16 PERSONALITÀ

TEST DI MYERS BRIGGS

Classificazione

INDICE

1. Introduzione

Il test delle 16 personalità, noto come Myers-Briggs Type Indicator (MBTI), è un test psicologico volto a categorizzare le persone in base a specifici tratti di personalità. La sua origine affonda le radici negli studi di Carl Gustav Jung, psicologo svizzero, che nel 1921 pubblicò "Tipi Psicologici", un'opera fondamentale che introduce l'idea di diverse modalità di percezione e giudizio. Jung identificò tre dicotomie principali: Estroversione/Introversione, che descrive la direzione del flusso energetico; Sensazione/Intuizione, che riguarda il modo in cui le persone percepiscono il mondo; e Pensiero/Sentimento, che concerne il processo decisionale. Tuttavia, la teoria di Jung rimase a lungo un costrutto teorico senza una concreta applicazione pratica. Fu Katharine Cook Briggs, appassionata di psicologia, a raccogliere l'eredità di Jung negli anni '20. Osservando i comportamenti umani, Briggs intuì la possibilità di strutturare un sistema per comprendere meglio le diverse personalità. Negli anni '40, sua figlia Isabel Briggs Myers si unì a lei, e insieme diedero vita al Myers-Briggs Type Indicator (MBTI), un questionario standardizzato che traduceva la teoria junghiana in uno strumento pratico. La loro innovazione più significativa fu l'introduzione della quarta dicotomia, Giudizio/Percezione, che permetteva di analizzare il modo in cui le persone interagiscono con il mondo esterno. Grazie a questo contributo, il MBTI divenne un metodo per identificare e comprendere i diversi tipi di personalità.

1.1 Come funziona il test MBTI

Il test si basa su quattro coppie di preferenze dicotomiche, che combinandosi tra loro danno origine a 16 possibili tipi di personalità, descrivendo come una persona reagisce in determinati contesti sociali e nell'affrontare i problemi della vita di tutti i giorni.

1.2 Introversione (I) vs Estroversione (E):

Questa dicotomia descrive il modo in cui le persone si pongono di fronte alla questione delle interazioni sociali. L'Introversione caratterizza un individuo che si sente più a proprio agio in ambienti tranquilli e solitari, traendo energia attraverso la riflessione e il tempo da soli. L'Estroversione, al contrario, è tipica di un individuo che trae serenità dalle interazioni sociali, preferendo ambienti più dinamici che favoriscono le interazioni con altri individui.

1.3 Sensazione (S) vs Intuizione (N):

Questa dicotomia riguarda il modo in cui le persone raccolgono ed interpretano le informazioni. chi predilige la Sensazione, si basa sulla raccolta di fatti reali e tangibili, concentrandosi molto sui dettagli. Chi invece predilige l'Intuizione, lascia spazio alla

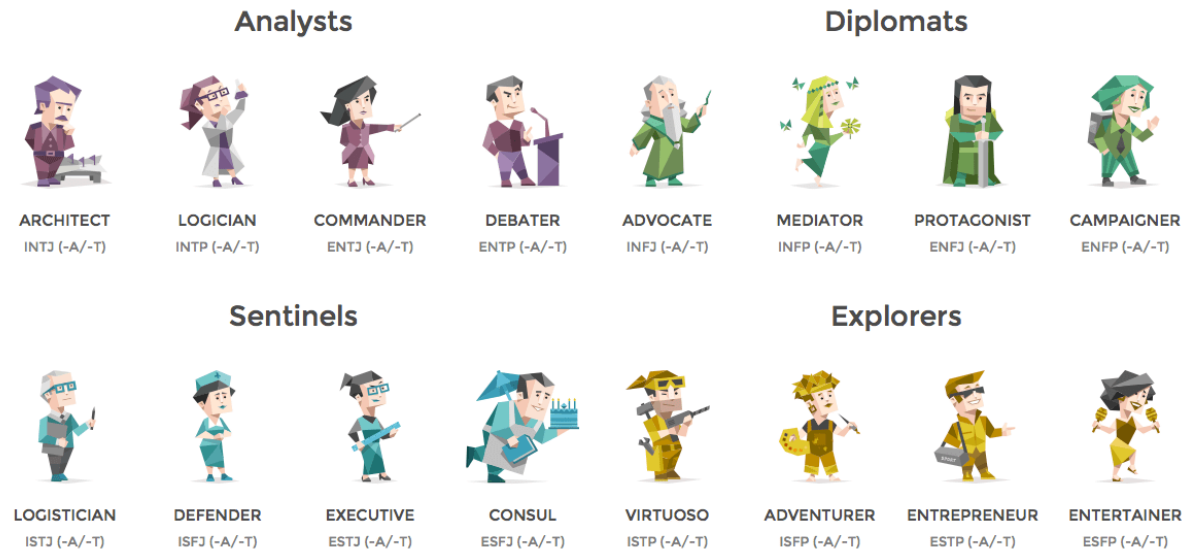
creatività, favorendo l' interpretazione personale nel captare le informazioni raccolte.

1.4 Pensiero (T) vs Sentimento (F):

Descrive come le persone affrontano una presa di decisione. Coloro che prediligono il Pensiero, prendono le decisioni nel modo più razionale possibile, analizzando il problema in modo analitico ed oggettivo. Invece coloro che prediligono il Sentimento, prendono le decisioni lasciandosi influenzare dall'emotività, considerando particolarmente i valori personali che li contraddistinguono.

1.5 Giudizio (J) vs Percezione (P):

Questa dicotomia distingue come le persone si approcciano alla vita ed agli impegni che riempiono le loro giornate. Il Giudizio contraddistingue coloro che amano la pianificazione, per avere sempre la situazione sotto controllo. Chi invece predilige la Percezione, cerca di godersi al pieno ogni giornata, senza lasciarsi influenzare da ciò che gli riserverà il futuro, senza troppe pianificazioni, lasciando sempre aperta ogni opzione.



2. Dataset

Il dataset "16P" si basa sul noto test delle 16 personalità di Myers-Briggs, descritto nell'Introduzione. Sebbene sia un dataset sintetico, l'autore ha dichiarato di averlo sviluppato con cura, agendo su ogni singola feature. In particolare, ha migliorato i dati dopo averli testati su modelli di machine learning, analizzando le prestazioni al variare degli iperparametri degli algoritmi utilizzati. Di conseguenza, il dataset risulta adeguato per l'applicazione prevista in questo progetto.
È composto da un totale di 62 colonne: due di queste contengono l'identificativo dell'intervistato/a e la label di classe, mentre le restanti 60 rappresentano gli attributi. Questi ultimi corrispondono a 60 domande, con risposte espresse tramite valori interi ordinali su una scala Likert. L'autore del dataset ha opportunamente scalato i valori di questa scala, che in questo caso è a 7 punti.
La scala Likert è uno strumento di misurazione numerica ampiamente utilizzato per permettere alle persone di esprimere la propria opinione su determinati argomenti. Solitamente, è composta da 5 o 7 livelli, con numeri dispari per includere un'opzione neutra, spesso rappresentata dallo zero.

I valori sono ordinati come segue:

3 : Molto d'accordo
2: D'accordo
1: Abbastanza d'accordo
0: Né d'accordo né in disaccordo
-1: Abbastanza in disaccordo
-2: Disaccordo
-3: Molto in disaccordo

L'utilizzo della scala Likert è tra i più diffusi nelle analisi statistiche basate sull'interpretazione di questionari, progettati per raccogliere e descrivere atteggiamenti e opinioni degli individui in diversi contesti. Questa scala consente di sintetizzare le risposte, trasformandole in valori numerici che rappresentano il grado di accordo o disaccordo rispetto alle affermazioni proposte. La conversione da dati qualitativi a dati numerici facilita l'analisi statistica e rende la scala Likert uno strumento fondamentale non solo negli studi psicometrici, ma anche in molte altre discipline. Nel contesto del Machine Learning, questa trasformazione permette di analizzare opinioni soggettive che altrimenti sarebbero difficili da interpretare. L'assegnazione di punteggi numerici consente il confronto tra individui e aiuta i modelli di apprendimento automatico a individuare schemi e correlazioni nei dati raccolti

