', 'Marketing', 'Other']`, campo di studi in cui è specializzato l'impiegato
- `EnvironmentSatisfaction: [1: Low, 2: Medium, 3: High, 4: Very High]`, livello di soddisfazione rispetto all'ambiente di lavoro
- `JobSatisfaction: [1: Low, 2: Medium, 3: High, 4: Very High]`, livello di soddisfazione rispetto al proprio lavoro
- `MaritalStatus: ['Single', 'Married', 'Divorced']`
- `MonthlyIncome: [1.009, 20.000]`, stipendio mensile
- `NumCompaniesWorked: [0, 9]`, numero di compagnie per cui l'impiegato ha precedentemente lavorato
- `WorkLifeBalance: [1: Bad, 2: Good, 3: Better, 4: Best]`, bilancio tra vita privata e lavoro
- `YearsAtCompany: [0, 40]`, numero di anni in cui l'impiegato ha lavorato per IBM

I valori di feature il cui dominio è ampio sono stati raggruppati in *fasce*:

- `Age: [1: 18-28, 2: 29-39, 3: 40-50, 4: 51-101]`
- `DistanceFromHome: [1: 1-9, 2: 10-19, 3: 20-28, 4: 29+]`
- `MonthlyIncome: [1: 1.000-6.999, 2: 7.000-13.999, 3: 14.000-18.999, 4: 19.000+]`
- `NumCompaniesWorked: [1: 0-2, 2: 3-5, 3: 6-8, 4: 9+]`
- `YearsAtCompany: [1: 0-9, 2: 10-19, 3: 20-29, 4: 30-39, 5: 40-101]`

Successivamente, al fine di una computazione della matrice di correlazione di Pearson più efficiente, è stata applicata la *one hot binary encoding*, un metodo che permette la conversione dei valori categorici in nuove colonne a cui vengono assegnati dei valori binari, 1 o 0.
Per ogni feature categorica si avrà quindi un vettore di valori binari con un solo valore posto a 1, quello assunto dalla feature per quel particolare record.



Matrice di correlazione

La matrice mostra la correlazione tra due variabili per tutte le possibili coppie di valori. Esistono diversi modi per calcolare il valore di correlazione, il più popolare e utilizzato in questo caso è il *coefficiente di correlazione di Pearson*. Questo misura esclusivamente una correlazione lineare tra due variabili, mentre non è possibile rilevare correlazioni non lineari. Il valore dell'indice di correlazione di Pearson assume valori tra +1 (*perfetta correlazione lineare positiva*) e -1 (*perfetta correlazione lineare negativa*), o indica *assenza* di correlazione lineare, quindi le variabili sono indipendenti.
Dalla matrice si evince correlazione tra:

- `EducationField` e `Department`
- `Age` con `MonthlyIncome` e `YearsAtCompany`
- `YearsAtCompany` e `MonthlyIncome`

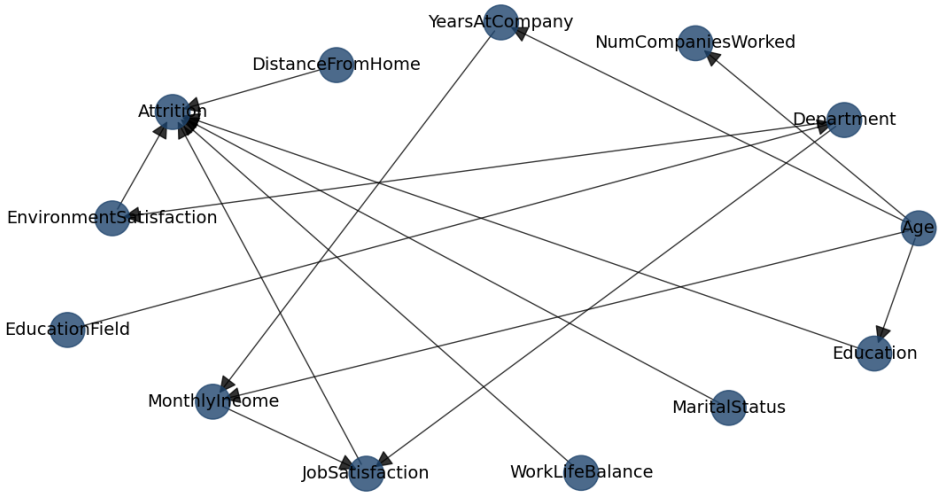
Il dataset in formato one hot encoding non viene più utilizzato in seguito.

