Predizione della Soddisfazione Lavorativa dei Dipendenti

Gruppo di lavoro

- Angelo Gadaleta, 649800, a.gadaleta21@studenti.uniba.it a.gadaleta21@phd.uniba.it

https://github.com/gadda-angy-inf-bari/Ing-conoscenza-Employee_Survey

AA 2023-24

Introduzione

In un momento storico in cui c'è concorrenza, sfruttamento e precarietà nell'ambito lavorativo, il dipendente si accontenta del proprio lavoro vivendo dalle 8 alle 12 ore all'interno delle stesse mura, a tal punto da dover combattere per poter avere un briciolo di tempo per se stessi. La mancanza di tempo per se stessi e di riposo è tale da non permettere un risultato efficiente nel campo lavorativo, e l'accumulo di richieste può portare a stress, scatti d'ira ed emozioni forti. La conseguenza di ciò comporta a non saper rapportarsi con i propri colleghi di lavoro, creando ambienti disfunzionali.

Ad oggi nell'Accordo quadro europeo del 2004, lo stress lavoro-correlato (SLC) viene definito come "una condizione che può essere accompagnata da disturbi o disfunzioni di natura fisica, psicologica o sociale a tal punto che i singoli dipendenti non si sentono in grado di corrispondere alle richieste o alle aspettative date dai propri datori". Data la situazione critica, Italia, il vigente quadro normativo, costituito dal Decreto Legislativo n. 81 del 2008, stabilisce l'obbligo per il datore di lavoro di valutare e gestire il rischio stress lavoro-correlato al pari di tutti gli altri rischi per la salute e sicurezza. E nel 2010 la Commissione consultiva permanente per la salute e la sicurezza sul lavoro ha elaborato le indicazioni necessarie alla valutazione del rischio stress lavoro-correlato individuando un percorso obbligatorio.

OBBIETTIVO: In tal proposito, ho cercato di portare tutti i dati e le ricerche nel progetto d'esame, il quale pone l'obiettivo di costruire dei modelli che possano prevedere la soddisfazione lavorativa dei dipendenti a partire da una serie di caratteristiche come l'equilibrio vita-lavoro, livello di stress, l'ambiente di lavoro, il numero di ore di straordinari e altri feature. Con questi dati si può prendere decisioni aziendali per migliorare il benessere dei dipendenti e migliorare la produzione aziendale, o eventualmente può essere utilizzata come arma a doppio taglio per affondare l'azienda.

Requisiti Funzionali

Per la realizzazione del progetto ho deciso di utilizzare il linguaggio di programmazione Python in quando offre a disposizione delle librerie utili per realizzare un sistema di apprendimento in modo semplice e intuitivo.

Le librerie utilizzate sono:

bnlearn - pacchetto per la scoperta causale imparando la struttura grafica delle reti bayesian, l'apprendimento dei parametri, l'inferenza e i metodi di campionamento.

matplotlib.pyplot – fornisce degli strumenti per poter visualizzare graficamente vari dati (curve di apprendimento, grafici a barre, grafici a torta)

networkx - visualizza graficamente il grafo (usato per osservare la struttura della rete bayesiana) ma è anche utile per elaborare le principali caratteristiche del grafo.

numpy - consente di lavorare con vettori e matrici in maniera più efficiente e veloce di quanto non si possa fare con le liste e le liste di liste (matrici).

pandas - libreria per la manipolazione di dati in formato sequenziale o tabellare, quali serie temporali o dati di microarray. In questo caso manipola un dataset in formato "csv".

seaborn – una libreria di visualizzazione dei dati per Python che fornisce un'interfaccia ad alto livello per la creazione di grafici statistici attraenti e informativi.

DATASET: https://www.kaggle.com/datasets/lainguyn123/employee-survey Nell'analisi del dataset sono stati identificati i dipendenti in base alle nozioni base quali al genere, età, stato civile, e sulle nozioni utili per sviluppare il progetto, quali livello di lavoro(junior, medio, senior), carico, numero di ore.

Le feature sono 23 tutte incentrate sul dipendente:

- **EmpID:** identificatore univoco
- **Gender:** genere [maschio, femmina, altro]
- **Age:** età del dipendente [22, 60]
- **MaritalStatus:** stato civile [single, sposato, divorziato, vedovo]
- **JobLevel:** livello di lavoro [Intern/Fresher, junior, medio, senior, lead]
- **Experience:** numero di anni di esperienza lavorativa acquisita [0, 29]
- **Dept:** dipartimento in cui lavora [IT, HR, finanza, marketing, vendite, legale, operazioni, servizio, clienti]
- **EmpType:** tipo di occupazione [tempo pieno, part-time, contratto]
- **WLB**: valutazione del rapporto tra vita-lavoro [1, 5]
- **WorkEnv:** valutazione dell'ambiente di lavoro [1, 5]
- **PhysicalActivityHours**: numero di ore di attività fisica a settimana [0, 5]
- **Workload:** valutazione del carico di lavoro [1, 5]
- **Stress:** valutazione del livello di stress [1, 5]
- **SleepHours:** numero di ore di sonno per notte [4, 10]
- **CommuteMode:** modo di spostarsi [auto, trasporto pubblico, bicicletta, passeggiata, moto]
- **CommuteDistance:** distanza percorse durante il tragitto in chilometri [1, 29]
- **NumCompanies:** numero di diverse aziende per cui ha lavorato [0, 12]
- **TeamSize:** dimensioni della squadra di cui fa parte [5, 30]
- **NumReports**: numero di persone segnalate, ma applicabile per i livelli senior e piombo [0, 9]
- **EduLevel:** il più alto livello di istruzione raggiunto [scuola, superiori, laurea triennale, master, PhD]
- **haveOT:** informa sui suoi straordinari [vero/falso]
- **TrainingHoursPerYear:** numero di ore di formazione ricevute all'anno [10, 64.5]
- **JobSatisfation:** valutazione della soddisfazione lavorativa [1, 5]

Sommario

Paragrafo sul KBS e su come integri moduli che dimostrino competenze sui diversi argomenti (specificati sotto)

Elenco argomenti di interesse

- **Paragrafo su argomento 1 :** Ragionamento logico del primo ordine
- **Paragrafo su argomento 2 :** Apprendimento supervisionato, modelli base e modelli composti)
- **Paragrafo su argomento 3 :** Apprendimento e incertezza, apprendimento probabilistico e non supervisionato
- **Paragrafo su argomento 4 :** Apprendimento e incertezza, apprendimento di Belief Network

Main e Preprocessing del Dataset

Il progetto di ingegneria della conoscenza su 'Employee Survey' è stato attuato dal file 'main.py', il quale verrà eseguito quando gli verrà chiesto di scegliere uno dei moduli possibili, nei quali, ognuno eseguirà un pezzo di argomento presente negli argomenti citati precedentemente.

Qualsiasi scelta si effettuasse, ogni modulo farà accesso ad dataset 'employee_survey', come fonte per costruire un dataset interno su cui effettuare le analisi, elaborare i dati e restituire un risultato.

