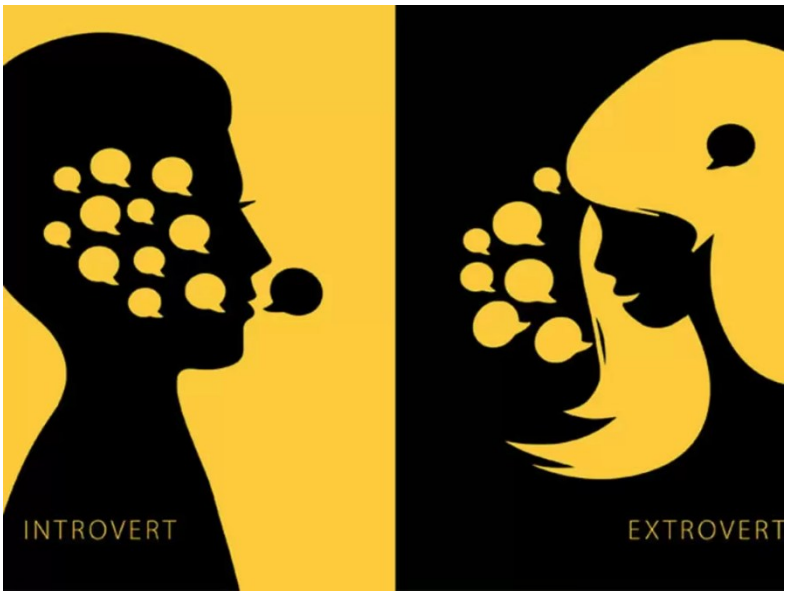
Studentessa: **Finiguerra Alessia**

Matricola: **735326**

Email: a.finiguerra1@studenti.uniba.it

URL GitHub: https://github.com/alefiniguerra01/ICON-Personality_Prediction.git

UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO



SISTEMA DI PREDIZIONE DELLA PERSONALITA': INTROVERSO VS ESTROVERSO

A.A. 2024/2025

Sommario

1. INTRODUZIONE

La personalità di una persona è l'insieme delle caratteristiche psicologiche che influenzano il modo in cui pensa, sente e si comporta nelle diverse situazioni. Essa ci rende unici e può comprendere tratti come l'emotività, la socievolezza, la stabilità e la creatività.

Tra le molte dimensioni della personalità, una classificazione degli individui può essere quella tra persone "*introverse*" e persone "*estroverse*": le persone introverse tendono a essere più riservate, riflessive e a preferire ambienti tranquilli, mentre quelle estroverse sono generalmente più socievoli, energiche e stimolate dal contatto con gli altri.

L'obiettivo di questo progetto è quello di costruire un modello in grado di prevedere accuratamente la personalità di un individuo, utilizzando le principali tecniche di apprendimento automatico.

2. ARGOMENTI DI INTERESSE

Gli argomenti affrontati in questo progetto sono:

- **Apprendimento supervisionato**: è una tecnica in cui il modello impara da un dataset fornito in input; attraverso l'applicazione di vari algoritmi, il modello viene poi addestrato per essere in grado di effettuare delle previsioni accurate;
    - **Apprendimento supervisionato con iperparametri**: attraverso la definizione di iperparametri si affinano le capacità del modello in modo da massimizzare l'accuratezza delle sue previsioni;
- **Sistema esperto**: viene creata una base di conoscenza e si applica un motore inferenziale che sarà in grado di ragionare logicamente e arrivare ad una conclusione, emulando le capacità decisionali di un esperto umano.

3. APPRENDIMENTO SUPERVISIONATO

3.1 STRUMENTI UTILIZZATI

Il progetto è stato realizzato nel linguaggio *Python* utilizzando l'IDE *Visual Studio Code* come ambiente di sviluppo. Tra le varie librerie necessarie abbiamo:

- **Pandas:** per la manipolazione e l'analisi dei dati;
- **Matplotlib e Seaborn:** per creare e gestire i grafici;
- **Scikit-learn:** per addestrare e valutare il modello;
- **Numpy:** per eseguire operazioni matematiche su specifici dati.

3.2 DATA EXPLORATION

La fase di Data Exploration è la fase iniziale di ogni progetto di Machine Learning. Il suo obiettivo principale è quello di esplorare il dataset per comprenderne le sue caratteristiche.

3.2.1 DATASET

Per valutare e fare previsioni, un modello di Machine Learning ha bisogno di un set di dati da cui "imparare", che farà quindi da base per costruire la sua conoscenza.

Per predire il tipo di personalità, il nostro modello ha utilizzato il dataset "personality_dataset.csv" (disponibile su Kaggle) contenente 2900 righe e 8 colonne.

Di seguito abbiamo la spiegazione delle varie features:

- **Time_spent_Alone:** numero di ore che l'individuo scorre da solo ogni giorno;
- **Stage_fear:** presenza della paura da palcoscenico;
- **Social_event_attendance:** frequenza di partecipazione ad eventi sociali;
- **Going_outside:** frequenza con cui l'individuo esce;
- **Drained_after_socializing:** presenza della sensazione di svuotamento dopo aver socializzato;
- **Friends_circle_size:** numero degli amici più intimi;
- **Post_frequency:** frequenza di pubblicazione sui social media;
- **Personality:** questa rappresenta la variabile **target** e indica appunto se l'individuo è "Introverso" o "Estroverso".

3.2.2 VALORI DUPLICATI

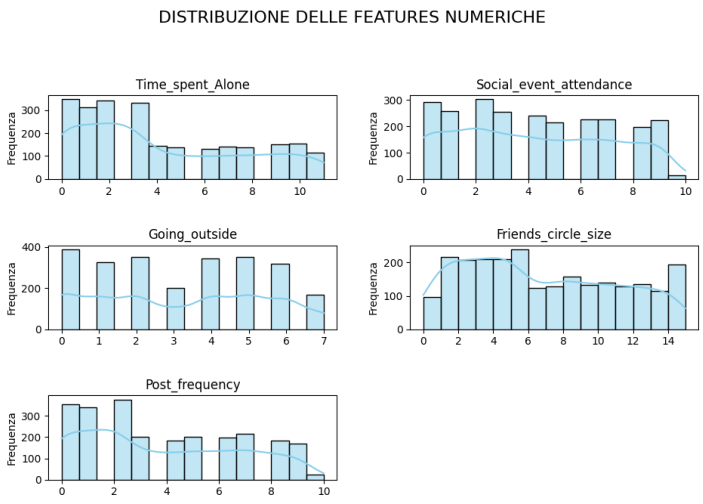
Per garantire l'integrità dei dati e prevenire un addestramento distorto del modello, è stato ripulito il dataset eliminando eventuali valori duplicati presenti al suo interno.