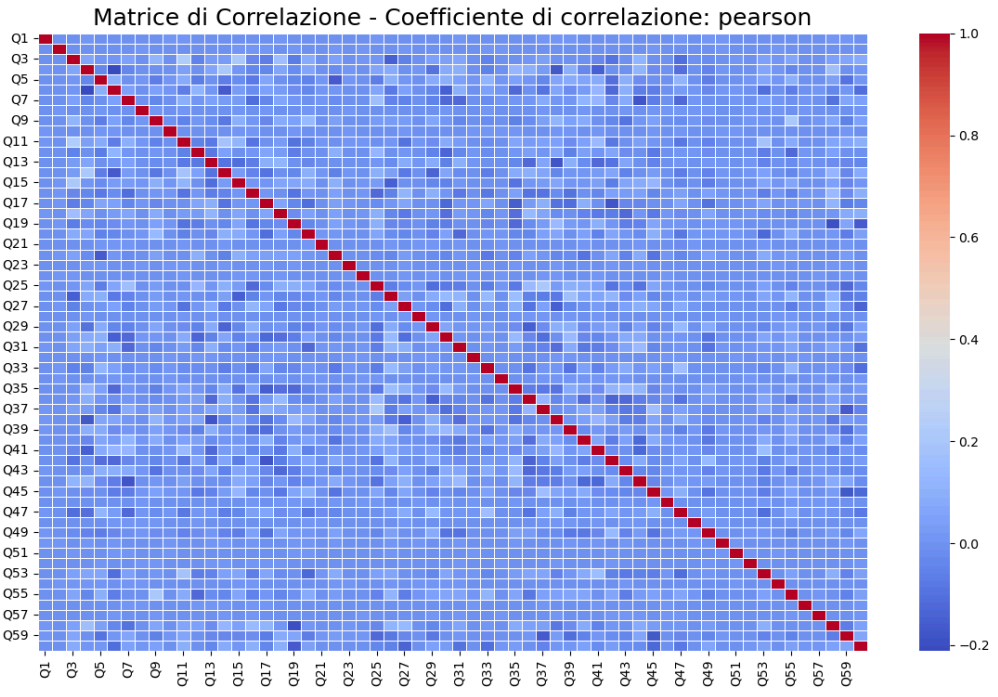
3. Data Analysis

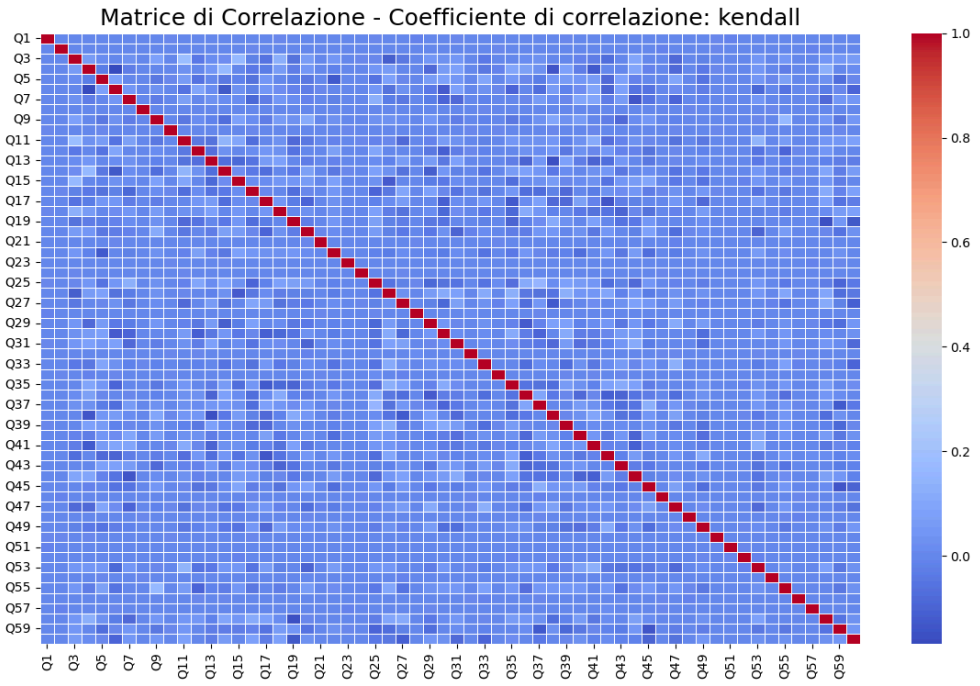
3.1 Qualità dei dati

La qualità dei dati raccolti attraverso la scala Likert può essere influenzata da diversi fattori, tra cui la soggettività degli intervistati, il loro stato emotivo al momento della compilazione e altri eventi esterni. Le risposte riflettono le loro percezioni personali, che possono variare significativamente da individuo a individuo. L'umore dell'intervistato al momento della compilazione del questionario può influenzare le risposte; ad esempio, una persona che sta attraversando una giornata negativa potrebbe valutare più severamente determinati aspetti rispetto a quando è di buon umore. Sebbene il dataset sia sintetico è buona norma considerare gli aspetti appena elencati e utilizzare delle tecniche in grado di "sopportare i bias" oppure selezionare solo gli attributi significativi.

3.2 Correlazione delle features

Prima di applicare tecniche di machine learning, è fondamentale comprendere il contesto e le relazioni tra i dati disponibili. Un'analisi preliminare delle correlazioni tra gli attributi (le domande) potrebbe fornire intuizioni preziose sulla struttura del dataset. Calcolare la matrice di correlazione tra le features del dataset consente di individuare relazioni lineari tra le variabili: il coefficiente di correlazione di Pearson è comunemente utilizzato a questo scopo, poiché misura la forza e la direzione della relazione lineare tra due variabili. Tuttavia, è importante notare che le scale Likert producono dati ordinali, il che potrebbe rendere il coefficiente di Spearman più appropriato per valutare le correlazioni, in quanto misura la relazione monotona tra le variabili senza assumere linearità.
Sono state eseguite entrambe le analisi, utilizzando sia il coefficiente di Pearson, sia quello di Spearman grazie alle funzioni apposite fornite di scikit-learn.





Dai plot sopra riportati, si può osservare che il coefficiente di correlazione di Pearson, progettato per evidenziare relazioni lineari tra le features, restituisce valori molto bassi. Allo stesso modo, la seconda matrice di correlazione calcolata utilizzando il coefficiente di Kendall mostra risultati pressoché identici.

Questi dati suggeriscono che non esiste una correlazione lineare (ovvero, di tipo matematico) significativa tra le features analizzate. Tuttavia, è possibile avanzare una considerazione più approfondita sulla natura delle correlazioni all'interno del dataset.

Analizzando il funzionamento del test e basandosi su una conoscenza a priori della sua struttura, si può ipotizzare l'esistenza di una correlazione di tipo "contestuale" tra le diverse domande. Infatti, ogni domanda è progettata per identificare una specifica dicotomia della personalità e distinguere tra due tratti opposti (termini dicotomici).

Ad esempio, la domanda "Stringi regolarmente nuove amicizie?" potrebbe essere associata alla dicotomia Introversione (I) - Estroversione (E). Di conseguenza, è plausibile che essa presenti una correlazione con altre domande che mirano a distinguere la stessa dimensione della personalità, anche se tale relazione non viene necessariamente catturata dai classici indici di correlazione statistica.

In sintesi, pur non emergendo correlazioni lineari evidenti, la struttura stessa del test suggerisce una relazione sottostante tra le domande, che riflette il modello dicotomico su cui il test delle 16 personalità si basa.

4. Classificatore per Lettere Custom

In questa sezione viene trattata un'ipotesi formulata inizialmente durante l'analisi preliminare del dataset e del test di myers-briggs, con conseguente analisi dei risultati e considerazioni finali.
È possibile eseguire la porzione di codice riguardante questa analisi nella cartella /utils/dichotomy/.

4.1 Ipotesi

Analizzando le classi del dataset e il funzionamento del test di Myers-Briggs si può constatare che:

- Ogni classe è suddivisa in 4 dicotomie.
- Ogni dicotomia è rappresentata da 2 lettere (E/I, S/N, T/F, J/P) e descrive una caratteristica della personalità di un individuo.
- Le domande sono mirate a descrivere specificamente una o più dicotomie.
- Conseguente all'affermazione precedente, un determinato attributo può avere correlazione diversa tra le dicotomie e ne deriva quindi che descrive in modo diverso la classe totale rispetto ad altri attributi.

Vogliamo quindi ora, calcolare la possibilità di classificare la personalità di un individuo, classificando individualmente le dicotomie che lo descrivono.

Definiamo le dicotomie in classi binarie, dove 1 corrisponde alla prima lettera e 0 alla seconda:

$$E/I = 1 => E$$

$$E/I = 0 => I$$

Dividiamo ogni classe in 4 dicotomie:
e.g. La classe INTP vale:

$$E/I = I = 0$$

$$S/N = N = 0$$

$$T/F = T = 1$$

$$J/P = P = 0$$

$$INTP = 0010$$

È possibile quindi classificare correttamente se non con maggiore accuratezza la personalità di un individuo, con l'utilizzo di questo dataset, nello stesso modo in cui funziona effettivamente il test di Myers-Briggs?