Prima di poter lavorare sul dataset, all'interno di 'preprocessing_data.py' il dataset viene pulito rimuovendo colonne inutili (es. EmpID inutile hai nostri scopi) e vengono effettuali eliminazioni di transizioni vuote o con dati mancanti.

Alcuni moduli hanno effettuato altre pulizie sul dataset convertendo dei tipi di dati in altri per poter essere elaborati all'interno di algoritmi (es. in alberi decisionali), utilizzando funzioni come LabelEncoder, OneHotEncoder, StandardScaler appartenenti alla libreria sklearn.

Non è stata approfondita la strategie di pulizia o di selezione delle feature o altro sul dataset tale da poter migliorarlo in quanto appartenente al 'Feature Engineering' che non viene trattata.

Il 'main.py' richiama le librerie riferenti agli argomenti citati precedentemente secondo l'ordine definito: knowledge_base_logic.py, supervised_learning.py, probability_learning.py, unsupervised_learning.py, bayesian_network.py

Ragionamento Logico

Il sistema dà la possibilità di interrogare la base di conoscenza chiamata 'employee_survey' il quale è stata utilizzando la libreria pytholog all'interno della libreria knowledge_base_logic.py.

Nel modulo knowledge_base_logic.py sono stati scritti degli assiomi i questi elaborano ogni volta i dati a disposizione nel dataset, permettendo all'utente di lavorare con una KB sempre aggiornata.

Infatti un cambio del dataset con aggiunte o eliminazioni di transazioni non comporterebbero la necessità di riscrivere gli assiomi che coinvolgano i nuovi dati.

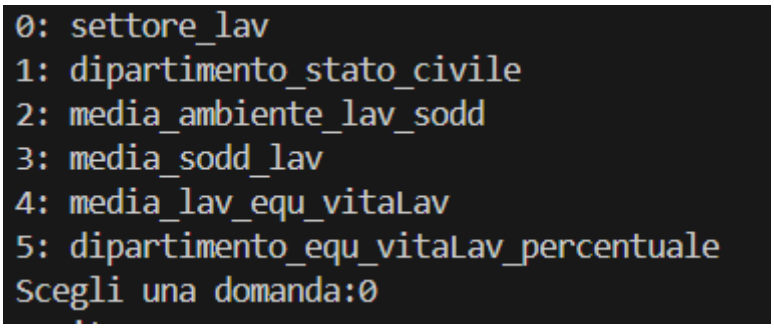
Le query disponibili permettono, attraverso interrogazioni, di avere informazioni relative al dipendente e il suo rapporto con il suo lavoro, riportando i valori che la variabile può assumere in base all'input.

Gli assiomi presenti nella KB sono stati proposti degli esempi per una maggior comprensione di utilizzo:

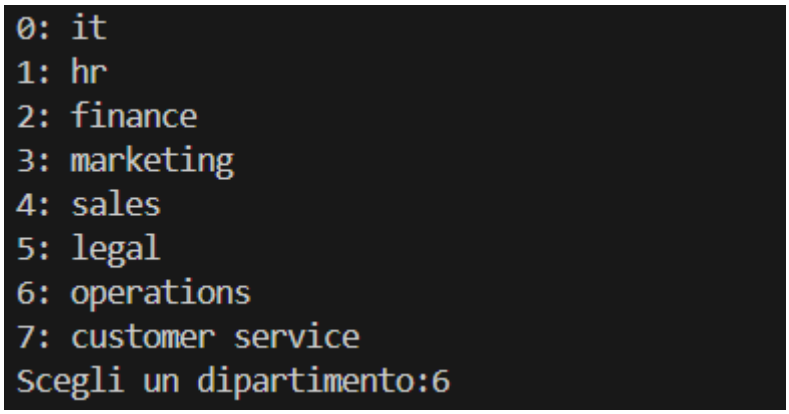
- settore_lav(Dept, EduLevel)
  Fornendo uno specifico valore di Dept, restituisce i valori di EduLevel dei dipendenti che lavorano con una qualifica scolastica in quel dipartimento
  - settore_lav(marketing, Q)
    {'Q': 'bachelor'}, {'Q': 'highschool'},
    {'Q': 'master'}, {'Q': 'phd'}
- dipartimento_stato_civile(Dept, MaritalStatus, %)
  Fornendo un valore di Dept e un valore di MaritalStatus, restituisce la percentuale di impiegati presenti nel dipartimento il cui stato corrisponde a quello specificato
  - dipartimento_stato_civile(sales, single, Q)
    {'Q': '48.754448398576514'}
- media_ambiente_lav_sodd(Dept, avg_sat)
  Fornendo un valore di Department, restituisce la media di EnvironmentSatisfaction per il dipartimento specificato
  - media_ambiente_lav_sodd(sales, Q)
    {'Q': '3.0498220640569396'}
- media_sodd_lav(Dept, avg_sat)
  Fornendo un valore di Dept, restituisce la media di JobSatisfaction per il dipartimento specificato
  - media_sodd_lav(operations, Q)
    {'Q': '3.465934065934066'}
- media_lav_equ_vitaLav(Dept, WLB, avg_sat)
  Fornendo un valore di Dept e di WLB restituisce la media di JobSatisfaction
  - media_lav_equ_vitaLav(sales, 4, Q)
    {'Q': '2.5217391304347827'}
    average_job_wlb(legal, 4, Q)
    {'4': '3.4285714285714284'}
- dipartimento_equ_vitaLav_percentuale(Dept, WLB, %)
  Fornendo un valore di Dept e di wlb restituisce la percentuale di tasso di abbandono
  - dipartimento_equ_vitaLav_percentuale(legal, 4, Q)
    {'4': '20.588235294117645'}

Esempio di utilizzo:

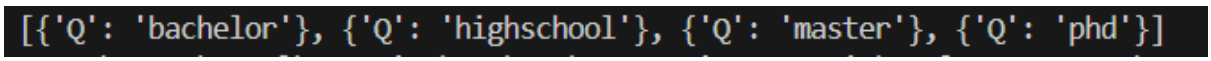
1. L’utente sceglie una query



2. L’utente seleziona i valori degli argomenti



3. Il sistema restituisce la risposta



Apprendimento Supervisionato

Il modello, prendendo in considerazione l'input e dal target dello stress preso dal dataset, analizza l'intensità dello stimolo dato, grazie a utilizzi di esempi di stress su lavoro, identificandolo in un minimo di punteggio 1 e un massimo punteggio di 5. Esempio nel caso in cui un dipendete avesse una quantità di sonno inferiore alla media consigliata, questo comporterebbe a un punteggio massimo di stress, quindi un livello alto d'intensità.

Il progetto è stato sviluppato inizialmente utilizzando la metrica K Fold Crossing Validation, la quale si rende valida nel momento in cui il modello non va in sovradattamento, e quindi non avvenga il overfitting. Dopo ciò, viene utilizzato l'apprendimento supervisionato su quattro modelli: albero di decisione, regressione logistica, random forest e rete neurale. Inseguito vengono impegnate le metriche accuratezza, precision, recall, f1 score, le quali mi identificano quale tra i modelli sia il migliore.