3.2.3 EDA: EXPLORATORY DATA ANALYSIS

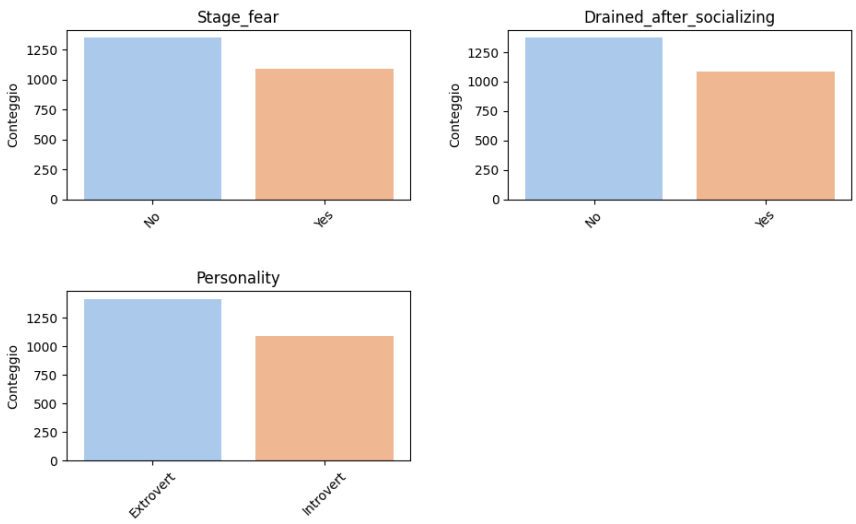
L'*EDA* (*Exploratory Data Analysis*) è la fase iniziale dell'analisi dei dati in cui si esplorano, visualizzano e comprendono le caratteristiche principali di un dataset.

L'obiettivo è identificare anomalie, distribuzioni e relazioni tra variabili, senza ancora applicare modelli predittivi.

Per visualizzare la frequenza con cui determinati valori compaiono nel dataset, sono stati creati dei grafici per confrontare visivamente come essi sono distribuiti rispetto alle relative caratteristiche.