3 Rete bayesiana

Una rete bayesiana, o *belief network*, è un modello grafico probabilistico. Data una variabile X, solamente alcune altre variabili influenzano in maniera diretta il suo valore; ciò significa che X è condizionatamente indipendente dalle variabili restanti. Le variabili che influenzano direttamente X sono dette *parents(X)*. Una rete bayesiana dunque sfrutta l'ordinamento delle variabili fornito dall'indipendenza condizionata per costruire un grafo orientato aciclico (*Directed Acyclic Graph*). Per adattare una rete bayesiana a un particolare dominio e per fare inferenza su di essa è necessario definire:

- **Variabili rilevanti**: ogni feature osservata è una variabile rilevante. Nel nostro caso, tutte le 13 feature sono state considerata variabili rilevanti;

- **Relazioni** tra le variabili: la probabilità che assumono i valori di `Attrition`, `yes` e `no`, è influenzata direttamente da `EnvironmentSatisfaction`, `JobSatisfaction`, `WorkLifeBalance`, `MaritalStatus`, `Education` e `DistanceFromHome`. Di seguito il grafo completo delle relazioni, formalizzato in seguito all'osservazione dei dati:




```
[('EducationField', 'Department'),
       ('Department', 'EnvironmentSatisfaction'),
       ('Department', 'JobSatisfaction'),
       ('Age', 'NumCompaniesWorked'),
       ('Age', 'MonthlyIncome'),
       ('Age', 'YearsAtCompany'),
       ('Age', 'Education'),
       ('YearsAtCompany', 'MonthlyIncome'),
```

```
('MonthlyIncome', 'JobSatisfaction'),
('EnvironmentSatisfaction', 'Attrition'),
('JobSatisfaction', 'Attrition'),
('WorkLifeBalance', 'Attrition'),
('MaritalStatus', 'Attrition'),
('DistanceFromHome', 'Attrition'),
('Education', 'Attrition')]
```

La struttura illustrata viene in seguito confrontata con tre strutture apprese tramite *HillClimb search*, un approccio di ricerca euristica utile quando sono coinvolti molti nodi, e tre funzioni di score, *BIC*, *K2* e *BDeu*. HillClimb search implementa una ricerca greedy locale che, partendo da un DAG disconnesso, manipola iterativamente i singoli archi al fine di massimizzare lo score. La ricerca termina quando viene trovato il massimo locale. La ricerca HillClimb rientra nella categoria di *score-based structure learning*, composta principalmente da:

- l'algoritmo di ricerca, che attraversa lo spazio di ricerca di tutti i possibili DAG per trovare la soluzione migliore (HillClimb search, Exhaustive search, etc.)
- la funzione di score, che indica quanto la rete bayesiana si adatta ai dati (Bayesian Dirichlet scores: BDeu, K2, Bayesian Information Criterion: BIC, etc.)

```
# Score-based structure learning with hillclimb search
print('[Structure learning with hillclimb search, BIC score...]')
self.bic_model = bnlearn.structure_learning.fit(self.df, methodtype='hc', scoretype='bic', verbose=1)
print('[Structure learning with hillclimb search, K2 score...]')
self.k2_model = bnlearn.structure_learning.fit(self.df, methodtype='hc', scoretype='k2', verbose=1)
print('[Structure learning with hillclimb search, BDeu score...]')
self.bdeu_model = bnlearn.structure_learning.fit(self.df, methodtype='hc', scoretype='bdeu', verbose=1)
```

All'avvio del programma vengono mostrati in console i valori di *accuracy*, *precision*, *recall, F-score* ed *error* del DAG proposto rispetto ai tre modelli appresi con HillClimb search. Nonostante i valori di queste metriche varino leggermente di volta in volta, il DAG realizzato offre complessivamente una buona performance.