I modelli di apprendimento supervisionato (o anche detti automatico) sono:

- **Albero di decisione:** Classificatore strutturato ad albero in cui la radice e i nodi interno rappresentano delle condizioni sulle feature di input, mentre le foglie descrivono le classi di appartenenza o le probabilità di appartenenza a tali classi. Seguendo il percorso passando tra le condizioni, si arriverà alla classe di appartenenza.
- **Regressione logistica:** Classificatore simile alla regressione lineare ma viene utilizzata per calcolare le probabilità di appartenenza di un esempio a ogni classe. Per ogni classe si ha una funzione lineare di cui apprende i pesi mediante la tecnica della discesa di gradiente. In essa è presente un peso per ogni feature più un termine noto w0. Il risultato viene "schiacciato" da una funzione nota come softmax, la quale trasforma un vettore di valori reali in una distribuzione di probabilità su più classi.
- **Random Forest:** Classificatore composto che si ottiene creando tanti alberi di decisione. Il valore di output si ottiene mediando sulle predizioni di ogni albero appartenente alla foresta, detta anche tecnica di bagging. Esso si pone come soluzione che minimizza l'overfitting del training set rispetto agli alberi di decisione.
- **Rete neurale:** Classificatore composto che si ottiene implementando una funzione di predizione dati le feature di input $F(x) = Fn(Fn-1(..(F2(F1(X)))..))$ dove ogni funzione Fi mappa un vettore di valori in un vettore di valori e dove Fi è lo strato i-esimo. Il numero di funzioni composte n, è la profondità della rete neurale. L'ultimo strato Fn è lo strato di output. Gli altri strati sono chiamati strati nascosti mentre l'input X è chiamato strato di input. A differenza degli altri modelli, qui si conosce l'input e l'output ma non si conosce lo strato intermedio o nascosto.

Per prima cosa si sono scelti gli iper-parametri da inserire, i quali sono i parametri di un modello di apprendimento supervisionato nel quale non vengono appresi durante la fase di addestramento come i normali parametri del modello, ma devono essere necessariamente fissati prima che il modello possa cominciare l'addestramento. Ogni modello di classificazione prevede la presenza di parametri opportunamente passati in fase di costruzione di un determinato modello. Se non esplicitati, ai parametri verranno associati valori di default, che molto spesso non permettono al modello di esaltare la sua massima accuratezza.

Un iper-parametro, invece, viene calcolato utilizzando la tecnica di K-fold Cross Validation (CV). La sua funzione è quella di prendere un dataset, dividerlo in k fold, nel quale l'insieme delle k sono disgiunti, e il modello viene addestrato k volte. Per ogni iterazione 1 fold viene usato il testing mentre gli altri k-1 fold vengono utilizzati per il training. La K-fold cross validation viene applicato prima della fase di testing vero e proprio per verificare che il modello addestrato sui dati di training sia ottimale. In questo caso per ricercare gli iper-parametri dei modelli si è utilizzato il GridSearch con CrossValidation nel quale vengono specificate le griglie dei valori per gli iper-parametri e si cerca tutte le combinazioni possibili nella ricerca della miglior combinazione possibile.

Qui mostro i modelli utilizzati e gli iperparametri scelti per ogni modello.

```
models = {
    "DecisionTree": DecisionTreeClassifier(random_state=42),
    'Logistic Regression': LogisticRegression(random_state=42),
    "RandomForest": RandomForestClassifier(random_state=42),
    'Neural Network': MLPClassifier(random_state=42)
}
```

```
param_grids = {
    "DecisionTree": {
        "criterion": ['gini', 'entropy', 'log_loss'],
        "max_depth": [2, 3, 4, 5, 10, 20],
        "min_samples_split": [2, 3, 4, 5, 10, 20],
    },
    'Logistic Regression': {
        'penalty': ['l1', 'l2', 'elasticnet', 'none'],
        'C': [0.01, 0.1, 1, 10, 100, 1000],
        'max_iter': [100, 200, 500, 1000],
    },
    "RandomForest": {
        "n_estimators": [10, 50, 100, 200, 500, 1000],
        "max_depth": [1, 2, 5, 10, 20],
        "criterion": ['gini', 'entropy'],
    },
    'Neural Network': {
        'hidden_layer_sizes': [(50,), (100,), (50, 50), (100, 50)],
        'learning_rate_init': [0.001, 0.01],
        'max_iter': [200, 300, 500]
    }
}
```

Per l'**albero di decisione**:

- Criterion: Misura la qualità dello spit effettuato sui nodi. Può assumere i seguenti valori: gini, entropy e log_loss
- Max_depth: Indica l'altezza massima dell'albero

- Min_samples_split: Il numero minimo di esempi necessari affinché possa essere inserito un criterio di split. Se il numero è minore viene innestata una foglia

Per la **Regressione logistica**:

- Penalty: Stabilisce la penalizzazione applicata al modello, per ridurne la complessità e cercare di evitare *overfitting*
- C: Specifica quanto "forte" è la regolarizzazione. Più il valore di C è basso e più forte è la regolarizzazione. Valori di C piccoli prevengono *overfitting* ma rendono il modello più rigido. Valori alti rendono il modello più flessibile ma sono soggetti a *overfitting*
- Max_iter: Indica il numero di iterazioni utilizzate

Per l'**Foresta casuale** utilizzo gli stessi iperparametri dell'albero di decisione, in più aggiungo:

- n_estimators: il numero di alberi nella foresta

Per il **Neural network**:

- Hidden_layer_sizer: Dimensione e numero di strati nascosti
- Learning_rate_init: Tipo di apprendimento
- Max_iter: Indica il numero massimo di iterazioni

Nel mio caso i parametri che sono stati restituiti sono:

```
Migliori iperparametri per DecisionTree: {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 2}
Miglior score: 0.6045454545454546
Migliori iperparametri per Logistic Regression: {'C': 10, 'max_iter': 100, 'penalty': 'l2'}
Miglior score: 0.6082644628099174
Migliori iperparametri per RandomForest: {'criterion': 'entropy', 'max_depth': 10, 'n_estimators': 100}
Miglior score: 0.6057851239669422
Migliori iperparametri per Neural Network: {'hidden_layer_sizes': (50,), 'learning_rate_init': 0.001, 'max_iter': 300}
Miglior score: 0.5987603305785123
```

Successivamente si è eseguito la fase di validazione utilizzando solamente il training. Il training viene splittato in training e in validazione, questa divisione permetterà di testare i modelli iper-parametrizzati utilizzando i dati di training come apprendimento e la validazione come strumento di valutazione del modello. Ovviamente per individuare un buon modello, questa fase viene ripetuta più volte, splittando più volte il training originale in due parti (training e validation) e ad ogni iterata si avrà un training o validation diverso dal precedente. Con questo metodo permette di testare su tutti i tipi diversi di training e validation per individuare un miglior modello.

```python
# LOOP DI CROSS-VALIDAZIONE
for train_idx, val_idx, in kf.split(X, y):
    X_tr = X.iloc[train_idx]
    y_tr = y.iloc[train_idx]

    X_val = X.iloc[val_idx]
    y_val = y.iloc[val_idx]

    # addestrare un modello
    best_model.fit(X_tr, y_tr)
    # eseguire la predizione e salvare in una lista
    pred = best_model.predict(X_val)
```

Fuori dal ciclo del cross-validation e aver eseguito la predizione del modello su più fold durante la validazione incrociata, utilizzo i dati risultanti per calcolare l'**AUC (Area Under the Curve)** e la **deviazione standard**. L'AUC fornisce una misura della capacità discriminante del modello, ovvero la sua abilità nel distinguere correttamente tra le classi. Questo è calcolato considerando le probabilità predette dal modello. La deviazione standard, invece, misura la consistenza di questa capacità discriminante (o di altre metriche, come l'accuratezza) su più valutazioni, evidenziando la stabilità del modello.