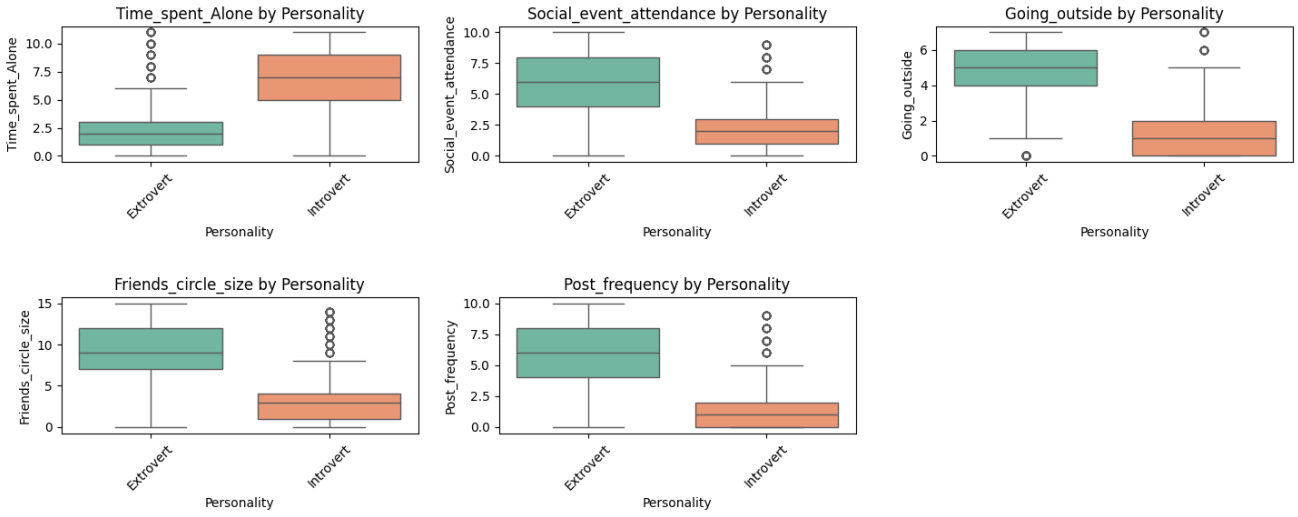


DISTRIBUZIONE DELLE FEATURES CATEGORICHE

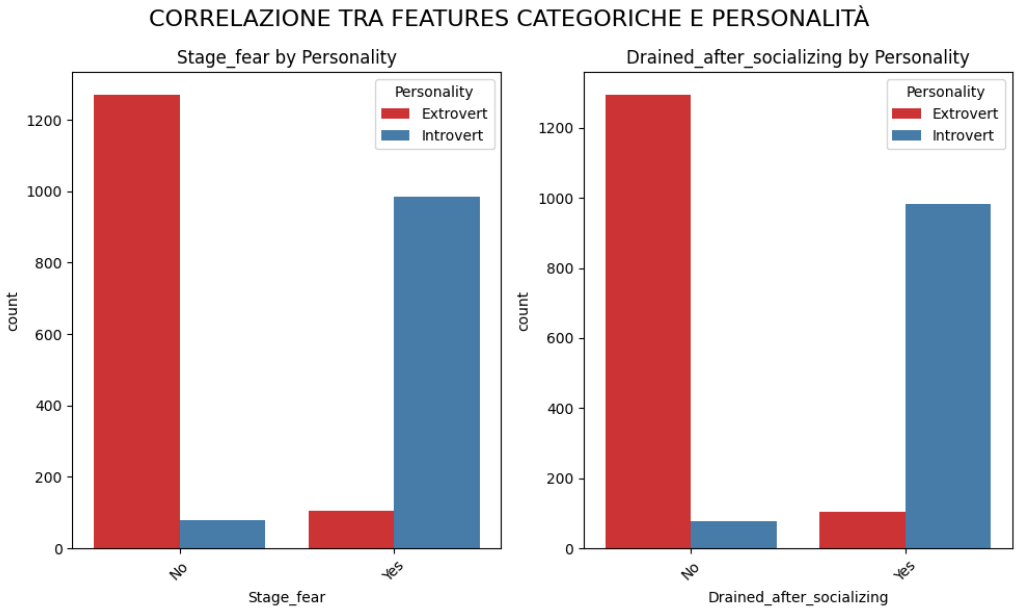


Successivamente, è stata effettuata l'analisi della correlazione tra le varie features con la feature target "Personality".

CORRELAZIONE TRA FEATURES NUMERICHE E PERSONALITÀ



Come si può notare dai grafici, l'analisi esplorativa conferma che l'insieme delle feature numeriche è fortemente correlato con la variabile target "Personality".
I grafici mostrano che per ogni metrica esaminata, come il tempo trascorso da soli o la partecipazione a eventi sociali, i comportamenti degli individui estroversi e introversi si raggruppano in modo nettamente distinto, confermando che ciascuna di queste feature contribuisce a definire il loro profilo di personalità.



Anche le features categoriche contribuiscono in modo determinante alla definizione della personalità di un individuo. Dai grafici, infatti, si può osservare una netta separazione nei comportamenti dei due gruppi.

3.2.4 OUTLIER

All'interno di un dataset possono essere presenti dei valori che a primo impatto possono sembrare anomali, i cosiddetti "outlier", ovvero dei valori che si discostano in modo significativo dal resto delle osservazioni.

Dopo un'attenta analisi del nostro dataset, si è deciso di non procedere con l'eliminazione degli outlier identificati, in quanto si è supposto che tali valori non rappresentino errori di misurazione o di inserimento, ma piuttosto variazioni autentiche e naturali del comportamento umano.

3.3 DATA PREPROCESSING

Una volta terminata la fase di Data Exploration si passa alla fase di Data Preprocessing.

In questa fase, i dati a nostra disposizione sono stati preparati per essere utilizzati successivamente dal modello durante l'addestramento.

L'obiettivo è quello di migliorare la qualità del dataset e di strutturarlo in modo adeguato.

3.3.1 VALORI NULLI

Nel dataset a disposizione erano presenti dei valori nulli sia per le features numeriche sia per quelle categoriche.

Per le features numeriche, si è deciso di sostituire i valori mancanti utilizzando un approccio basato sull'algoritmo KNN: per ogni valore mancante, sono stati trovati i tre vicini che avevano il valore più simile ad esso; l'algoritmo KNN ha calcolato poi la media dei tre valori e l'ha usata per sostituire il valore mancante.

Quest'approccio è stato valutato come il più performante perché tiene conto dei valori delle altre features e delle relazioni che intercorrono tra esse, portando a un dataset più realistico.

Per le features categoriche, invece, si è deciso di riempire i valori mancanti con la moda di ogni specifica feature.

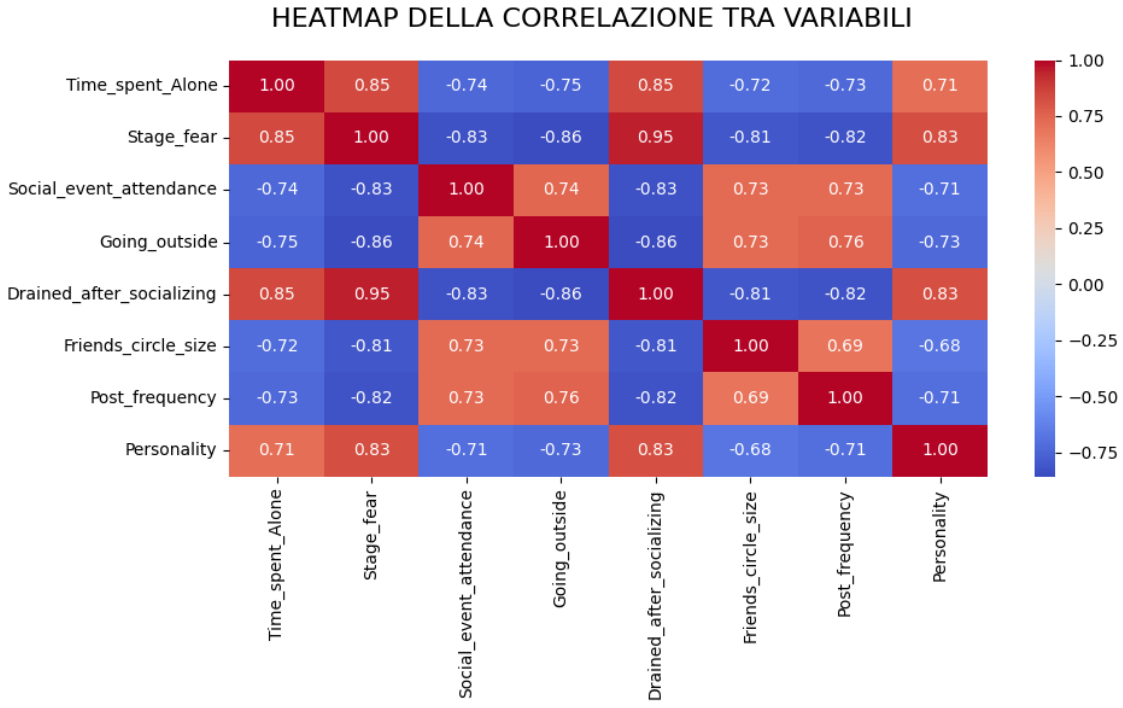
3.3.2 CONVERSIONE DELLE FEATURES CATEGORICHE

Gli algoritmi di apprendimento supervisionato richiedono che le features di input siano rappresentate mediante valori numerici.