4.2 Classificazione su tutti gli attributi

Un primo approccio a questo problema è stato classificare le singole dicotomie con l'utilizzo di tutti gli attributi (domande) senza considerare la correlazione logica o matematica.

Questo approccio di per sé non ha prodotto risultati eccessivamente negativi, con un albero decisionale di profondità massima uguale a 12, l'accuratezza per ogni dicotomia è di ≈ 84.5%, quella totale invece scende al 52%. Mentre con l'utilizzo di Random Forest l'accuratezza per dicotomia è del ≈ 97.5% e quella totale della classe di circa il 90%.

Questo perché gli errori si accumulano e l'accuratezza della singola dicotomia si propaga all'intera classe, in forma

$$AccuratezzaMedia^4 = AccuratezzaTotale$$

e.g.

$$0.975^4 \approx 0,90$$

4.3 Classificazione sugli attributi logicamente correlati

In questo approccio si è cercato di suddividere seguendo una correlazione logica gli attributi. Questo metodo si può definire il più simile possibile all'originale test di Myers-Briggs. E le aspettative, almeno inizialmente erano alte e il risultato atteso poteva, in una situazione ottimale, rappresentare nel modo più realistico e accurato possibile il vero funzionamento del test.

Tuttavia i risultati sono stati pessimi, questo dovuto sia all'accumulazione dell'errore ma anche alla presenza di diversi valori neutri (0) in determinati attributi e conseguente bassa correlazione matematica con la dicotomia che dovrebbero influenzare.

Ad esempio, la domanda:

"You regularly make new friends."

Dicotomia	E/I	S/N	T/F	J/P
Correlazione	0.006474	0.001916	0.002279	0.000910

Ha una correlazione lineare molto bassa con la dicotomia E/I, nonostante dovrebbe descriverla significativamente dal punto di vista logico.

4.4 Conclusioni

Questo evidenzia come l'applicazione di modelli di Machine Learning non sempre riflette fedelmente il funzionamento reale di un fenomeno. Anzi, la capacità predittiva di un modello è strettamente legata alla qualità e alla disponibilità dei dati, rendendo complesso e, in questo caso, persino svantaggioso tentare di replicare il processo di valutazione umana attraverso algoritmi di Machine Learning: in particolar modo il tentativo di addestrare, forzatamente, i modelli seguendo le stesse regole su cui è basato il test, ha evidenziato una carenza nel rilevare correlazioni significative in modo autonomo.

5. Data Pre-processing

Il dataset selezionato permette l'applicazione di alcune tecniche di data preprocessing. Di seguito l'elenco delle tecniche che potrebbero essere utilizzate, con quelle effettivamente utilizzate nel dataset in esame.

5.1 Aggregazione

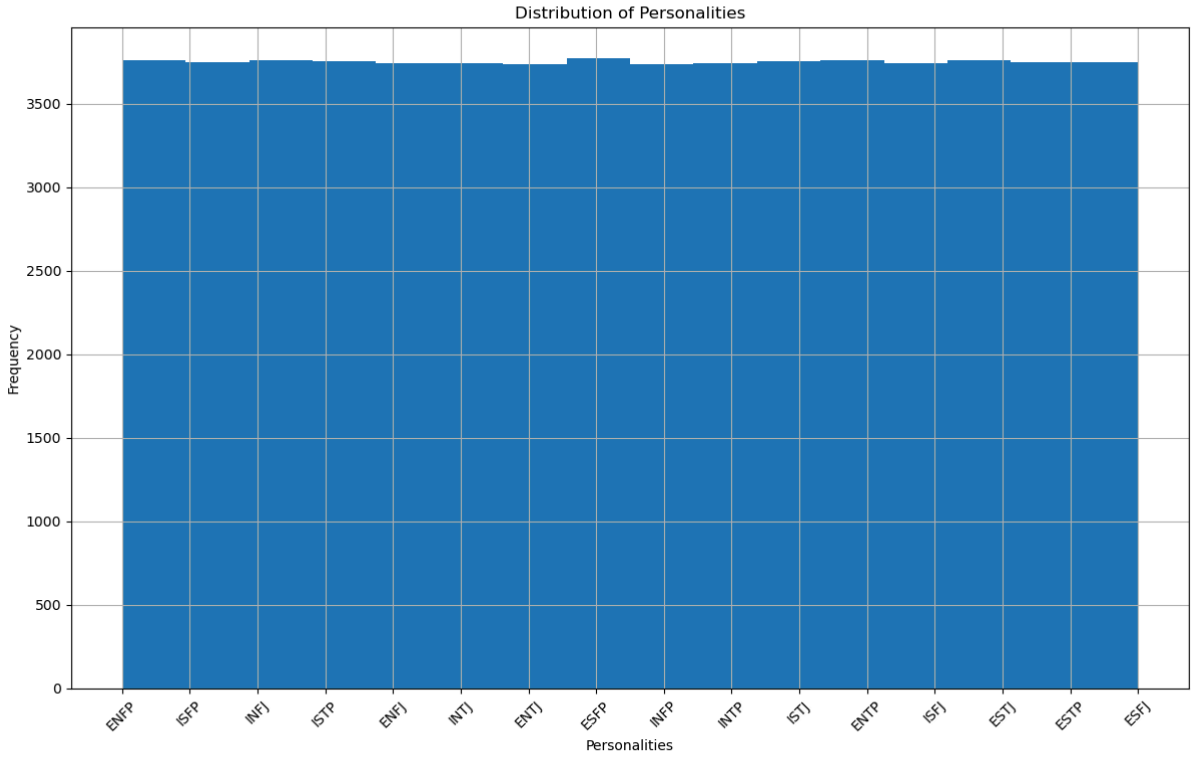
A prima vista, l'applicazione di questa tecnica sembra difficile, poiché le features del dataset sono rappresentate dalle singole domande. Applicare un'operazione di aggregazione significherebbe unire più domande in un'unica feature, riducendo così la dimensionalità del dataset. Tuttavia, questo comporterebbe una modifica significativa alla struttura originale del test e di conseguenza anche al dataset, non potendo garantire gli stessi risultati.

Questa riflessione potrebbe fornire però uno spunto interessante per l'applicazione del machine learning: un modello potrebbe, in qualche modo, riprodurre l'efficacia del test originale, ma superarlo in termini di "forza" o numero di domande elaborate.
Va osservato che l'opportunità di applicare tale tecnica dipende direttamente dall'efficacia di quanto osservato precedentemente. In pratica, potrebbe essere possibile combinare alcune domande (soprattutto quelle con bassa varianza nelle risposte e quindi potenzialmente più fuorvianti) in un singolo quesito, il quale, a sua volta, potrebbe presentare una varianza maggiore.

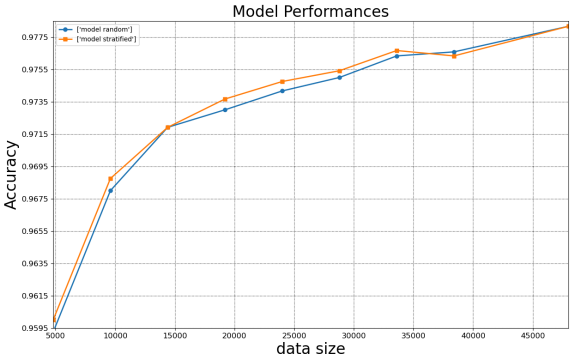
Va sottolineato che questa è una considerazione personale, basata sull'analisi del dataset in questione e perciò non ne viene discussa alcuna implementazione.

5.2 Campionamento

Prima di valutare se il campionamento possa effettivamente ridurre i tempi di computazione senza compromettere l'accuratezza, è stato verificato il bilanciamento del dataset. In particolare, è stata verificata la presenza di classi rappresentate maggiormente.
Il grafico sottostante mostra che le classi sono distribuite in modo quasi uniforme, escludendo la presenza di una classe predominante. Per questa ragione e per un'altra che vedremo fra poco, si è scelto di applicare il parametro stratify ad ogni chiamata di train_test_split() durante tutte le operazioni di training di ogni modello discusso.



Per valutare se sia possibile ottenere un campione di dimensioni ridotte, ma comunque rappresentativo e in grado di garantire la stessa accuratezza nei modelli scelti, è necessario analizzare le curve di apprendimento. Queste mostrano come l'accuratezza del modello che si sta addestrando, cambi in base alla dimensione del training set. A questo scopo, il dataset è stato inizialmente suddiviso in due parti: l'80% dei record è stato assegnato al training set e il restante 20% al test set. Successivamente, il training è stato ripetuto su otto campioni di dimensioni diverse, scalando progressivamente la percentuale di dati utilizzati (dall' 80% fino al 10%). I record sono stati estratti in modo casuale, quindi alcuni campioni possono contenere record in comune.