In questo modo, è possibile ottenere una valutazione completa che combina sia la qualità delle predizioni (AUC) sia la loro stabilità (std_dev), utilizzando i dati derivati da ciascun fold del processo di validazione incrociata.

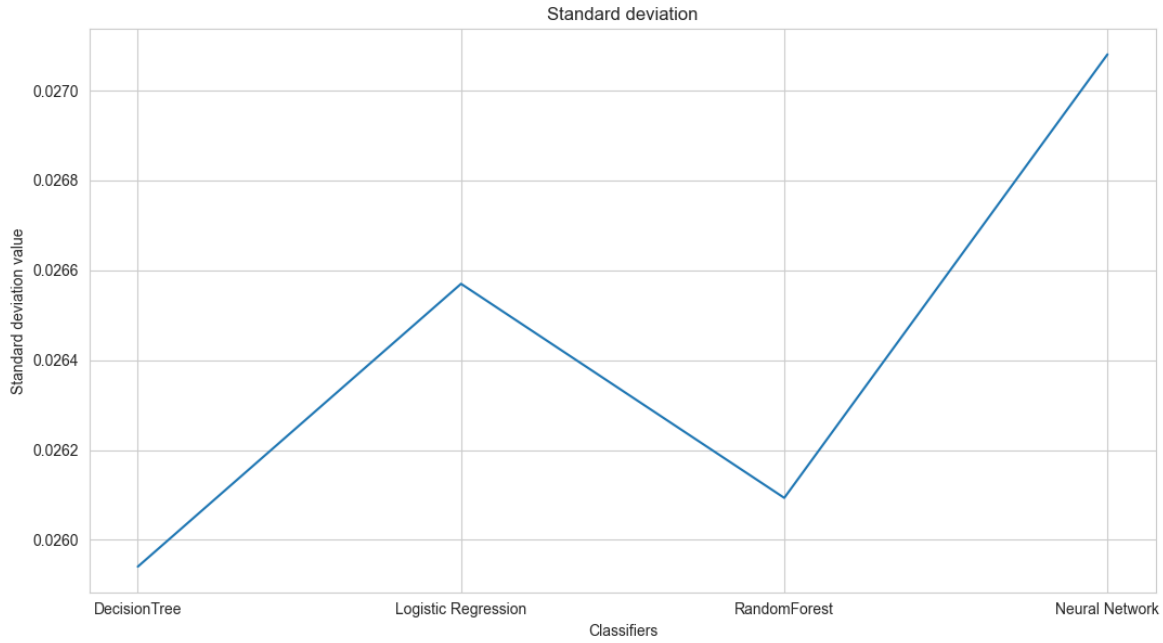
```python
general_auc_score = np.mean(aucs)
print(f'\nOur out of fold AUC score is {general_auc_score:0.4f}')

# calcolare la deviazione standard dei punteggi ottenuti
std_dev[model_name] = np.std(scores)

print(f"La deviazione standard di {model_name} dei punteggi è: {std_dev[model_name]}")
```

Otteniamo i risultati:

```
Our out of fold AUC score is 0.6090
La deviazione standard di DecisionTree dei punteggi è: 0.025940460642082646
Our out of fold AUC score is 0.5995
La deviazione standard di Logistic Regression dei punteggi è: 0.026569959967471003
Our out of fold AUC score is 0.5938
La deviazione standard di RandomForest dei punteggi è: 0.026093475457735597
Our out of fold AUC score is 0.5912
La deviazione standard di Neural Network dei punteggi è: 0.027080678935322 44
```



I risultati delle AUC calcolate durante la validazione incrociata evidenziano il **Decision Tree** ha ottenuto il valore più alto dimostrando una capacità leggermente migliore nel distinguere tra le classi rispetto agli altri modelli. La **Logistic Regression** ha una performance comparabile ma leggermente inferiore; i modelli **Random Forest** e **Neural Network** suggerendo la difficoltà a discriminare efficacemente tra le classi. Quindi il **Decision Tree** si comporta marginalmente meglio in termini di capacità discriminante per questo dataset.

I valori della deviazione standard calcolati durante la validazione incrociata informano che il **Decision Tree** risulta il più stabile, mentre il **Neural Network** è il più variabile in termini di performance.

Alla fine si è voluto valutare i modelli mettendoli a confronto tra loro per capire quale algoritmo riesce meglio ad apprendere passandogli il dataset.

Queste metriche forniscono una visione complessiva delle prestazioni del modello, consentendo di scegliere quella più adeguata in base agli obiettivi specifici del problema.

Per effettuare un maggior confronto tra i modelli, sono stati utilizzate le metriche accuratezza, precisione, rumore e f1_score, utilizzate per valutare modelli:

- L'accuratezza è la proporzione di previsioni corrette rispetto al numero totale di campioni.
- La precisione misura la percentuale di campioni classificati positivamente dal modello che sono effettivamente positivi.
- Il recall rappresenta la capacità del modello di identificare correttamente tutti i campioni positivi.
- L'F1-score è la media armonica di precision e recall, offrendo un bilanciamento tra le due metriche. È utile quando si desidera considerare sia i falsi positivi che i falsi negativi.

Queste metriche evidenziano diverse prospettive sulle performance del modello e vengono spesso utilizzate insieme per una valutazione completa, specialmente in contesti di classificazione binaria o multiclasse.