Nel nostro dataset sono presenti due features categoriche (oltre alla variabile target). Per questo motivo, si è reso necessario convertire i loro valori in valori numerici, rappresentando il valore "Yes" come "1" e il valore "No" come "0".

Per la variabile target, invece, si è utilizzato un LabelEncoder che, dopo aver analizzato la feature "Personality", ha assegnato un numero per ogni valore trovato, convertendo alla fine "Extrovert" con "0" e "Introvert" con "1".

Dopo aver fatto ciò, è stata creata una heatmap che evidenzia la correlazione tra tutte le features.



E' possibile notare che le variabili "Stage_fear" (paura da palcoscenico) e "Drained_after_socializing" (senso di svuotamento dopo aver socializzato) sono fortemente correlate alla variabile target "Personality": alti valori delle due features prese in considerazione corrispondono ad un valore alto di "Personality", il che indica che si tratta di una persona introversa; al contrario, siamo di fronte ad una persona estroversa.

3.4 MODEL BUILDING

Terminata la fase di Preprocessing, si procede con l'addestramento del modello.

3.4.1 DATA SPLITTING

Per l'addestramento del modello si è deciso di considerare l'80% dei dati del dataset come training set e il restante 20% come test set.

3.4.2 STANDARDIZZAZIONE

Per mettere tutte le features su una scala comune in modo tale che l'algoritmo possa valutare l'importanza di ciascuna di esse basandosi solo sulla loro reale capacità predittiva, i valori di ogni feature sono stati standardizzati mediante la formula dello Z-score.

3.5 MODEL TRAINING AND EVALUATION

Per eseguire la predizione sulla personalità di un individuo, sono stati utilizzati diversi modelli di classificazione. Tenendo conto della quantità limitata dei dati presenti nel dataset, si è preferito scegliere modelli con complessità più semplice rispetto alla regressione lineare o alle reti neurali che richiedono una miglior precisione legata ad un dataset molto più fornito.

I modelli di classificazione utilizzati sono: Random Forest, Gradient Boosting, Decision Tree, SVM, e KNN.

All'interno del file "*train_eval.py*" sono stati addestrati e testati i diversi modelli senza l'utilizzo di particolari parametri, solo per prendere visione dei risultati iniziali.

Di seguito i risultati:

```
-----Decision Tree= Accuracy: 0.849-----

 Classification Report:
              precision    recall  f1-score   support

   Extrovert       0.86      0.87      0.87       282
   Introvert       0.83      0.82      0.83       221

    accuracy                           0.85       503
   macro avg       0.85      0.85      0.85       503
weighted avg       0.85      0.85      0.85       503

------------------
```

```
-----Gradient Boosting= Accuracy: 0.932-----

 Classification Report:
              precision    recall  f1-score   support

   Extrovert       0.94      0.94      0.94       282
   Introvert       0.92      0.92      0.92       221

    accuracy                           0.93       503
   macro avg       0.93      0.93      0.93       503
weighted avg       0.93      0.93      0.93       503

------------------
```

```
-----SVM= Accuracy: 0.932-----

 Classification Report:
              precision    recall  f1-score   support

   Extrovert       0.94      0.94      0.94       282
   Introvert       0.92      0.92      0.92       221

    accuracy                           0.93       503
   macro avg       0.93      0.93      0.93       503
weighted avg       0.93      0.93      0.93       503

------------------
```

```
-----Random Forest= Accuracy: 0.905-----

 Classification Report:
              precision    recall  f1-score   support

   Extrovert       0.91      0.92      0.92       282
   Introvert       0.89      0.89      0.89       221

    accuracy                           0.90       503
   macro avg       0.90      0.90      0.90       503
weighted avg       0.90      0.90      0.90       503

------------------
```

```
-----KNN= Accuracy: 0.920-----

 Classification Report:
              precision    recall  f1-score   support

   Extrovert       0.92      0.94      0.93       282
   Introvert       0.92      0.90      0.91       221

    accuracy                           0.92       503
   macro avg       0.92      0.92      0.92       503
weighted avg       0.92      0.92      0.92       503

------------------
```

Sono state condotte valutazioni preliminari delle prestazioni dei modelli, includendo metriche chiave come accuracy, precision e recall all'interno di un classification report.

L'accuracy indica la percentuale di previsioni totali che il modello ha indovinato correttamente.

La precision misura la percentuale di previsioni positive che erano effettivamente corrette. E' una metrica fondamentale per ridurre al minimo i falsi positivi, che si verificano nel momento in cui il modello erroneamente classifica un caso come positivo quando in realtà non lo è. Una precision elevata indica che i casi classificati come positivi sono molto probabilmente effettivamente positivi, riducendo così al minimo i falsi positivi.

La recall misura la percentuale di tutti i casi positivi reali che il modello è stato in grado di identificare. Essa, invece, è utile per ridurre al minimo i falsi negativi, che si verificano nel momento in cui il modello classifica erroneamente un caso come negativo quando in realtà è positivo.

La F1-score rappresenta la media armonica tra *Precision* e *Recall*. La sua importanza deriva dal fatto che un alto valore di questa metrica è indice di un alto valore sia di precision sia di recall.

Dopo una prima analisi è possibile notare che il Gradient Boosting e l'SVM sono risultati essere i modelli più performanti, con identici e alti livelli di precision, recall e F1-score, e con entrambi un'accuracy del 93.2%.

Seguono poi il KNN e il Random Forest con valori leggermente inferiori.

Infine, il Decision Tree si è rivelato essere il modello con le prestazioni più basse.

4. APPRENDIMENTO SUPERVISIONATO CON IPERPARAMETRI

Dopo l'addestramento e le valutazioni iniziali, si è deciso di migliorare le prestazioni di KNN, Random Forest e Decision Tree perché sono risultati essere gli algoritmi che hanno registrato i livelli più bassi accuracy tra tutti gli algoritmi testati.