I grafici soprastanti illustrano l'andamento dell'accuratezza in funzione della dimensione del campione di training: a sinistra kNN, a destra Random Forest. Da entrambi i grafici emerge chiaramente quanto accennato in precedenza: l'incremento dell'accuratezza, pur essendo visibile, risulta estremamente ridotto se confrontato con la scala utilizzata per rappresentare i dati.

I risultati mostrano dunque che, anche riducendo significativamente la dimensione del campione, l'accuratezza del modello rimane quasi invariata; la differenza tra la massima e la minima accuratezza registrata (98.875% e 98.450%) è infatti ≈ 0.420%. Inoltre, si osserva che il campionamento stratificato, in media, offre prestazioni migliori rispetto a quello casuale nel caso in esame. Questo conferma la scelta di adottare un campionamento stratificato, come anticipato in precedenza. Concludendo, si può dire che questi risultati indicano che è possibile ridurre la dimensione del campione per ridurre i costi in fase di training senza compromettere significativamente l'accuratezza del modello. Resta quindi da determinare il miglior compromesso tra la dimensione del campione e l'accuratezza.

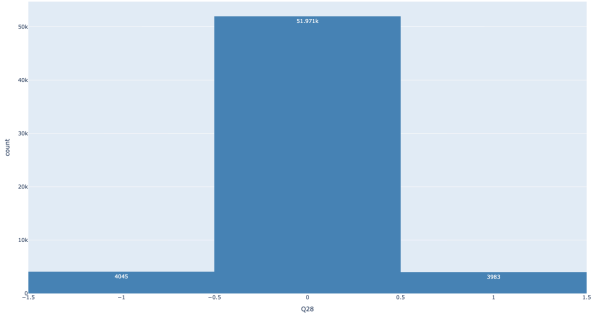
5.3 Creazione degli Attributi

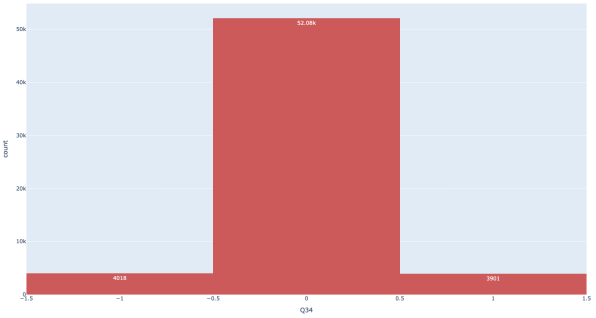
Il caso in esame mostra da sé come non sia necessaria la creazione di nuovi attributi, in primis per il numero elevato di features (60), per motivi legati alla natura stessa del test e all'assenza di competenze in ambito psicologico. Evitare di snaturare il test è fondamentale al fine di preservare e garantire prestazioni ottimali in fase di predizione.
Quest'ultima affermazione però è valida solo per la creazione di nuovi attributi, mentre non lo è, vedremo fra poco, per la riduzione della dimensionalità.

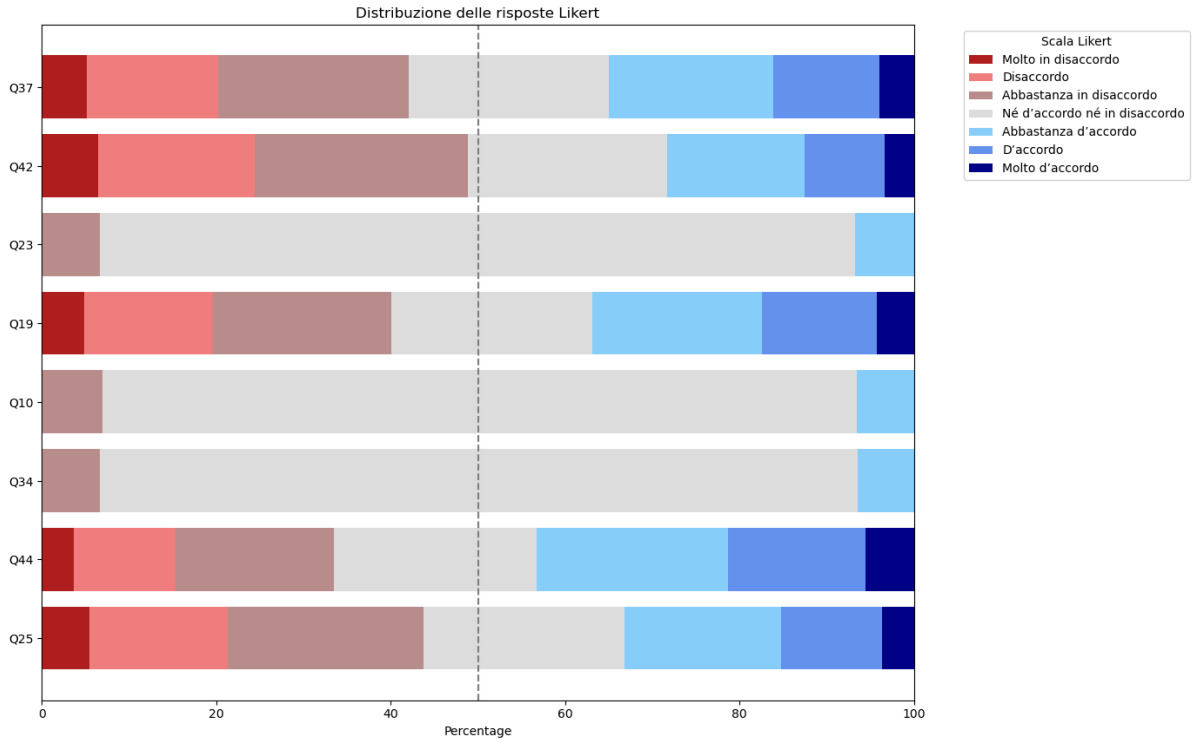
Occam's Razor: La teoria migliore è la più semplice teoria in grado di descrivere i fatti

5.4 Selezione degli Attributi (Features Selection)

L'obiettivo di questa tecnica è eliminare sia gli attributi irrilevanti, come gli ID o altre variabili non utili all'analisi, sia quelli ridondanti, ovvero le features che contengono informazioni derivabili da altre. Per garantire un training più efficace, è stato eseguito un filtraggio manuale delle features. Dall'analisi delle distribuzioni degli attributi, è emerso che molti di essi presentano una distribuzione unimodale con moda pari a 0. Di seguito vengono riportati due grafici che illustrano le distribuzioni di due domande con questa caratteristica. Questi rappresentano solo alcuni esempi di attributi con distribuzioni simili, che possono essere sfruttate per applicare la tecnica di features selection descritta.