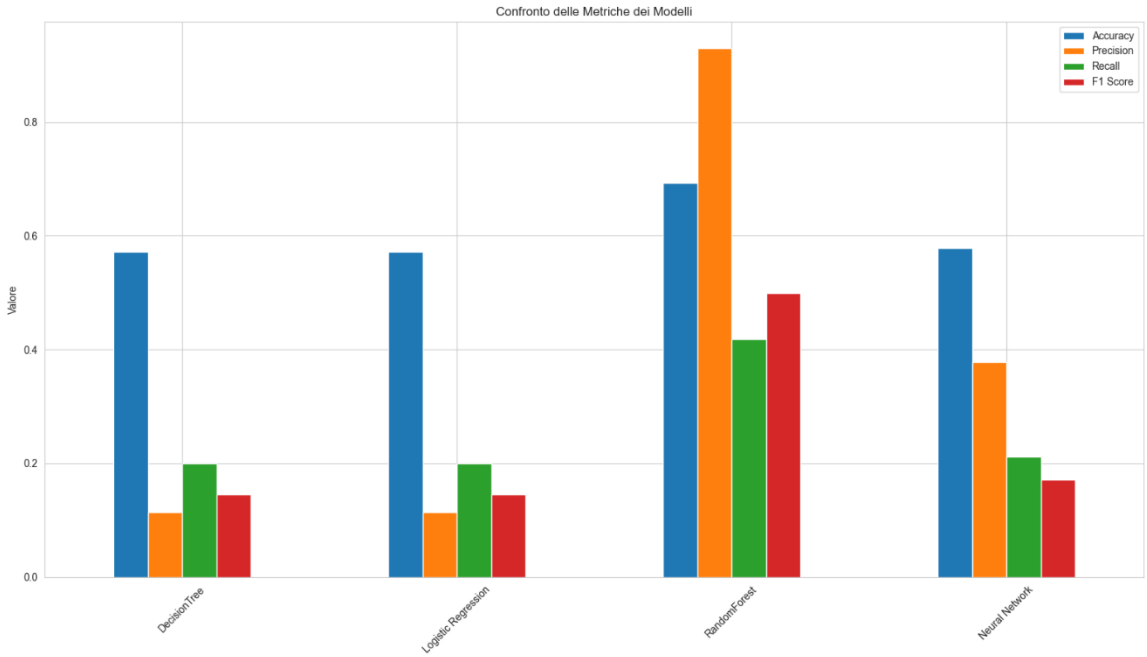
```
Model: DecisionTree | Test Accuracy: 0.571900826446281
                      Test precision: 0.11438016528925621
                        Test recall: 0.2
                            Test f1_score: 0.1455310199789695
```

```
Model: Logistic Regression | Test Accuracy: 0.571900826446281
                      Test precision: 0.11438016528925621
                        Test recall: 0.2
                            Test f1_score: 0.1455310199789695
```

```
Model: RandomForest | Test Accuracy: 0.6925619834710743
                      Test precision: 0.9300751879699248
                        Test recall: 0.4186731802373241
                            Test f1_score: 0.4998290602860763
```

```
Model: Neural Network | Test Accuracy: 0.5785123966942148
                     Test precision: 0.37814863102998697
                       Test recall: 0.21219631936593206
                           Test f1_score: 0.17074905770557947
```

Per avere una visione grafica, ho messo a confronto i modelli utilizzando i risulta avuti dalle metriche appena viste.



Si osserva che il random forest risulta il migliore nell'accuratezza, nella precision, nel recall e nel f1-score, mentre la rete neurale risulta migliore come modello rispetto ai restanti modelli nella precision, nel recall e nel f1-score. Riguardo l'accuratezza, i tre modelli se la contendono alla pari.

Tuttavia, ci sono diverse ragioni per cui un modello come il random forest risulta funzionare rispetto ad altri modelli come la rete neurale per varie ragioni: overfitting, semplicità dei dati, ottimizzazione dei parametri, interpretabilità, rumore dei dati, capacità di gestire variabili reali e discrete/categoriche.

```
Best Model: RandomForest with Test Accuracy: 0.6925619834710743
```

Apprendimento Probabilistico

L'apprendimento probabilistico utilizza modelli matematici basati sulle distribuzioni di probabilità per rappresentare i dati e fare previsioni.

Un sottoinsieme dell'apprendimento probabilistico è l'apprendimento bayesiano che si basa esplicitamente sul **Teorema di Bayes**. A differenza dell'apprendimento probabilistico classico, l'approccio bayesiano consente di trattare l'incertezza e integrare le conoscenze a priori con le evidenze dei dati osservati.

Tra i vari modelli bayesiani disponibili, ho scelto il classico **Naive Bayes**, che assume che le caratteristiche siano condizionatamente indipendenti, semplificando così i calcoli e rendendo l'algoritmo estremamente efficiente e particolarmente adatto ai problemi di classificazione.

Come per l'apprendimento supervisionato, per prima cosa si sono scelti gli iper-parametri da inserire: var_smoothing controlla la quantità di varianza aggiunta ai dati per garantire la stabilità numerica durante i calcoli.

```
param_grid = {
    'var_smoothing': [1e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4]
}
```

Nel mio caso i parametri che sono stati restituiti sono:

```
Miglior iperparametro: {'var_smoothing': 0.0001}
```

Successivamente ho eseguito il CV per individuare il miglior modello sul Naive Bayes iper-parametrizzato.

```
for train_index, val_index in kf.split(X_train):
    X_ktrain, X_kval = X_train.iloc[train_index], X_train.iloc[val_index]
    y_ktrain, y_kval = y_train.iloc[train_index], y_train.iloc[val_index]

    # Addestramento del modello sul fold corrente
    best_model.fit(X_ktrain, y_ktrain)

    # Predizione sul validation set
    y_kpred = best_model.predict(X_kval)

    # Accumula predizioni e vere etichette
    y_pred_cv.extend(y_kpred)
    y_true_cv.extend(y_kval)

    # Calcolo delle metriche
    cv_accuracies.append(accuracy_score(y_kval, y_kpred))
    cv_precisions.append(precision_score(y_kval, y_kpred, average='weighted'))
    cv_recalls.append(recall_score(y_kval, y_kpred, average='weighted'))
    cv_f1s.append(f1_score(y_kval, y_kpred, average='weighted'))
```

In fine ho calcolato le metriche perché forniscono una visione complessiva delle prestazioni del modello, consentendo di scegliere quella più adeguata in base agli obiettivi specifici del problema.

```
Metriche finali sul test set:
Test Accuracy: 0.50
Test precision: 0.44
Test recall: 0.50
Test f1_score: 0.43
```

Confrontando i valori delle metriche di valutazione del modello NB con i modelli dell'apprendimento supervisionato, abbiamo che il random forest continua a essere migliore con una accuratezza di 0.69

Apprendimento non supervisionato

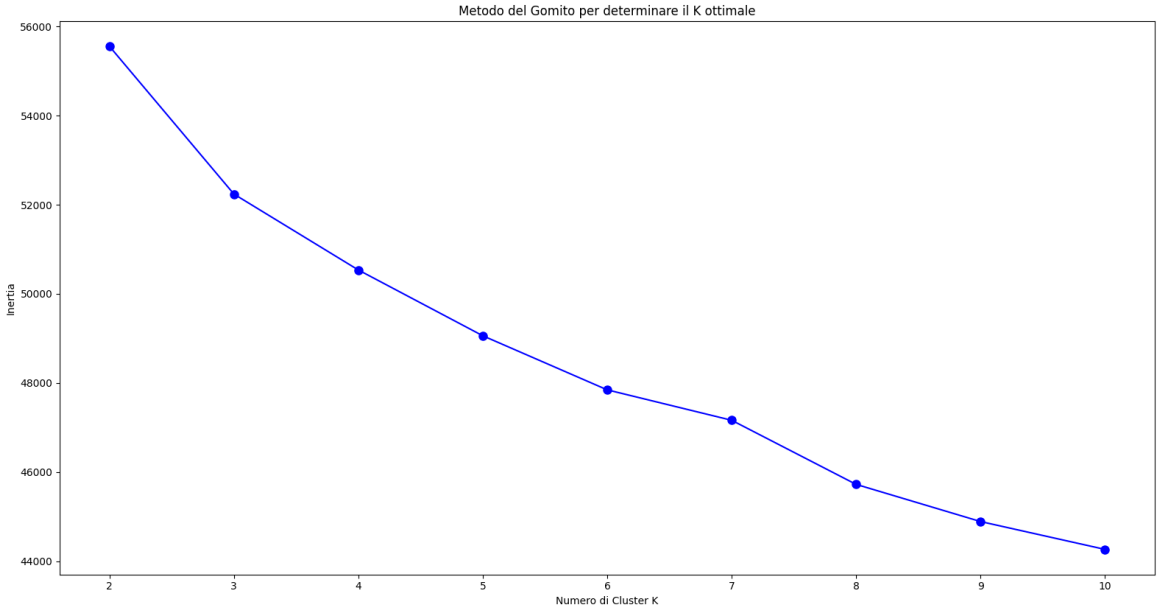
L'apprendimento non supervisionato è una branca d'apprendimento nel quale si richiede di imparare a riconoscere schemi nell'input senza alcuna indicazione specifica dei valori in uscita.