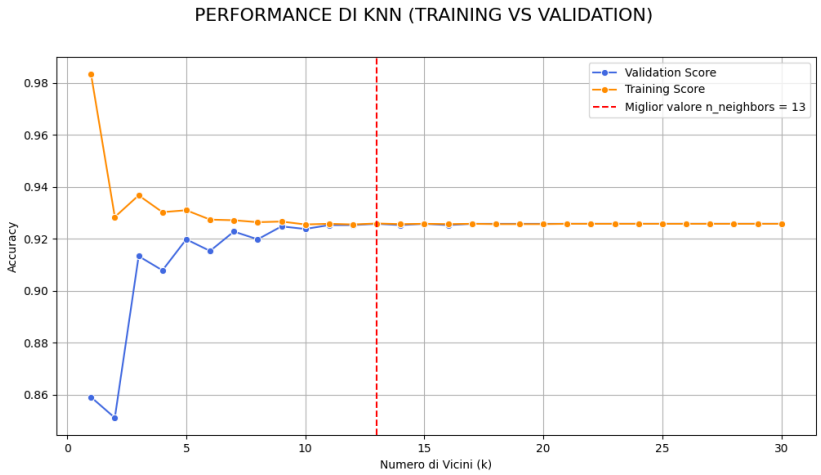
4.1 KNN

Il K-Nearest Neighbors (KNN) è un algoritmo il cui funzionamento si fonda sul principio di "somiglianza": per classificare un nuovo dato, l'algoritmo individua i "k" punti più vicini a esso nel set di addestramento e assegna la classe che risulta essere più comune tra questi vicini.

Per aumentare la performance del modello, si è preso in considerazione il suo iperparametro principale, n_neighbors, che definisce il numero di vicini da consultare.

Nel file "*optimized_KNN.py*", al fine di identificare il valore ottimale di n_neighbors, è stato eseguito un processo di tuning degli iperparametri utilizzando la tecnica Grid Search con Cross Validation a 5-fold. Durante questa fase, sono stati testati sistematicamente tutti i valori interi di n_neighbors da 1 a 30, con l'obiettivo di massimizzare la metrica di performance Accuracy.

4.1.1 VALUTAZIONI FINALI



Dal grafico soprastante è possibile notare una fase di overfitting del modello per valori di n_neighbors bassi (da 1 a 5); in questi casi, il modello è troppo "specifico" e impara a memoria i dati di addestramento, diventa cioè perfetto nel prevedere i dati che già conosce (da qui l'accuracy di training altissima), ma fallisce nel prevedere i nuovi dati.

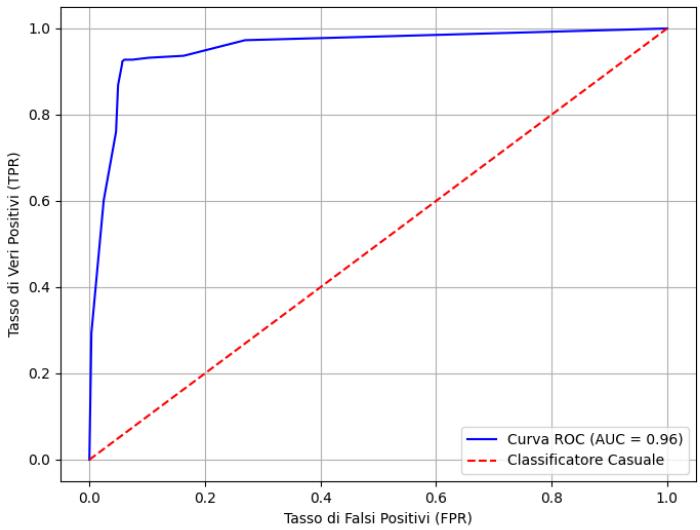
Aumentando il valore di n_neighbors, il modello diventa meno specifico e inizia a imparare i pattern generali dei dati invece di memorizzarli, migliorando la sua capacità di generalizzare.

La performance del modello raggiunge infine una fase ottimale e stabile per valori di n_neighbors maggiori di 11, identificando il suo valore migliore in 13.

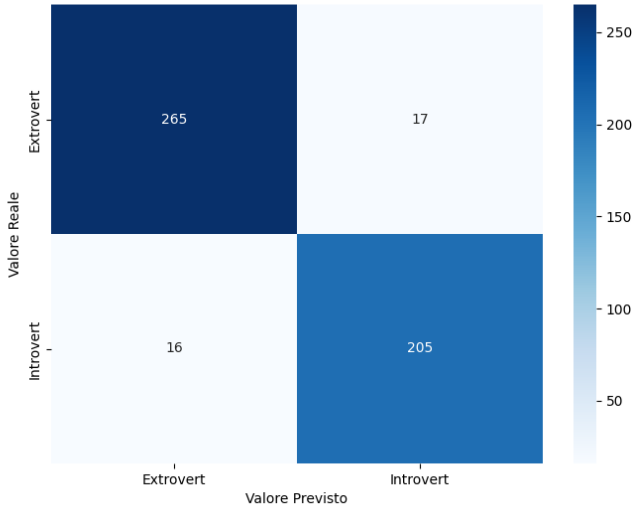
```
-----Valutazione finale del modello KNN ottimizzato-----

 Classification Report:
Accuracy: 0.934
              precision    recall  f1-score   support

   Extrovert       0.94      0.94      0.94       282
   Introvert       0.92      0.93      0.93       221

    accuracy                           0.93       503
   macro avg       0.93      0.93      0.93       503
weighted avg       0.93      0.93      0.93       503
```

Dalle valutazioni finali si può concludere che impostando l'iperparametro n_neighbors = 13 si sono migliorate le performance generali del modello, raggiungendo un accuracy del 93.4% (a fronte di un accuracy iniziale del 92%).

CURVA ROC PER KNN OTTIMIZZATO



MATRICE DI CONFUSIONE PER KNN OTTIMIZZATO



Anche la *matrice di confusione* e la *curva ROC* ci confermano che il processo di tuning dell'iperparametro n_neighbors ha avuto successo, generando un modello accurato e ben bilanciato nel gestire le due diverse classi di personalità.

La *matrice di confusione* confronta le etichette reali con le previsioni fatte dal modello; in questo modo è possibile visionare quante previsioni sono corrette o sbagliate e che tipo di errori vengono commessi.