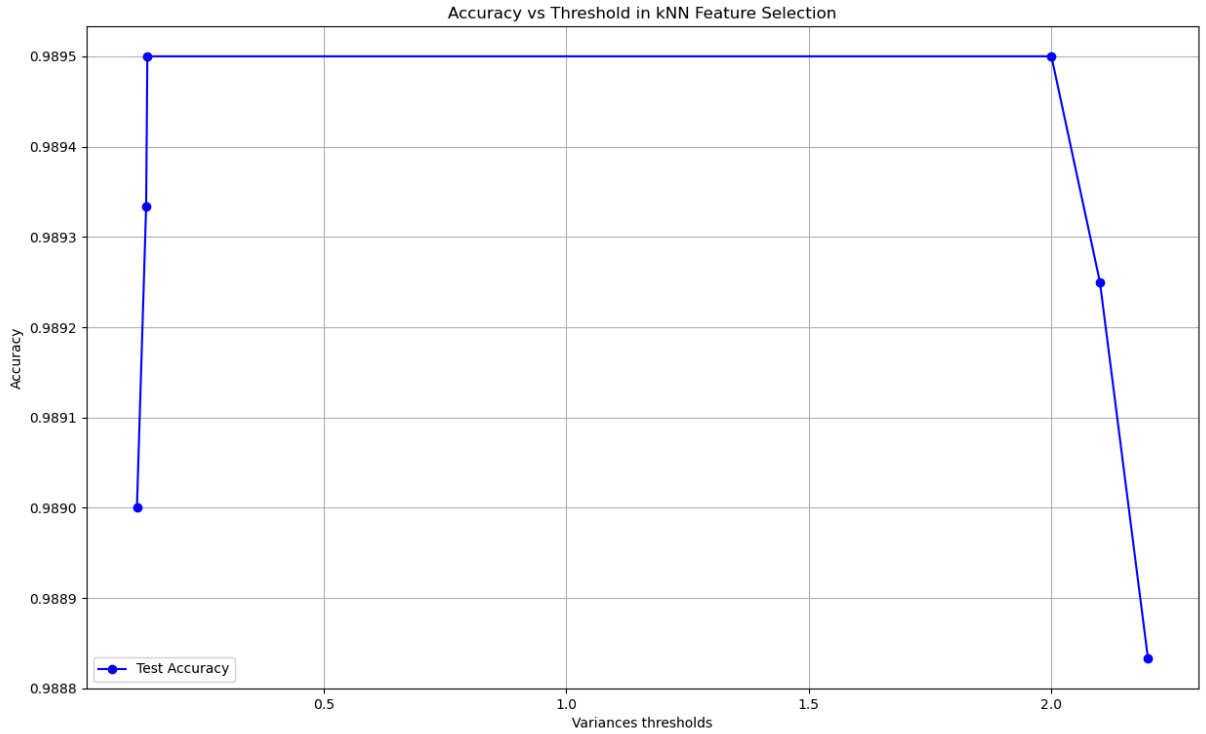


Osservando questo altro plot, è possibile osservare meglio la distribuzione delle risposte per ciascuna domanda, mettendo in evidenza quelle che presentano una bassa varianza, poiché presentano un'alta percentuale di risposte neutre (Né d'accordo né in disaccordo). Per questo motivo, le domande con queste caratteristiche possono essere considerate poco rappresentative.

Sulla base di questa osservazione, è possibile avanzare un'ulteriore ipotesi: *gli attributi con bassa varianza possono essere rimossi poiché forniscono un contributo limitato al modello?* Per verificare questa ipotesi, l'analisi successiva si è concentrata sullo studio delle varianze di ciascun attributo. Come primo riferimento, è stata considerata la mediana, il cui valore è ≈ 2.279; tuttavia, eliminare gli attributi con varianza superiore a questo valore comporterebbe una perdita eccessiva di informazioni, con un conseguente peggioramento delle prestazioni del modello. Osservando i dati, si è notato che le varianze tendono a concentrarsi attorno a due valori principali; per questo motivo, è sorta la necessità di individuare una soglia ottimale di varianza (threshold), in grado di selezionare gli attributi più significativi per la fase di induzione del modello.

Se l'ipotesi appena formulata fosse confermata, ovvero se le prestazioni del modello rimanessero pressoché invariate, sarebbe possibile addestrare un modello utilizzando un numero significativamente ridotto di attributi. Questo risultato sarebbe in linea con il principio del rasoio di Occam, secondo il quale, a parità di prestazioni, è preferibile un modello più semplice.

Per verificare questa ipotesi, sono stati utilizzati diversi algoritmi per creare differenti modelli. Il training è stato eseguito iterativamente variando il valore della varianza minima (threshold) considerata per la selezione delle *features*. In ciascuna iterazione, sono stati mantenuti solo gli attributi con una varianza superiore alla soglia impostata.



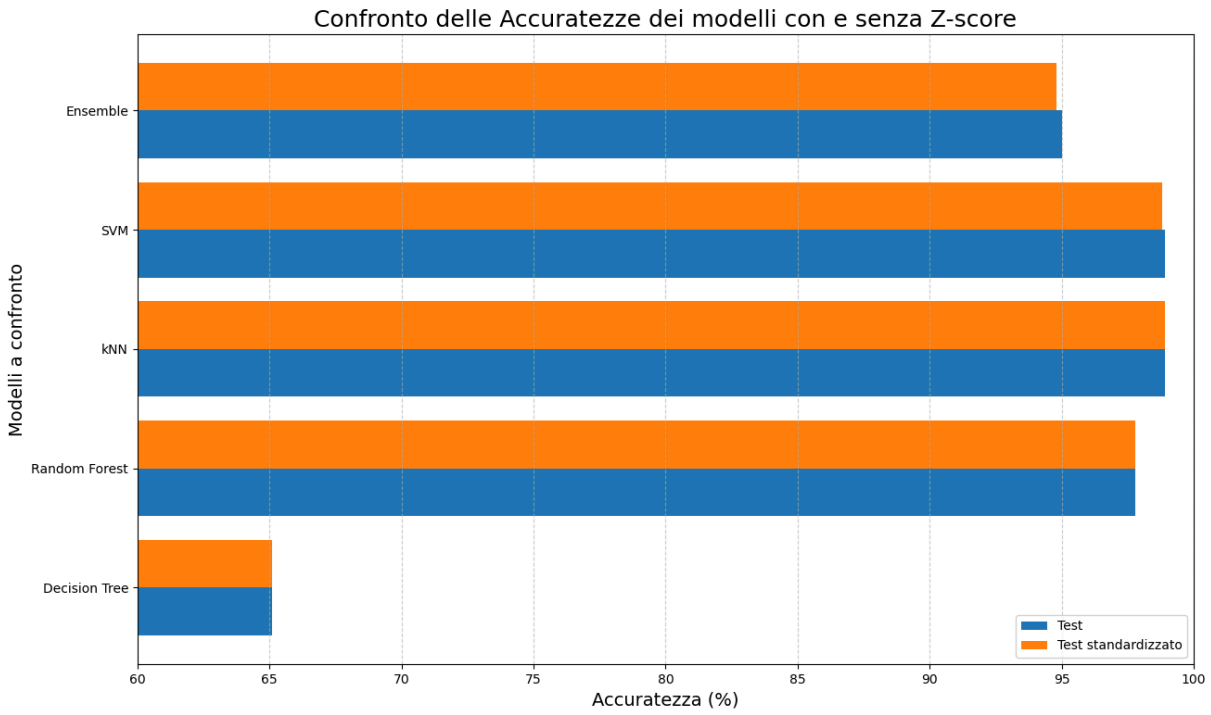
Analizzando il grafico soprastante, si osserva che l'accuratezza del modello converge verso un valore ottimale quando la varianza degli attributi è pari a 2.00. Questo indica che ulteriori riduzioni non migliorerebbero le prestazioni, mentre l'eliminazione di feature con varianza inferiore ha permesso di semplificare il modello senza comprometterne l'efficacia.

In conclusione, è stato selezionato un sottoinsieme di 42 feature, rispetto alle 60 iniziali. Questa riduzione consente di mantenere le stesse prestazioni predittive, riducendo al contempo il costo computazionale del modello.

5.5 Trasformazione degli Attributi

Il dataset è composto da attributi categorici di tipo ordinale, in cui ogni valore rappresenta un grado di risposta associato a una domanda. I valori variano da -3 (totalmente in disaccordo) a 3 (totalmente d'accordo). Data la natura e la distribuzione del dataset, si è ipotizzato che la trasformazione degli attributi non avrebbe apportato miglioramenti significativi alle prestazioni del modello.

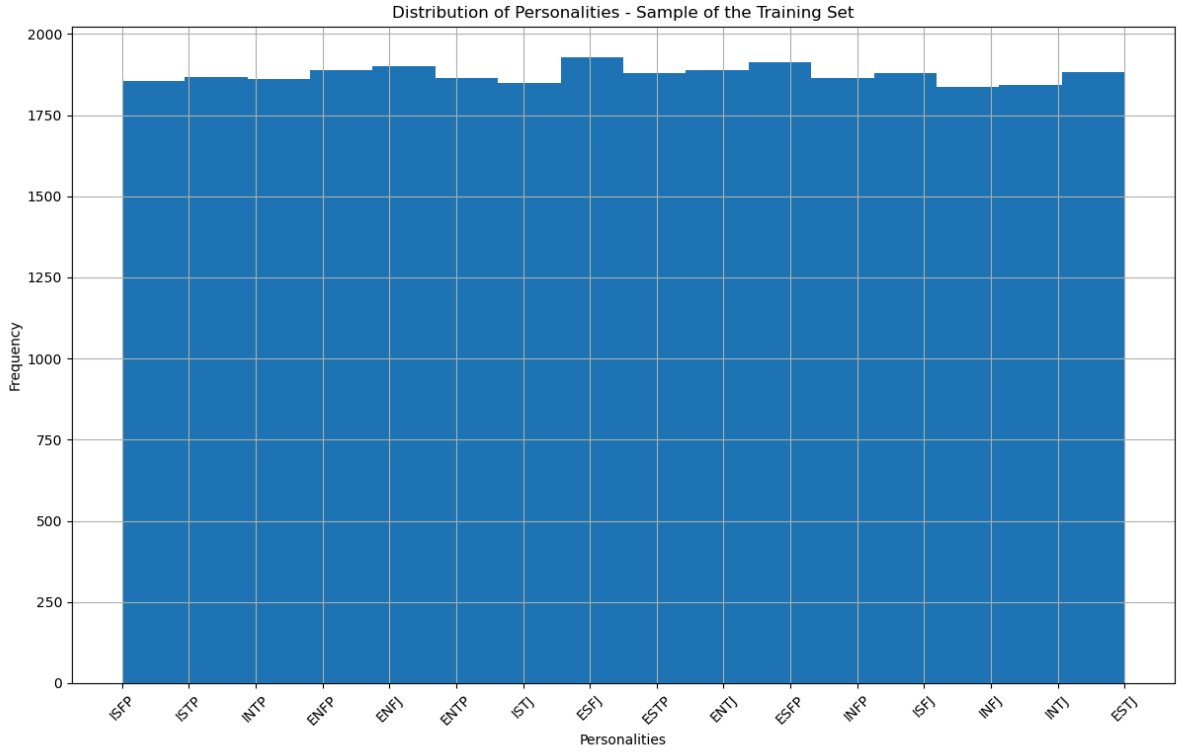
Di fatto il calcolo della media dei valori non standardizzati risulta essere pari a -0.01987 mentre la varianza pari a 1.17147; si evidenziano quindi dei valori molto simili a quelli che caratterizzano una distribuzione normale standard, la quale è caratterizzata da $\mu = 0$ e $\sigma^2 = 1$. Per consolidare questa analisi, sono stati testati alcuni algoritmi con e senza la standardizzazione dei dati, producendo i seguenti risultati.