Clustering: L'obbiettivo è raggruppare gli elementi del dataset in base a delle similitudini.

Nel mio caso, l'apprendimento non supervisionato è stato utilizzato nel raggruppamento delle classi generali che descrivono il dipendente, quali, il genere, età, occupazione, quantità di lavoro da svolgere ecc.

Nel mio caso ho deciso di utilizzare l'algoritmo di hard clustering KMeans e soft clustering EM. Il KMeans ha come problema principale trovare il numero ideale di cluster da fornire in input all'algoritmo. Per risolvere questo problema ho seguito un metodo definito come "curva del gomito".

Applicando l'Elbow Method, "Metodo del Gomito", ho scoperto il numero di cluster più adatto per il dataset. Esso è stato scelto poiché è un modo totalmente oggettivo per determinare il numero di cluster. Esso mostra la somma delle distanze quadrate tra ogni punto dati e il centro del cluster assegnato al variare del numero di cluster, rilevando il punto in cui la diminuzione del calcolo diventa meno significativa.



Nel mio caso possiamo vedere come il numero ottimale di cluster è k = 3, dunque successivamente è stato eseguito l'algoritmo K-Means con 3 cluster.

Poi ho fatto stampare l'insieme dei valori delle feature del centroide di ogni cluster in K-means, nel quale ogni valore numerico corrisponde a un valore in stringa, precedente modificato:

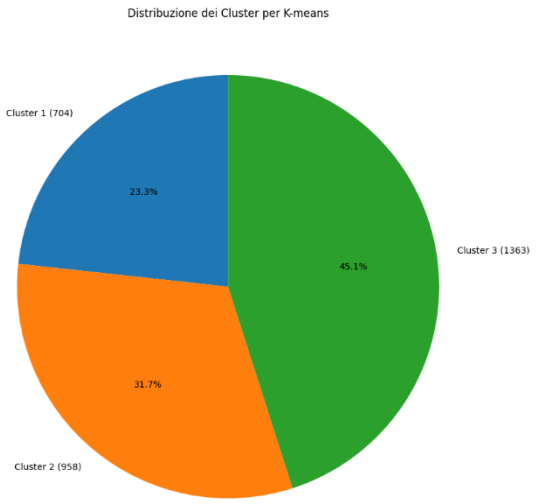
```
Centroidi di K-means (valori delle feature per ogni centroide):
Cluster 1 - Elemento più vicino al centroide: Indice 2174, Valori: [1, 48, 1, 4, 22, 2, 1, 3, 4, 1.5, 5, 2, 6.9, 1, 21, 8, 18, 4, 2, False, 51.0, 4]
Cluster 2 - Elemento più vicino al centroide: Indice 909, Valori: [1, 22, 2, 1, 0, 5, 1, 4, 3, 2.4, 3, 1, 6.6, 3, 19, 0, 10, 0, 0, False, 20.0, 3]
Cluster 3 - Elemento più vicino al centroide: Indice 2240, Valori: [1, 36, 1, 4, 14, 6, 1, 4, 2, 2.4, 3, 2, 7.1, 1, 16, 4, 23, 2, 0, False, 47.0, 4]
```

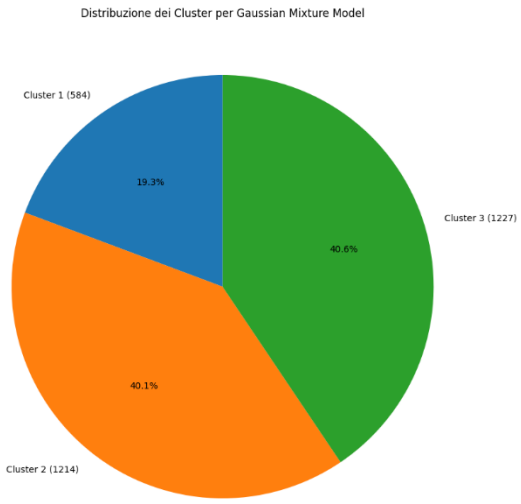
Il soft clustering, invece, è un algoritmo per apprendere modelli probabilistici con variabili latenti:

- problemi di *soft clustering* risolti combinandolo con un classificatore NB,

-analogo a k-Means, ma esempi assegnati ai cluster probabilisticamente stessi argomenti in input: esempi e numero di cluster k.

Alla fine ho fatto stampare la distribuzione dei cluster sui due modelli.