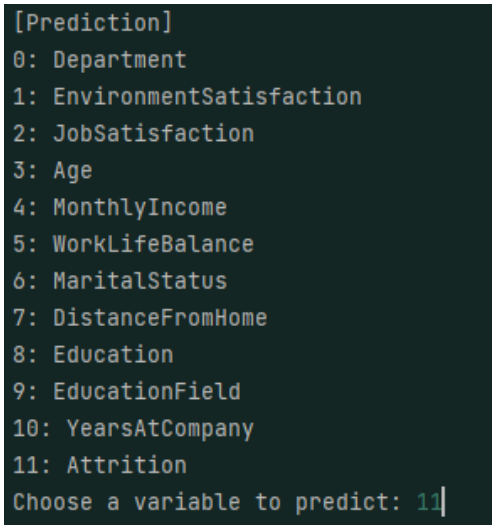
- **Dipendenza** della distribuzione di una variabile dai genitori: a ogni variabile è associata una *tabella delle distribuzioni condizionate* (*CPT*). La funzione `bnlearn.parameter_learning.fit`, prendendo in input il grafo delle relazioni, si occupa dell'apprendimento dei parametri, costruendo di fatto le CPT associate ai nodi del grafo. Esistono diversi metodi per la costruzione e popolazione delle tabelle di distribuzioni condizionate; per questo progetto è stata scelta la *Maximum Likelihood Estimation*, che consiste nel massimizzare la funzione di verosimiglianza.

Le CPT vengono costruite di volta in volta, acquisendo i dati necessari direttamente dal dataset: questo permette di avere tabelle sempre aggiornate, soprattutto nel caso in cui dovessero aggiungersi nuove osservazioni al dataset.

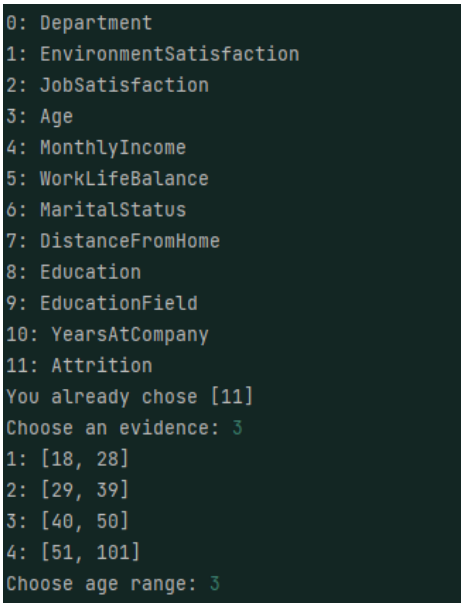
Nella rete bayesiana proposta l'algoritmo utilizzato per fare inferenza probabilistica è l'algoritmo di *eliminazione delle variabili*, che rientra nella categoria dell'inferenza **esatta**: le probabilità sono dunque calcolate precisamente. Il metodo più semplice per fare inferenza esatta è enumerare tutti i mondi coerenti in base all'evidenza disponibile, ma sfruttando la struttura della rete si possono ottenere risultati migliori, ad esempio applicando l'algoritmo VE, in grado di trovare la distribuzione a posteriori di una variabile in una belief network.

Esempio di utilizzo:

1. L'utente seleziona una feature su cui effettuare la predizione



2. L'utente sceglie quali sono le feature osservate e i rispettivi valori



3. Il sistema restituisce la predizione

```
Done choosing evidence(s)? y-ny
Chosen evidence(s):
{'Age': 3}
Finding Elimination Order: : 100%|██████████| 10/10 [00:00<00:00, 9965.08it/s]
Eliminating: DistanceFromHome: 100%|██████████| 10/10 [00:01<00:00,  7.17it/s]
[{'Attrition': 0, 'p': 0.7367722724723736}, {'Attrition': 1, 'p': 0.26322772752762646}]
```

Il modulo `bayesian_network.py` contiene tutto il necessario per fare inferenza sulla rete bayesiana proposta, realizzata con la libreria python bnlearn. Il sistema permette di selezionare più osservazioni, ma non impedisce di selezionare la medesima osservazione più volte nel corso della stessa query: se dovesse verificarsi questo caso, il sistema riporterebbe errore.