La *curva ROC*, invece, è un grafico che visualizza la capacità di un modello di distinguere tra le classi, mostra cioè come la performance del classificatore cambia al variare della sua soglia di decisione. Con essa, viene calcolata anche *l'Area Sotto la Curva*, un valore compreso tra 0 e 1 che riassume l'intera performance; per valori maggiori di 0.8, generalmente, il modello viene considerato buono.

4.2 RANDOM FOREST

Il Random Forest è un algoritmo di apprendimento supervisionato che costruisce una moltitudine di alberi decisionali durante la fase di training e aggrega i loro risultati per produrre una previsione finale. Questo approccio è noto per la sua robustezza e la sua capacità di ridurre l'overfitting rispetto a un singolo albero decisionale.

Al fine di identificare la configurazione ottimale del modello, è stata adottata una strategia di tuning degli iperparametri in due fasi:

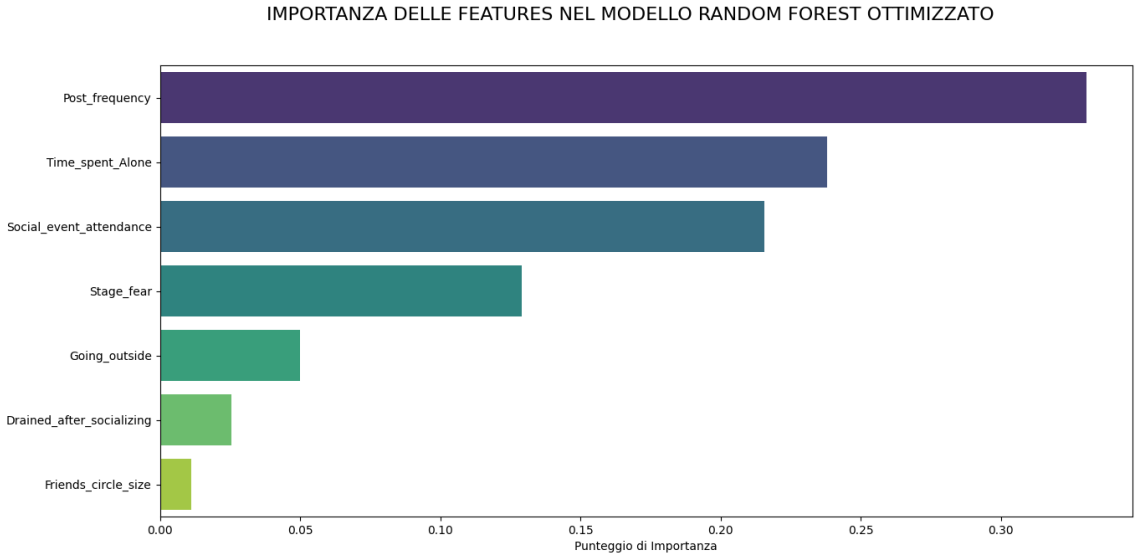
- **FASE 1**: per la prima fase si è deciso di implementare una Random Search con Cross Validation, eseguendo 25 iterazioni casuali sugli iperparametri:
  - **n_estimators**: numero di alberi decisionali che vengono costruiti;
  - **max_depth**: profondità massima che ogni singolo albero della foresta può aggiungere;
  - **min_samples_split**: numero minimo di campioni che un nodo interno deve contenere per poter essere ulteriormente suddiviso in altri nodi;
  - **min_samples_leaf**: numero minimo di campioni che devono trovarsi in un nodo terminale

  In questo modo, è stata trovata la "regione" più promettente dello spazio degli iperparametri, utilizzata nella fase successiva;
- **FASE 2**: nella seconda fase, i migliori valori per gli iperparametri emersi dalla fase precedente sono stati usati per eseguire una Grid Search con Cross Validation, che ha testato tutte le combinazioni possibili per determinare con precisione la configurazione ottimale, ottimizzando per la metrica accuracy.

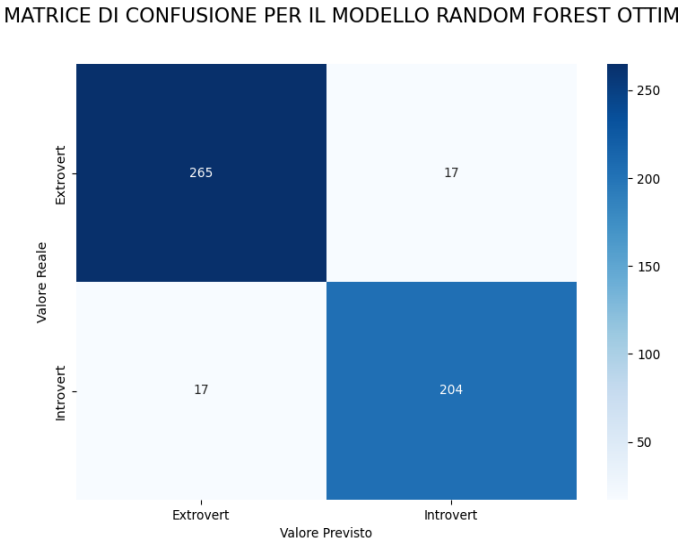
4.2.1 VALUTAZIONI FINALI

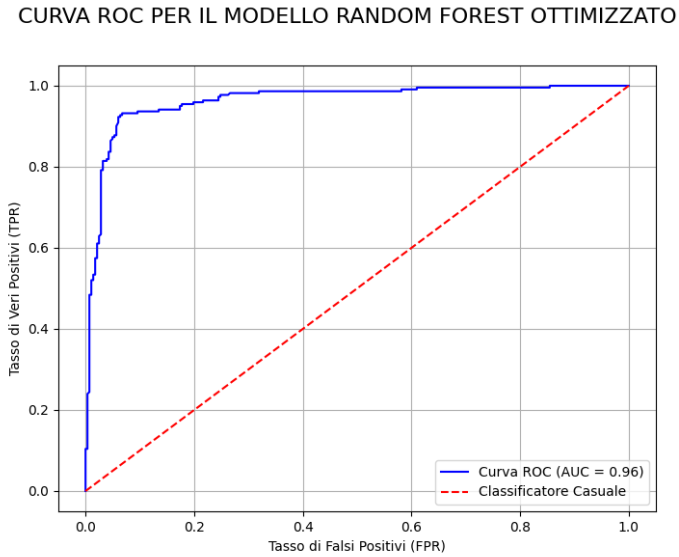
```
-----Valutazione finale del modello Random Forest ottimizzato-----

 Classification Report:
Accuracy: 0.932
              precision    recall  f1-score   support

   Extrovert       0.94      0.94      0.94       282
   Introvert       0.92      0.92      0.92       221

    accuracy                           0.93       503
   macro avg       0.93      0.93      0.93       503
weighted avg       0.93      0.93      0.93       503
```

Come si evince dal *classification report*, la fase di tuning del modello Random Forest ha portato ad un miglioramento generale e significativo del modello di classificazione, raggiungendo un'accuracy finale del 93.2% (a fronte di un'accuracy iniziale del 90.5%).