Osservando il grafico che confronta le prestazioni degli algoritmi in termini di accuratezza, si nota che l'applicazione della standardizzazione non produce variazioni significative nei risultati. Di conseguenza, possiamo concludere che non contribuisce a migliorare le prestazioni dei modelli, confermando l'ipotesi iniziale.

5.6 Bilanciamento del Dataset

In precedenza è stata analizzata la distribuzione delle classi, evidenziando che il dataset risulta già bilanciato. Di conseguenza, l'applicazione di tecniche di undersampling o oversampling sull'intero dataset sarebbe superflua, poiché non influirebbe sulle prestazioni degli algoritmi. Per testare comunque l'efficacia di queste tecniche, sono stati estratti campioni casuali al fine di simulare uno scenario di dataset sbilanciato.



In seguito a questo sbilanciamento è stato possibile applicare tecniche di oversampling, la quale, dopo aver individuato la classe minoritaria, genera dei record sintetici per bilanciare il numero di record posseduti dalla classe maggioritaria. In maniera equivalente è possibile applicare tecniche di undersampling, i quali mettono in risalto la classe minoritaria, rimuovendo un numero di record dalle classi maggioritarie, eguagliando la numerosità appartenente alla classe minoritaria.

Sono stati scelti due metodi per ciascuna delle tipologie descritte:

- SMOTE: genera nuovi esempi sintetici creando punti interpolati tra un record della classe minoritaria e i suoi vicini..
- Random Oversampling: è una tecnica più semplice rispetto a SMOTE e consiste nell'aumentare la quantità di record nella classe minoritaria duplicando casualmente gli esempi esistenti.
- Cluster Centroids Undersampling: questa tecnica raggruppa i campioni della classe maggioritaria in cluster, sostituendo i campioni appartenenti ad esso col rispettivo centroide.
- Random Undersampling: quest'ultima tecnica riprende il principio della seconda descritta, rimuovendo in questo caso randomicamente dei record dalla classe maggioritaria per eguagliare la minoritaria.

Tutte le tecniche appena descritte sono stati testati sull'algoritmo kNN, scelto perché le analisi hanno evidenziato un buon rapporto tra accuratezza e tempi computazionali.

6. Decision Tree

Il Decision Tree è stato il primo algoritmo che ha prodotto un modello in tempi ragionevoli, scelto per la sua interpretabilità e la capacità di modellare relazioni non lineari nei dati.

6.1 Scelta degli iperparametri

Il tuning si è concentrato principalmente sul parametro max_depth, che controlla la profondità massima dell'albero e di conseguenza, il livello di dettaglio con cui il modello apprende le relazioni nel dataset. Dai test effettuati è emerso che, aumentando la profondità dell'albero, l'accuratezza migliora progressivamente fino a raggiungere un livello di stabilità per valori di max_depth ≥ 13. Oltre questa soglia, ulteriori incrementi di profondità non hanno portato a miglioramenti significativi nelle prestazioni. L'accuratezza massima registrata è intorno al 66%, suggerendo che il Decision Tree non è adatto considerando la struttura del dataset, al fine di ottenere predizioni altamente affidabili.

6.2 Addestramento e analisi

Un'analisi più approfondita ha dimostrato che le prestazioni relativamente limitate del modello derivano dalla natura degli attributi disponibili. Nel dataset utilizzato, nessuna feature assume un ruolo dominante sulle altre: tutte le variabili contribuiscono in modo relativamente equilibrato alla classificazione, senza che emergano attributi particolarmente discriminanti. Questa caratteristica riduce l'efficacia del Decision Tree, che si basa su divisioni gerarchiche per creare regole di classificazione ottimali. In mancanza di attributi fortemente influenti, l'albero fatica a costruire strutture decisionali altamente efficaci.

6.3 Risultati

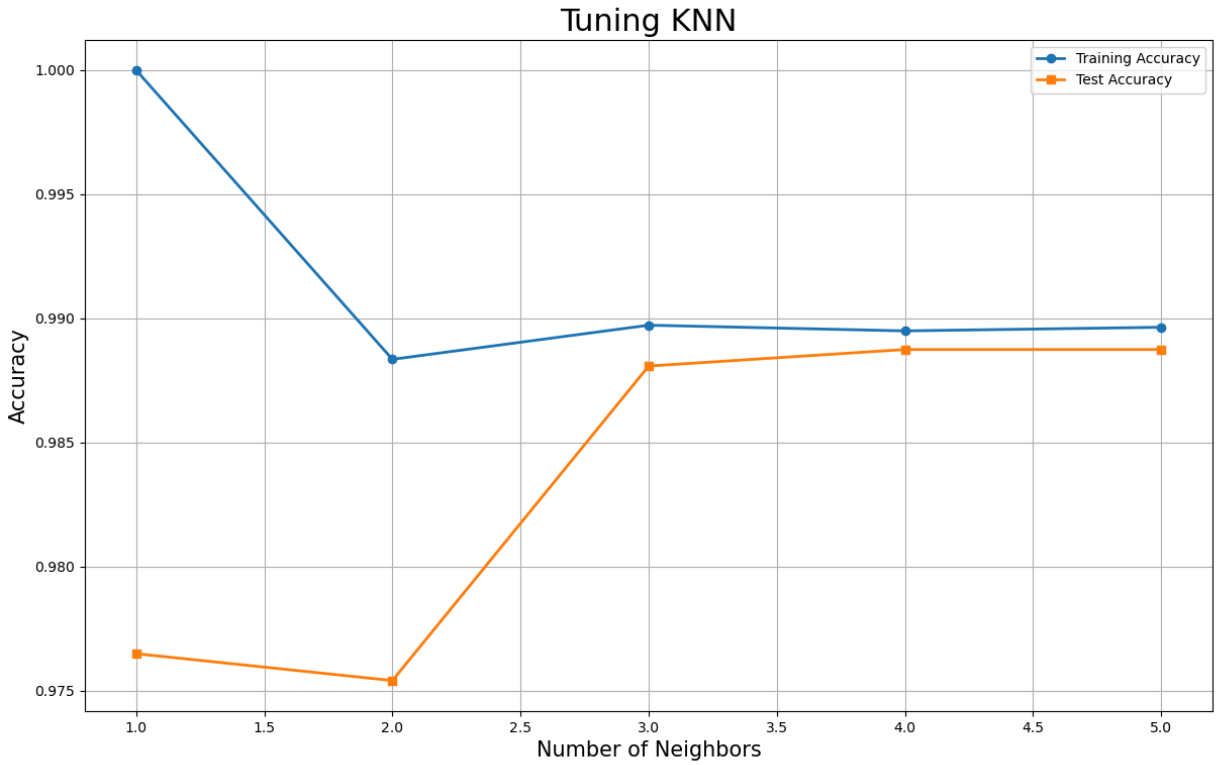
Per questi motivi, nonostante la semplicità del Decision Tree, il modello non è stato incluso tra gli algoritmi selezionabili nel menu di test. La scelta è ricaduta su altri classificatori in grado di offrire un'accuratezza sensibilmente superiore e una maggiore capacità di generalizzazione sui dati di test.