4 Base di conoscenza

Il sistema include la possibilità di interrogare una base di conoscenza. Il modulo `knowledge_base.py` contiene gli assiomi, scritti utilizzando le funzioni messe a disposizione da `dataset.py`: queste elaborano ogni volta i dati a disposizione nel dataset, permettendo all'utente di lavorare con una KB sempre aggiornata. Se al dataset si dovessero aggiungere nuove tuple non sarà necessario scrivere nuovi assiomi che coinvolgano i nuovi dati. Le query disponibili permettono sia di verificare se una clausola è conseguenza logica della KB, in tal caso rispondendo `yes` o `no`, sia permettono di comprendere variabili in modo che la risposta riporti i valori che la variabile può assumere in base all'input.
Gli assiomi presenti nella KB sono i seguenti, per ognuno è proposto un esempio di utilizzo:

- `suitable_department(EducationField,Department)`
  Fornendo uno specifico valore di `EducationField`, restituisce i valori di `Department` in cui è possibile lavorare con la specializzazione di input
  - `suitable_department(human resources,Q)`
    `{'Q' : 'humanresources'}`
- `working_field(Department,EducationField)`
  Fornendo uno specifico valore di `Department`, restituisce i valori di `EducationField` degli impiegati che lavorano nel suddetto dipartimento
  - `working_field(sales,Q)`
    `{'Q': 'lifesciences'}, {'Q': 'marketing'},`
    `{'Q': 'technicaldegree'}, {'Q': 'medical'},`
    `{'Q': 'other'}`
- `age_average_income(Age,MonthlyIncome)`
  Fornendo un valore di `Age` ([18+]), restituisce il valore medio di `MonthlyIncome` per quella età
  - `age_average_income(45,Q)`
    `{'45': '7811.219512195122'}`
- `department_marital_status(Department,MaritalStatus,%)`
  Fornendo un valore di `Department` e un valore di `MaritalStatus`, restituisce la percentuale di impiegati presenti nel dipartimento il cui stato corrisponde a quello specificato
  - `department_marital_status(sales,single,Q)`
    `{'Q': '34.30493273542601'}`
- `average_environment_satisfaction(Department,avg_sat)`
  Fornendo un valore di `Department`, restituisce la media di `EnvironmentSatisfaction` per il dipartimento specificato
  - `average_environment_satisfaction(sales,Q)`
    `{'Q': '2.679372197309417'}`

- `average_job_satisfaction(Department,avg_sat)`
  Fornendo un valore di `Department`, restituisce la media di `JobSatisfaction` per il dipartimento specificato
  - `average_job_satisfaction(human resources,Q)`
    `{'Q': '2.6031746031746033'}`
- `average_job_attrition(Department,Attrition,avg_sat)`
  Fornendo un valore di `Department` e di `Attrition` restituisce la media di `JobSatisfaction`
  - `average_job_attrition(sales,yes,Q)`
    `{'Q': '2.5217391304347827'}`
    `average_job_attrition(sales,no,Q)`
    `{'Q': '2.810734463276836'}`
- `department_attrition_percentage(Department,Attrition,%)`
  Fornendo un valore di `Department` e di `Attrition` restituisce la percentuale di tasso di abbandono
  - `department_attrition_percentage(sales,yes,Q)`
    `{'Q': '20.62780269058296'}`

Esempio di utilizzo:

1. L'utente sceglie una query

```
0: suitable_department
1: working_field
2: age_average_income
3: department_marital_status
4: average_environment_satisfaction
5: average_job_satisfaction
6: average_job_attrition
7: department_attrition_percentage
Choose a query: 0
```

2. L'utente seleziona i valori degli argomenti

```
0: life sciences
1: medical
2: human resources
3: technical degree
4: other
5: marketing
6: Q
Choose an education field 1
0: human resources
1: sales
2: research & development
3: Q
Choose a department: 3
```

3. Il sistema restituisce la risposta

```
[{'Q': 'research&development'}, {'Q': 'humanresources'}, {'Q': 'sales'}]
```

Questa KB è stata realizzata con pytholog, una libreria che permette di utilizzare la programmazione logica in python.