Con il grafico soprastante è possibile notare che soprattutto le features "Post_frequency", "Time_spent_Alone" e "Social_event_attendance" hanno contribuito maggiormente alle decisioni prese nella fase di tuning.





Anche la *curva ROC* e la *matrice di confusione* confermano che il processo di tuning ha avuto successo, portando ad un miglioramento significato della performance del modello Random Forest.

4.3 DECISION TREE

L'ultimo modello di cui si sono volute migliorare le performance è il Decision Tree.
Il Decision Tree è un algoritmo di apprendimento supervisionato che opera creando una struttura a flusso, simile ad un albero, dove ogni nodo interno decisionale rappresenta una

"domanda" su una specifica feature dei dati e ogni ramo una risposta. Seguendo il percorso dall'alto verso il basso, si arriva a una "foglia" finale che contiene la previsione della classe.

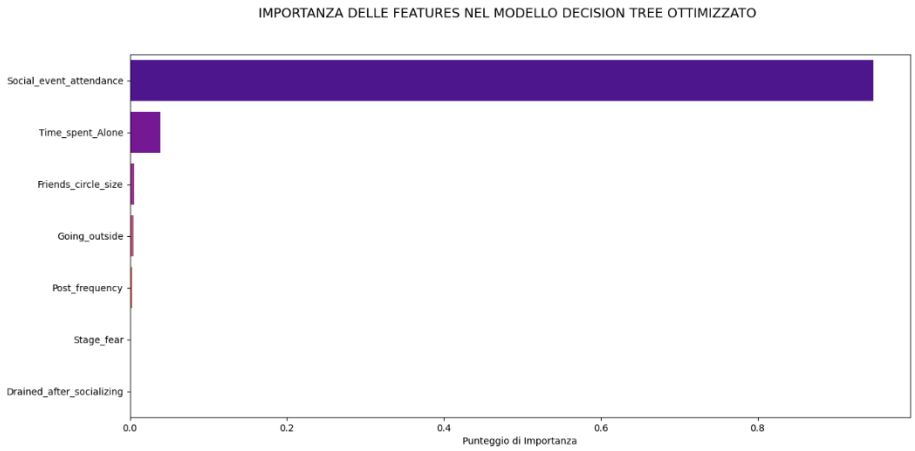
Se non ottimizzato, questo modello tende a soffrire di overfitting, imparando a memoria i dati di training. Per contrastare questo fenomeno, è stata eseguita una fase di tuning degli iperparametri tramite una Grid Search con Cross Validation, focalizzata sulla ricerca della configurazione ottimale per i seguenti iperparametri:

- **max_depth**: profondità massima dell'albero;
- **min_samples_split**: numero minimo di campioni per dividere un nodo;
- **min_samples_leaf**: numero minimo di campioni in un nodo foglia

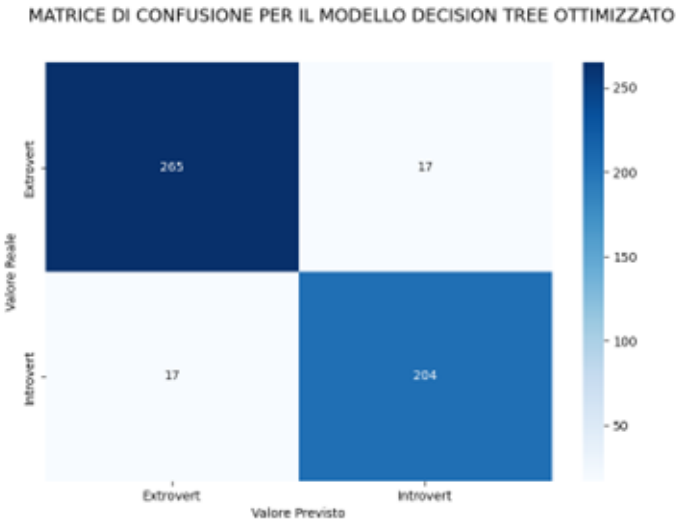
4.3.1 RISULTATI FINALI

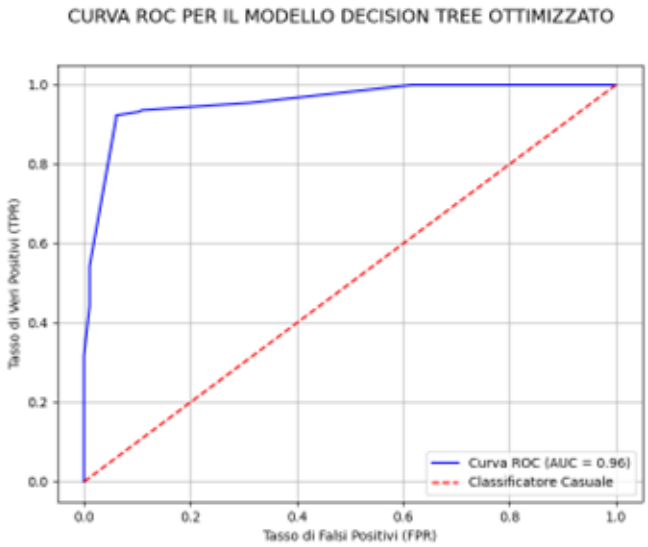
```
-----Valutazione finale del modello Decision Tree ottimizzato-----

 Classification Report:
Accuracy: 0.932
              precision    recall  f1-score   support

   Extrovert       0.94      0.94      0.94       282
   Introvert       0.92      0.92      0.92       221

    accuracy                           0.93       503
   macro avg       0.93      0.93      0.93       503
weighted avg       0.93      0.93      0.93       503
```

Il modello finale risultato di questo processo ha raggiunto una performance notevolmente migliorata sul test set, con un accuracy del 93.2% a fronte di un accuracy iniziale dell'84.9%. Un miglioramento così significativo è indice, molto probabilmente, di un Decision Tree che con i suoi iperparametri di default cresceva senza limiti andando incontro al fenomeno dell'overfitting.