7. K-Nearest-Neighbors

L' algoritmo K-Nearest-Neighbors si è dimostrato particolarmente efficiente. La natura del dataset favorisce questo tipo di approccio dato che i valori possibili sono racchiusi nel range (-3 ; 3). Inoltre i record possiedono un'alta similarità, comportando una maggiore accuratezza nella predizione della classe dei record di test. Si è ipotizzato quindi che personalità simili tendano a posizionarsi vicine nello spazio delle *features*, permettendo al kNN di identificare con efficacia i pattern tra le varie classi.

7.1 Scelta degli iperparametri

Dato il funzionamento dell'algoritmo kNN, è stato analizzato l'andamento dell'algoritmo applicando il tuning sull'iperparametro k, ossia il numero di vicini scelti per classificare il nuovo record. Si è seguito un approccio iterativo, tenendo a mente il rapporto tra costo computazionale e prestazioni; dall'analisi è emerso che l'accuratezza non varia notevolmente con valori di k >= 4, bensì si stabilizza in un range molto piccolo.



Tenendo conto dell'incremento del tempo computazionale in funzione del parametro k, è stato selezionato il valore k=4 (all'interno del range analizzato) come miglior iperparametro. Questo valore consente di ottenere un'accuratezza elevata (98,9%), riducendo al contempo la complessità computazionale grazie a un numero relativamente basso di vicini.

7.2 Addestramento e Validazione

Una volta identificato il valore ottimale di k, è stata approfondita, così come per gli altri algoritmi, la possibilità di ridurre la dimensionalità del dataset, applicando la features selection per le ragioni descritte sopra. La riduzione del numero di features, da 60 a 42, si è rivelata efficace: pur non modificando il potere di generalizzazione del modello, ha permesso di ottenere un significativo risparmio in termini di costo computazionale, grazie alla diminuzione del dataset di quasi un terzo del totale.

7.3 Risultati

L’algoritmo kNN ha raggiunto ottime prestazioni, sia in termini di accuratezza delle predizioni, con un’accuratezza in fase di test del 98,9%, che di efficienza computazionale. In particolare, è riuscito a superare, nella fase di predizione, alcuni algoritmi con tempi di addestramento e classificazione significativamente più elevati.

8. Random Forest

Poiché l'algoritmo di decision tree non ha fornito prestazioni ottimali, si è deciso di valutare se l'accuratezza potesse migliorare combinando più alberi decisionali in un modello di random forest. L'obiettivo è dunque verificare se questa strategia possa portare a un incremento significativo delle prestazioni.

8.1 Scelta degli Iperparametri

Oltre al numero di alberi (n_estimators), principale iper-parametro dell'algoritmo, è stato applicato il tuning anche a max_depth, min_samples_split e min_samples_leaf tramite la funzione GridSearch di scikit-learn. Tenendo sempre a mente il rapporto prestazioni e complessità computazionale, sono stati ottenuti i seguenti valori ottimali per gli iperparametri:

- n_estimators: 200
- max_depth: 16
- min_samples_split: 2
- min_samples_leaf: 1

Si noti che le prestazioni dell'algoritmo sono influenzate principalmente dai primi due parametri, rispettivamente il numero di alberi del classificatore e la loro profondità; modificando gli altri due parametri infatti (min_samples_split e min_samples_leaf) si registrano cambiamenti poco significativi.

8.2 Addestramento e Validazione

L'algoritmo viene addestrato tramite cross-validation, applicato per 5 volte, combinando iterativamente i valori degli iperparametri. In fase di predizione le probabilità assegnate dai singoli classificatori vengono combinate.

8.3 Risultati

L'idea alla base di utilizzare questo algoritmo per migliorare le prestazioni del singolo albero decisionale si è dimostrata corretta. Chiaramente la soluzione ottimale da noi scelta tramite il tuning potrebbe non rappresentare le reali prestazioni massime, tuttavia si riesce ad ottenere un ottimo livello di accuratezza del 97.8%.

9. Support Vector Machine (SVM)

Una degli algoritmi scelti è SVM (Support Vector Machine) per la sua versatilità, soprattutto con dataset ad alta dimensionalità come quello scelto. La natura del dataset si è rivelata favorevole, poiché le classi risultano ben separabili e distribuite in modo uniforme.

9.1 Scelta degli Iperparametri

Per ottimizzare il modello, è stata utilizzata la funzione GridSearchCV() della libreria Scikit-Learn, che esegue automaticamente il tuning degli iperparametri forniti in input.

- Kernel: è stato scelto RBF (Radial Basis Function), in quanto altamente flessibile e capace di catturare relazioni non lineari o complesse, una caratteristica essenziale per dataset con un numero elevato di features.

- Parametro C (Regolarizzazione) sono stati testati diversi valori:

  - 0.1 → per valutare il comportamento del modello con un margine decisionale più ampio.
  - 10 e 100 → per osservare le prestazioni con una bassa tolleranza agli errori.
  - 1 → si è rivelato il valore migliore, offrendo un buon compromesso tra accuratezza e generalizzazione.

- Parametro ϒ (Gamma)
  Sono stati testati valori a priori ([0.01, 0.1]).

  È stata utilizzata anche l'opzione scale di *GridSearchCV(),* che calcola automaticamente il valore di ϒ in base alla scala dei dati, secondo la formula:

$$\Upsilon = \frac{1}{n_{features} \times X.var().mean()}$$

dove *n_features* è il numero di attributi e *X.var().mean()* la media delle varianze delle features. Il valore ottimale risultante dal tuning è stato circa 0.275.

9.2 Addestramento e Validazione

L’addestramento è stato effettuato utilizzando una Cross-Validation con cv = 3, ovvero suddividendo i dati in tre diverse combinazioni tra training e test, selezionando quella che garantisce le migliori prestazioni.

9.3 Risultati

L’analisi dei risultati mostra un livello di accuratezza molto elevato in tutte le fasi dell'esecuzione dell’algoritmo, sia con l'applicazione della Feature Selection sia mantenendo tutte le feature. In particolare l’utilizzo o meno della tecnica di feature selection non comporta delle perdite sensibili di accuratezza che rimane al 98,8%. Tuttavia, l’utilizzo delle sole feature più rilevanti consente di ridurre il costo computazionale, garantendo ugualmente ottime prestazioni predittive.

10. Naive Bayes Custom

Come algoritmo custom, si è deciso di implementare un Naive Bayes. Questo algoritmo non è stato scelto tra quelli messi a disposizione della libreria di scikit-learn (precedentemente descritti), per questo motivo si era interessati ad analizzare le prestazioni di quest'ultimo visto l'approccio probabilistico utilizzato per la classificazione dei dati.

10.1 Addestramento e Validazione

Si è suddiviso l'algoritmo Naive Bayes nei seguenti punti e nelle relative funzioni:

- *get_class_proba*: questa funzione calcola la probabilità a priori di ciascuna classe nel dataset, ovvero la frequenza relativa con cui ciascuna classe appare nei dati di addestramento.
- *get_att_proba*: calcola la probabilità condizionata dei singoli attributi rispetto alla classe di appartenenza. Per ottenere questa probabilità, si utilizzano la media e la varianza di ogni attributo per ogni classe, assumendo che i dati seguano una distribuzione normale (dimostrato nel punto 4.5).
- *predict*: viene predetta la classe di appartenenza per ogni record del test. Vengono prima classificate tutte le probabilità condizionate per ogni singola classe, successivamente viene selezionata la maggiore, ossia quella predetta per il record in esame. Il calcolo viene eseguito tramite l'utilizzo della funzione di densità di probabilità.
- *Classificatore Naive Bayes*: l'algoritmo finale combina le informazioni fornite dalle due funzioni precedenti per calcolare la probabilità a posteriori di ciascuna classe dato un'osservazione e assegnare l'etichetta corrispondente alla classe con probabilità maggiore.

10.2 Risultati

L'implementazione di questo algoritmo sul dataset in studio ha permesso di ottenere un'accuratezza massima di circa 91%, dimostrando così la validità del Naive Bayes come metodo di classificazione efficace e relativamente semplice da implementare.
Viste le caratteristiche del dataset si sarebbe potuto implementare l'algoritmo basandosi solamente sui valori presenti nel dataset (da -3 a 3), ottenendo però delle restrizioni per quanto riguarda l'utilizzo di tale metodo con dati continui, che essi siano interi o decimali. Con l'implementazione tramite media e varianza, utilizzate per ottenere la funzione di densità di probabilità, è possibile calcolare le probabilità condizionate a prescindere dal tipo di dato analizzato.