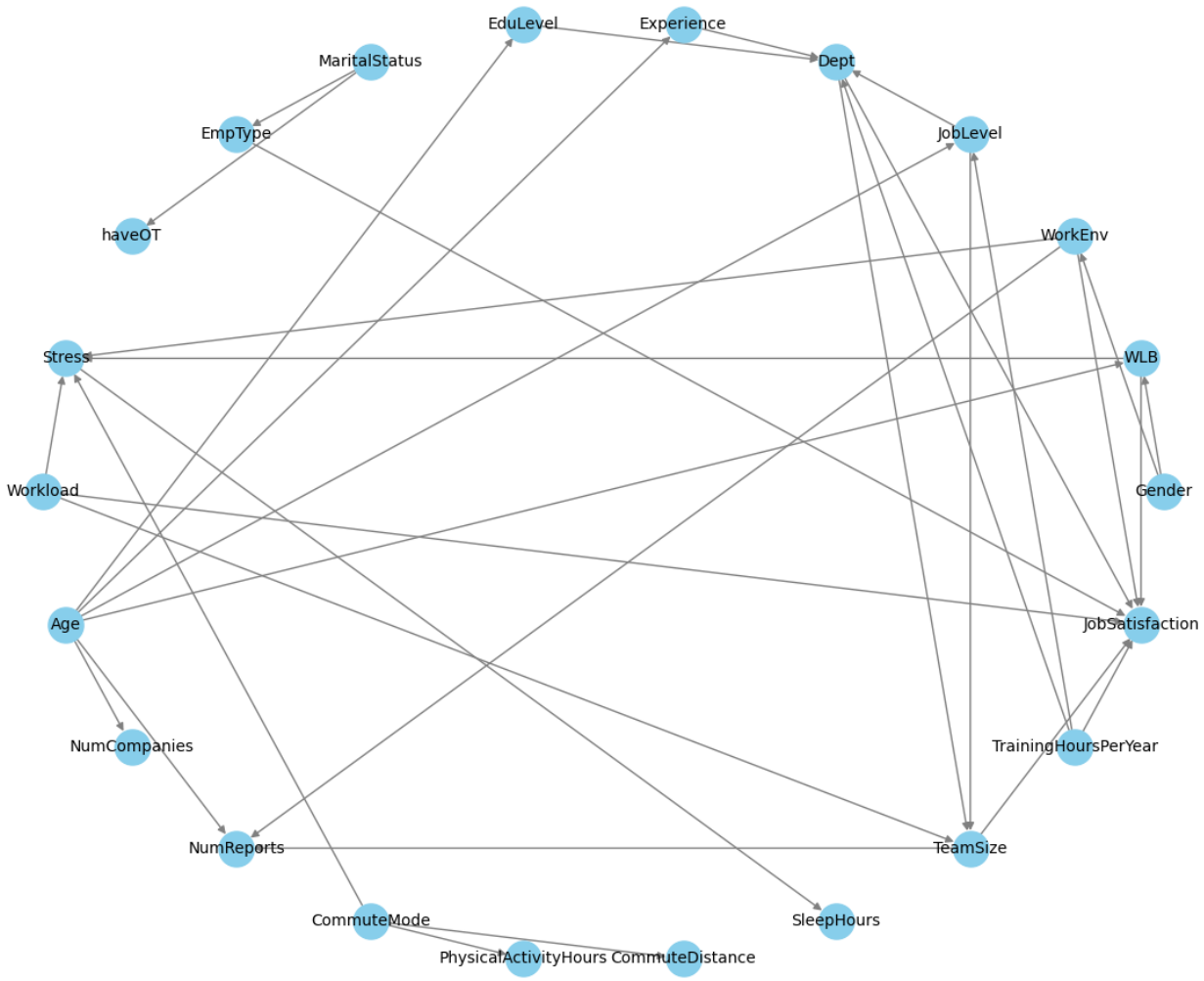
Ogni cluster viene identificato con un indice e con l'indice corrispondente nel dataset. Come possiamo vedere in entrambi i modelli la distribuzione dell'indagine sui dipendenti nei vari cluster non è perfettamente bilanciata, questo perché dipende da come è stato costruito ogni cluster.

Apprendimento Belief Network

Sommario

Una Belief Network è un modello grafico probabilistico. Considerata 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 utilizza 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:

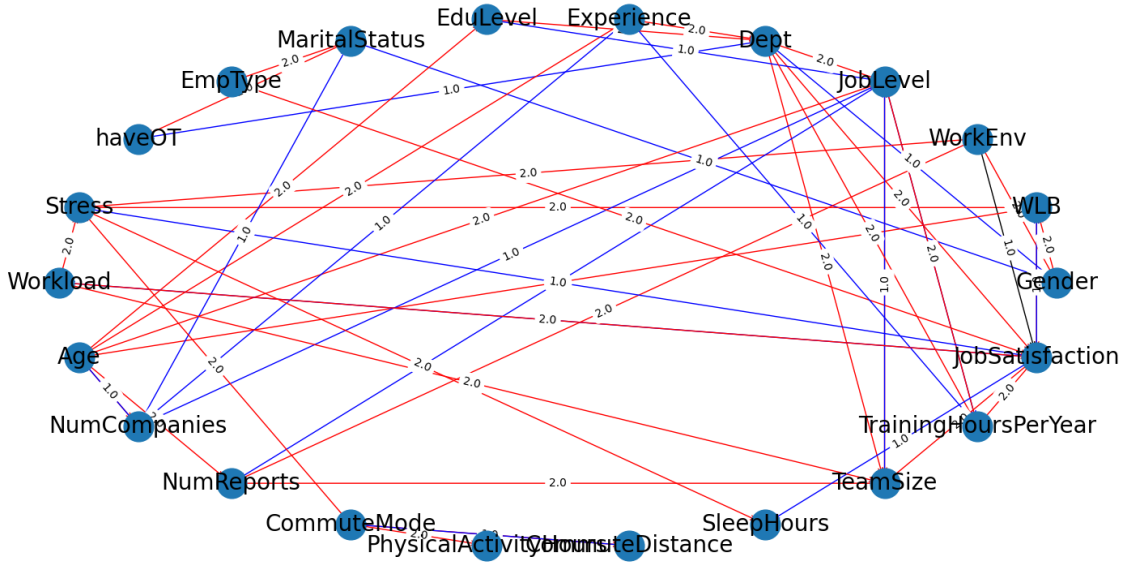
- **Variabili rilevanti**, ogni feature osservata è una variabile rilevante. Nel mio progetto, tutte le 23 feature sono state considerata variabili rilevanti;
- **Relazioni tra le variabili**: tra le variabili: la probabilità che assumono i valori di stress, yes e no, è influenzata direttamente da ('WLB', 'Stress'), ('WorkEnv', 'Stress'), ('Workload', 'Stress'), ('CommuteMode', 'Stress'). Di seguito il grafo completo delle relazioni, formalizzato in seguito all'osservazione dei dati:

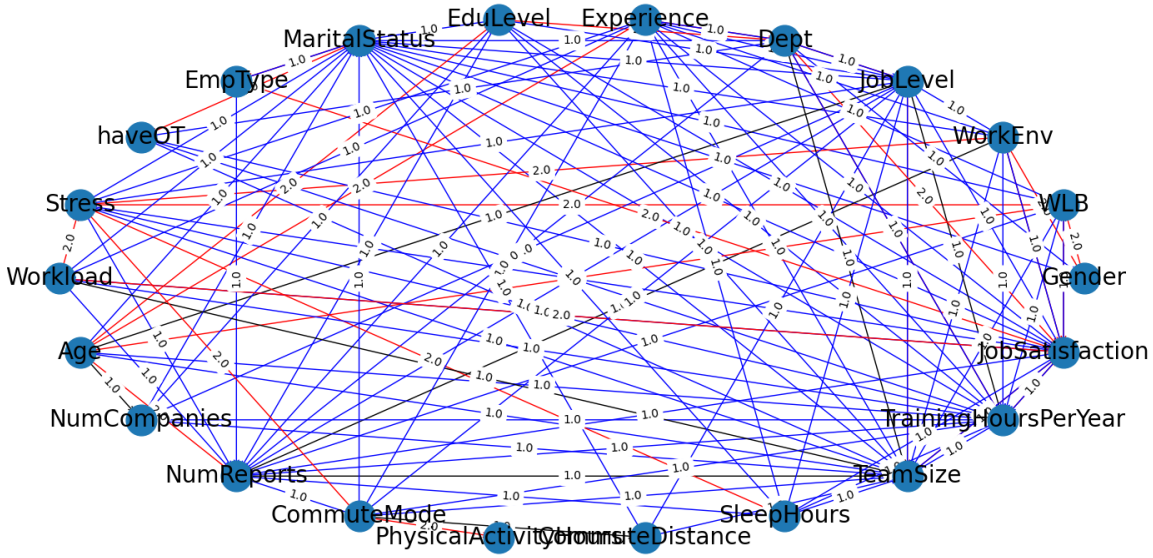


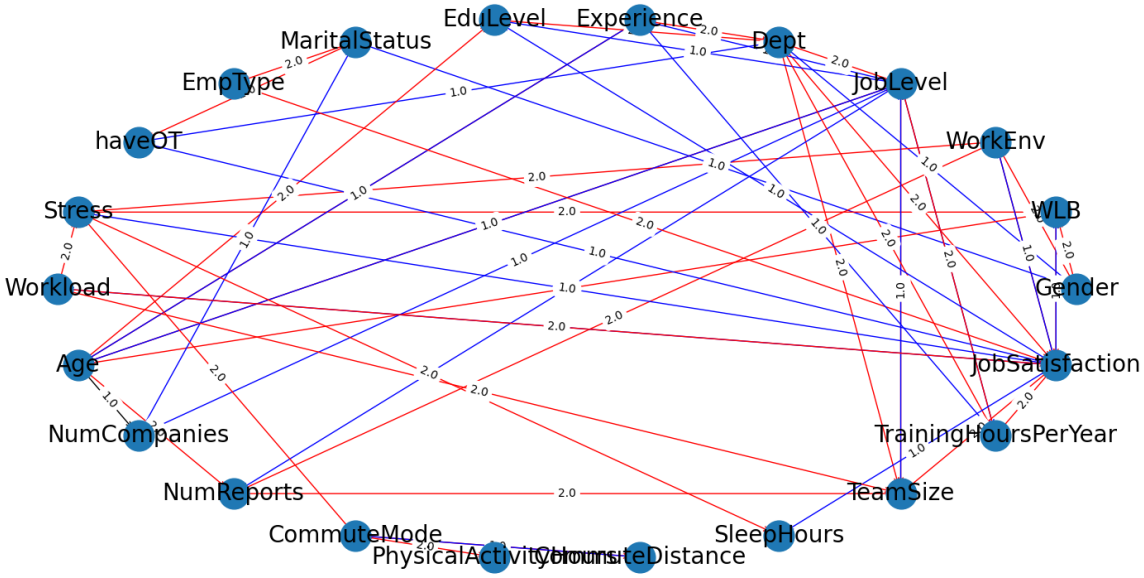
Ho deciso di fornire una struttura arbitraria alla rete bayesiana cercando di individuare le possibili dipendenze tra le feature sfruttando le possibili relazioni che possono influenzare una feature rispetto ad un'altra come per esempio lo stress può essere causato da un carico alto di lavoro.

La struttura illustrata 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*, articolata principalmente da: 1 l'algoritmo di ricerca, che attraversa lo spazio di analisi di tutti i possibili DAG per trovare la soluzione migliore (HillClimb search,Exhaustive search, etc.) 2 la funzione di score, che indica quanto la rete bayesiana è conforme ai dati (Bayesian Dirichlet scores: BDeu, K2, Bayesian Information Criterion:BIC, etc.).

Questi sono i risultati grafici:







Nel momento in cui c'è l'avvio al 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.

```
[Confronto della struttura con il modello BIC...]
Accuracy 0.8966942148760331
Precision 0.9291845493562232
Recall 0.9976958525345622
F-Score 0.9622222222222223
Error 0.10330578512396695
```

```
[Confronto della struttura con il modello K2...]
Accuracy 0.7727272727272727
Precision 0.9336734693877551
Recall 0.9786096256684492
F-Score 0.9556135770234987
Error 0.22727272727272727
```

```
[Confronto della struttura con il modello BDeu...]
Accuracy 0.8884297520661157
Precision 0.9285714285714286
Recall 0.9976744186046511
F-Score 0.9618834080717489
Error 0.1115702479338843
```

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 o dati 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 utilizzato:

```
[Prediction]
0: Gender
1: Age
2: MaritalStatus
3: JobLevel
4: Experience
5: Dept
6: EmpType
7: WLB
8: WorkEnv
9: PhysicalActivityHours
10: Workload
11: Stress
12: SleepHours
13: CommuteMode
14: CommuteDistance
15: NumCompanies
16: TeamSize
17: NumReports
18: EduLevel
19: haveOT
20: TrainingHoursPerYear
21: JobSatisfaction
Choose a variable to predict:11
```

2. L'utente sceglie quali sono le feature osservate e i rispettivi valori per esempio predizione sullo stress osservando la valutazione dell'ambiente di lavoro e sull'esperienza lavorativa:

```
0: Gender
1: Age
2: MaritalStatus
3: JobLevel
4: Experience
5: Dept
6: EmpType
7: WLB
8: WorkEnv
9: PhysicalActivityHours
10: Workload
11: Stress
12: SleepHours
13: CommuteMode
14: CommuteDistance
15: NumCompanies
16: TeamSize
17: NumReports
18: EduLevel
19: haveOT
20: TrainingHoursPerYear
21: JobSatisfaction
You already chose [11]
Choose an evidence:8
1: Very low
2: Low
3: Medium
4: High
5: Very high
Choose a work environment rating:3
```

```
Done choosing evidence(s)? y-nn
----------------------------------
0: Gender
1: Age
2: MaritalStatus
3: JobLevel
4: Experience
5: Dept
6: EmpType
7: WLB
8: WorkEnv
9: PhysicalActivityHours
10: Workload
11: Stress
12: SleepHours
13: CommuteMode
14: CommuteDistance
15: NumCompanies
16: TeamSize
17: NumReports
18: EduLevel
19: haveOT
20: TrainingHoursPerYear
21: JobSatisfaction
You already chose [11, 8]
Choose an evidence:4
1: [0, 7]
2: [8, 15]
3: [16, 23]
4: [24, 29]
Choose a work experience range:2
```

3. Il sistema restituisce la predizione

```
Done choosing evidence(s)? y-ny
Chosen evidence(s):
{'WorkEnv': 4, 'Experience': 3}
[{'Stress': 1, 'p': 0.5920087284734746}, {'Stress': 2, 'p': 0.1883172129396488}, {'Stress': 3, 'p': 0.1272536758186637}, {'Stress': 4, 'p': 0.06440510037298434}, {'Stress': 5,
 'p': 0.028015282396912856}]
```

Conclusioni

Nel presente lavoro sono state analizzate le principali tematiche e le valutazioni emerse, offrendo un quadro complessivo delle osservazioni e dei risultati raggiunti. Alcuni aspetti rilevanti, rimasti inesplorati a causa di limitazioni temporali, richiedono ulteriori approfondimenti, offrendo opportunità per sviluppi futuri ed estensioni in lavori successivi.

Esempi di argomenti da approfondire:

Approfondire l'argomento del ragionamento logico mediante l'aggiunta di clausole e la formulazione di query più dettagliate sulle relazioni tra dipendenti, salute e stress.

Combinando le previsioni di soddisfazione con altri fattori di stress e bilancio vita-lavoro, è possibile costruire un modello che predica il rischio di turnover.

Costruire un sistema che suggerisca interventi specifici per migliorare la soddisfazione lavorativa, come la promozione di attività fisica o programmi di benessere.

Attraverso l'apprendimento, simulare l'impatto di possibili cambiamenti nelle politiche aziendali (come l'aumento del numero di ore di formazione) che possano ridurre lo stress dei dipendenti.