Attraverso il grafico soprastante è possibile notare che il modello Decision Tree prevede il tipo di personalità di un individuo basando la sua classificazione principalmente sulla feature Social_event_attendance perché la ritiene in grado di suddividere molto bene gli estroversi dagli introversi.





Anche la *curva ROC* e la *matrice di confusione* confermano il netto miglioramento del modello Decision Tree dopo essere stato sottoposto alla fase di tuning.

4.4 CONCLUSIONI

La fase di tuning dei modelli ha dimostrato un notevole miglioramento delle prestazioni complessive rispetto ai modelli non ottimizzati.

Inizialmente, i modelli più efficienti erano il Gradient Boosting e l'SVM con un accuracy del 93.2%, seguiti poi da KNN, Random Forest e Decision Tree con un accuracy rispettivamente del 92%, 90.5% e 84.9%.

Dopo l'ottimizzazione dei parametri, tutti i modelli testati (Random Forest, KNN e Decision Tree) hanno mostrato un incremento significativo dell'accuratezza e delle altre metriche di valutazione. In particolare, il KNN ottimizzato è risultato essere il modello che ha raggiunto l'accuratezza più alta con un valore del 93.4%, superando leggermente il Random Forest e il Decision Tree che si sono stabilizzati ad un valore di accuracy pari a 93.2%.

5. SISTEMA ESPERTO

Un sistema esperto è un programma di intelligenza artificiale progettato per replicare le capacità di ragionamento di un esperto umano in un dominio specifico e ristretto.

È composto principalmente da tre parti:

1. la Base di Conoscenza (Knowledge Base), che contiene i dati e le regole del dominio;
2. il Motore Inferenziale (Inference Engine), ovvero il software che applica le regole ai dati per trarre conclusioni logiche;
3. l'Interfaccia Utente (User Interface), che permette a una persona di dialogare con il sistema, inserendo i fatti del problema e ricevendo le risposte.

5.1 STRUMENTI UTILIZZATI

Per implementare una base di conoscenza in logica è stato utilizzato il linguaggio Prolog con l'ausilio della libreria pyswip.

La componente di ragionamento del sistema esperto è affidata al motore inferenziale SWI-Prolog, un ambiente di programmazione che interpreta le regole definite nella Knowledge Base e le applica ai fatti specifici del caso per giungere alla determinazione della personalità.

5.2 KNOWLEDGE BASE

Per il progetto preso in considerazione è stata creata una Knowledge Base nel file "KB_personality.pl" sulla base del dominio che si desidera rappresentare, ovvero la determinazione dei tratti della personalità umana basata su un modello di comportamento.

Inizialmente sono state definite le due possibili categorie di output che il sistema è in grado di generare: tipo_personalita(introverso). e tipo_personalita(estroverso).
Successivamente, sono state definite le regole, cioè le proposizioni che saranno considerate vere nell'interpretazione del dominio. Esse rappresentano le verità fondamentali del dominio e costituiscono la base su cui è fondato il ragionamento logico all'interno del sistema.

Per rendere il sistema più accurato, è stato implementato un meccanismo di valutazione a punteggio: invece di avere regole che portano a una conclusione diretta, sono state definite delle regole tramite il predicato regola_punteggio, ognuna delle quali assegna un peso numerico (cioè un punteggio) a un determinato tratto della personalità (introverso oppure estroverso) quando le condizioni sono soddisfatte.

```
regola_punteggio(Persona, introverso, 3) :-
    ha_tratto(Persona, frequenza_uscite(bassa)),
    ha_tratto(Persona, svuotato_dopo_socializzazione(si)).
```

Successivamente, è stato definito il predicato punteggio_totale che si occupa di determinare il punteggio complessivo di ogni personalità sommando i punteggi parziali ottenuti dalle singole regole attivate.

```
% regola che calcola il punteggio totale di ogni personalità
punteggio_totale(Persona, Tipo, PunteggioTotale) :-
    findall(Punteggio, regola_punteggio(Persona, Tipo, Punteggio), ListaPunteggi),
    sum_list(ListaPunteggi, PunteggioTotale).
```

Infine, con la regola personalita_con_punteggio viene generata una lista di punteggi in cui ad ogni tipo di personalità viene assegnato il determinato punteggio.

```
% regola che determina la personalità
personalita_con_punteggio(Persona, Tipo, Punteggio) :-
    tipo_personalita(Tipo),
    punteggio_totale(Persona, Tipo, Punteggio).
```

5.3 INTERFACCIA UTENTE

L'interfaccia utente del sistema è stata implementata nel file "expert_system_personality.py". Il suo scopo principale è fare da mediatore tra l'utente finale e il motore inferenziale di Prolog, permettendo un'interazione semplice senza che l'utente debba conoscere la sintassi o la logica della Knowledge Base definita.

Inizialmente viene avviato un dialogo con l'utente in cui il sistema pone delle domande e, attraverso delle funzioni specifiche, raccoglie e valida le risposte fornite dall'utente, continuando a richiederle finchè non vengono inserite nel formato corretto.

```
Provi paura da palcoscenico? (si/no): no
```

```
Ti senti svuotato dopo aver socializzato? (si/no): 6
ERRORE: Inserimento non valido, per favore inserisci 'si' o 'no'.
```

Tutte le risposte fornite dall'utente vengono tradotte e asserite dinamicamente nella Knowledge Base. Infine, il sistema interroga il motore inferenziale per ottenere la previsione finale.

```
--- Risultato finale: ---

La personalità dell'individuo è: ESTROVERSO
```