11. Classificatore multiplo custom

È stato sviluppato un classificatore multiplo custom composto da tre algoritmi di base: k-Nearest Neighbors (k-NN), Naive Bayes e Decision Tree. Questa strategia permette di migliorare la robustezza e l'affidabilità delle predizioni, sfruttando le predizioni di più modelli diversi tra loro. In particolare, la scelta di implementare il Decision Tree tra i modelli di base è utile per evidenziare le differenze nelle prestazioni dei singoli classificatori, i quali presentano livelli di accuratezza diversi sul test set. Quest'ultima considerazione è stata consolidata effettuando diverse prove, scegliendo diverse combinazioni dei parametri del metodo majority voting, descritto successivamente.

11.1 Scelta degli Iperparametri

Nella costruzione del classificatore multiplo, non è stato effettuato un tuning specifico degli iperparametri per i tre algoritmi di base che lo compongono. Questo perché il processo di ottimizzazione degli iperparametri è già stato eseguito nelle sezioni precedenti del progetto, dove ciascun algoritmo è stato utilizzato singolarmente. Pertanto, nella combinazione dei modelli, ogni algoritmo è stato addestrato con gli iperparametri ottimali già individuati e testati in quelle fasi.

In particolare, per l'algoritmo Decision Tree è stata scelta una profondità massima (*max_depth*) pari a 13. Sebbene il modello, preso singolarmente, non mostri un'elevata capacità di generalizzazione, i test precedenti hanno evidenziato che tale valore rappresenta un buon compromesso, infatti aumentando ulteriormente la profondità, le prestazioni a livello predittivo non miglioravano significativamente.

Analogamente, per il k-Nearest Neighbors è stato adottato un valore di k=4, determinato in precedenza attraverso il tuning specifico condotto nella relativa sezione del progetto. Questo valore ottimale è stato mantenuto anche nella combinazione dei modelli.

Infine, per l'algoritmo Naive Bayes non è stato necessario selezionare particolari iperparametri. È stata direttamente utilizzata la funzione *GaussianNB()*, poiché la distribuzione del dataset segue in modo approssimato una distribuzione normale, infatti come già discusso nelle sezioni precedenti, la media del dataset è pari a -0.01987 e la varianza a 1.17147, valori molto vicini a quelli standardizzati di una distribuzione normale $N(\mu=0, \sigma=1)$.

11.2 Addestramento e Validazione

Per combinare le predizioni dei singoli modelli è stata implementata una procedura di majority voting, che stabilisce la label finale basandosi sul consenso tra le risposte dei classificatori singoli. Sono stati quindi sviluppati quattro schemi di voting, selezionabili in base ai parametri di input, per garantire flessibilità nel processo decisionale.
I metodi di voting si differenziano per l'utilizzo di hard e soft voting, i quali possono essere pesati o non pesati. Questi casi vengono gestiti dai parametri “voting” e “w” che corrispondono rispettivamente al tipo di voting e ai possibili pesi applicati.

Per implementare le funzioni custom necessarie alla creazione di un classificatore multiplo, è stata sviluppata la classe Ensemble, che gestisce le operazioni principali di addestramento e predizione. Nello specifico le fasi di addestramento e di predizione sono implementate nei seguenti metodi:

- fit: per ogni modello da addestrare viene suddiviso il training set di input in un ulteriore sub training set, sul quale viene successivamente addestrato il singolo classificatore tramite la classica funzione di fit.

- predict: per ogni record del test, raccoglie le probabilità fornite da ciascun classificatore e le combina secondo lo schema di majority voting scelto, la probabilità maggiore viene salvata in una lista che contiene le classi predette del test set.

Le predizioni vengono infine salvate per consentire un’analisi approfondita delle prestazioni complessive dell’ensemble, con l’obiettivo di sfruttare la complementarità dei modelli e raggiungere un bilanciamento ottimale tra accuratezza e capacità di generalizzazione.

11.3 Risultati

Indipendentemente dal tipo di voting scelto, le accuratezze ottenute risultano essere particolarmente positive. In particolare tra le due macro tipologie di voting, ossia hard e soft, risulta essere più precisa la seconda tipologia a parità di peso, superando la prima del 2%. Di seguito i risultati ottenuti:

	Hard voting peso uniforme	Hard voting pesato [2, 1, 2]	Soft voting peso uniforme	Soft voting pesato [2, 1, 2]
Accuratezza sul test set	0.951	0.956	0.971	0.978

Nel caso del voting pesato, i pesi sono stati assegnati in base alle prestazioni degli algoritmi utilizzati. In particolare, è stato attribuito un peso di 2 ai modelli addestrati con k-NN e Naive Bayes, poiché mostrano un'accuratezza significativamente superiore rispetto all'albero decisionale. Invertendo i pesi (assegnando peso 2 all'albero decisionale e 1 agli altri due modelli), le prestazioni peggiorano sensibilmente: in entrambi i tipi di voting, l'accuratezza sul test set scende all'81%. Aumentando ulteriormente il peso dell'albero decisionale, impostando 3 come valore, si ottiene un'accuratezza del 0.630, ossia le prestazioni conseguite in media dal singolo albero decisionale.

12. Classificatori a confronto

In quest'ultima sezione dedicata ai classificatori, viene presentata una tabella riassuntiva che mostra le accuratezze ottenute dai singoli modelli dopo l'applicazione delle diverse tecniche di pre-processing.. È necessario precisare che la tecnica di bilanciamento mostrata in tabella, ossia la SMOTE, è stata scelta in quanto si è dimostrata la più efficace tra quelle analizzate, garantendo le migliori prestazioni.

Accuratezze sul Test set	kNN	Random Forest	SVM	Naive Bayes	Ensemble Classifier
Iperparametri calibrati	0,989	0,978	0,988	0,912	0,950
Standardizzazione	0,989	0.978	0,988	0,912	0,948
Feature selection	0,988	0.978	0.989	0.913	0.955
SMOTE	0.986	0.968	0.987	0.903	0,943

La prima caratteristica che risalta all'occhio è la quasi invariabilità dell'accuratezza dei singoli classificatori rispetto alla tecnica di pre-processing adottata. Questo fatto può essere attribuito a diverse motivazioni: il dataset si trova in una forma bilanciata, gli attributi assumono valori in un range ristretto e la distribuzione generale dei valori segue l'andamento di una distribuzione normale. Queste proprietà favoriscono una buona accuratezza di classificazione, raggiungendo picchi che sfiorano il 99% nel caso dei classificatori kNN e SVM. Tuttavia, ciò comporta un impatto non particolarmente tangibile di alcune tecniche di pre-processing, come evidenziato in particolare nella standardizzazione, il cui effetto sulle prestazioni risulta pressoché nullo.

13. Conclusioni Finali

L'analisi si concentra sull'individuazione della combinazione ottimale tra modello di classificazione e tecniche di pre-processing applicabili al dataset in esame. Poiché il dataset risulta bilanciato, la simulazione di uno sbilanciamento introdurrebbe un' alterazione artificiale del processo di ottimizzazione, pertanto l'attenzione sarà rivolta esclusivamente alle tecniche di feature selection e sampling.

Inoltre, tra i cinque classificatori considerati, tre evidenziano prestazioni di base significativamente inferiori rispetto agli altri e, di conseguenza, vengono esclusi dall'analisi, anche in considerazione dell'elevato numero di combinazioni da testare e dei conseguenti tempi di computazione. Lo studio si focalizzerà dunque sul classificatore k-Nearest Neighbors (kNN), analizzando le accuratezze ottenute mediante la combinazione delle tecniche di pre-processing sopra menzionate.

kNN	Feature Selection: 60 selezionati	Feature Selection: 42 selezionate
Sampling: 5999 (10% TOT)	98.80833%	98.80000%
Sampling: 35999 (60% TOT)	98.84167%	98.83333%
Sampling: 47999 (80% TOT)	98.87500%	**98.87500%**

I risultati ottenuti evidenziano una variazione complessivamente ridotta nelle accuratezze registrate durante l'addestramento del modello k-Nearest Neighbors (kNN) con iperparametri già ottimizzati, fissando k=4k = 4k=4 e combinando le diverse tecniche di pre-processing considerate. Tale stabilità nei risultati suggerisce che le tecniche impiegate non incidano in modo significativo sulle prestazioni del modello in termini di accuratezza, sebbene possano influenzare altri aspetti del processo di apprendimento, come l'efficienza computazionale.

La scelta finale del modello ricade su una combinazione che prevede l'applicazione della *feature selection* e un *sampling* pari all'80% del totale dei record disponibili (circa 47999).
Questa decisione è motivata dal fatto che, a parità di numero di record utilizzati per il training, l'accuratezza ottenuta risulta identica sia applicando la *feature selection* sia omettendola. Tuttavia, l'inclusione della *feature selection* comporta un vantaggio significativo in termini di riduzione dei tempi computazionali, rendendo l'addestramento del modello più efficiente senza compromettere le prestazioni predittive.