Dipartimento di Informatica

Corso di Laurea Triennale in Informatica Applicata e Data Analytics

Relazione Progetto Machine Learning

Docente:
Prof. Lorenzo Putzu

Studenti:
Cariccia Marco
Stampatori Gabriele
Sumatra Matteo
Viglietti Marta
Zuddas Filippo Simone

Anno Accademico 2024/2025

Indice

1 Introduzione

I dati presenti all'interno del dataset provengono dal sito World Population Review, che a sua volta fa uso di dati provenienti dalle Nazioni Unite e dagli istituti di statistica di ciascuna nazione. Il dataset contiene principalmente informazioni relative ai valori della popolazione negli stati e nei territori del mondo, ed alcune informazioni di categoria più generale come il continente, l'area, la capitale e la densità abitativa nella nazione. Il dataset è quindi composto da record ciascuno descritto da 17 attributi, ed è raggiungibile nel sito web.

2 Analisi dei dati

Questo capitolo si occupa dell'analisi dei dati del dataset e della valutazione della loro qualità.

2.1 Descrizione

Il dataset è una matrice con 234 righe, una per record, e 17 colonne. 16 di queste colonne sono dedicate agli attributi, dei quali: 3 sono categorici ("*CCA3*", "*Country/Territory*", "*Capital*"), 3 sono *float* ("*Density*", "*Growth Rate*", "*World Population Percentage*"), mentre gli altri sono interi. Si hanno quindi solo 2 attributi continui, al contrario della maggioranza discreta. Inoltre, non sono presenti dati mancanti.

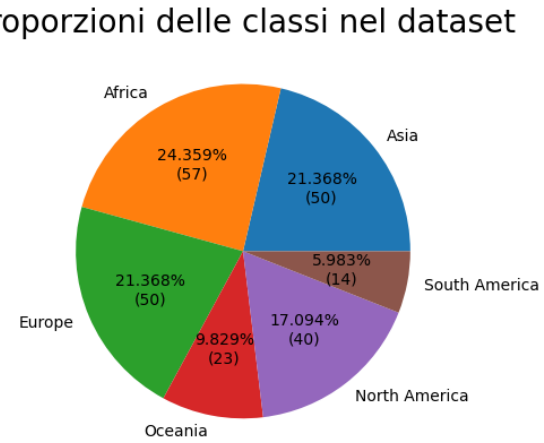
Gli attributi numerici sono di tipo ratio, mentre i categorici sono di tipo nominale – non è presente un ordine intrinseco nei nomi delle capitali e delle nazioni né negli acronimi usati per rappresentarle. Il vettore delle classi ("*Continent*") è composto da 6 classi, ciascuna rappresentante il continente di appartenenza della nazione o del territorio.

Il significato di ogni attributo è descritto nella tabella riportata:

Nome dell'attributo	Tipo	Descrizione	Unità
Rank	Discreto	Classifica sulla base della popolazione	
CCA3	Nominale	Codifica a 3 lettere dei nomi delle nazioni e dei territori	
Country/Territories	Nominale	Nome delle nazioni e dei territori	
Capital	Nominale	Nome della capitale	
2022 Population	Discreto	Popolazione nell'anno 2022	
2020 Population	Discreto	Popolazione nell'anno 2020	
2015 Population	Discreto	Popolazione nell'anno 2015	
2010 Population	Discreto	Popolazione nell'anno 2010	
2000 Population	Discreto	Popolazione nell'anno 2000	
1990 Population	Discreto	Popolazione nell'anno 1990	
1980 Population	Discreto	Popolazione nell'anno 1980	
1970 Population	Discreto	Popolazione nell'anno 1970	
Area	Discreto	Superficie	Km^2
Density	Continuo	Densita della popolazione	Ab./ Km^2
Growth Rate	Continuo	Tasso di crescita	
World Population Percentage	Continuo	Percentuale della popolazione mondiale	%

2.2 Risultati

Da una prima analisi emerge che si tratta di un dataset sbilanciato, in quanto si hanno: 57 record appartenenti alla classe "*Africa*" (24.36%), 50 alla classe "*Asia*" (21.37%), 50 alla classe "*Europe*" (21.37%), 40 alla "*North America*" (17.09%), 23 alla "*Oceania*" (9.83%) e 14 alla "*South America*" (5.98%). Si nota infatti una differenza significativa nel numero di record appartenenti alle varie classi, ma questo squilibrio non è causato dal metodo di raccolta dati, in quanto è noto che siano queste le suddivisioni geografiche delle nazioni.



L'analisi dei dati è stata eseguita tramite la funzione *data_analysis()* tratta dal file dataAnalysis.py, costruita appositamente per questo dataset. Questa lo prende in ingresso e, per ogni attributo numerico calcola media, moda, mediana, deviazione standard, varianza, range, ed i percentili del 10%, 25%, 50%, 75%, 90%. Gli attributi categorici sono stati inclusi nell'analisi a seguito di una conversione a valori numerici – necessaria per poterne analizzare la correlazione con gli altri. Questi dati vengono poi stampati sul terminale dalla funzione *print_analysis()*.

Nel file sono presenti anche altre tre funzioni, relative alla creazione dei grafici utili all'analisi del dataset. *boxplot()* crea una serie di box plot dei 13 attributi numerici. Per consentire una lettura più agevole dei grafici la funzione li stampa 2 volte – nella prima non considera gli outlier, consentendo di visualizzare il centro e la distribuzione dei dati; nella seconda vengono rappresentati anche i valori estremi, per identificarli e rendere evidenti gli effetti. La funzione *histogram()* genera un subplot con 13 istogrammi, che consentono di analizzare l'andamento di ciascun attributo in base alla classe di appartenenza – fondamentale per individuare gli attributi che possono contribuire di più nei modelli. La terza, *pie_chart()* genera un grafico a torta per rappresentare la percentuale delle classi presenti nel database, e si appoggia sulla funzione *pct_chart()* per una stampa migliore delle stesse.

I risultati della funzione *data_analysis()* sono disponibili nella tabella seguente.

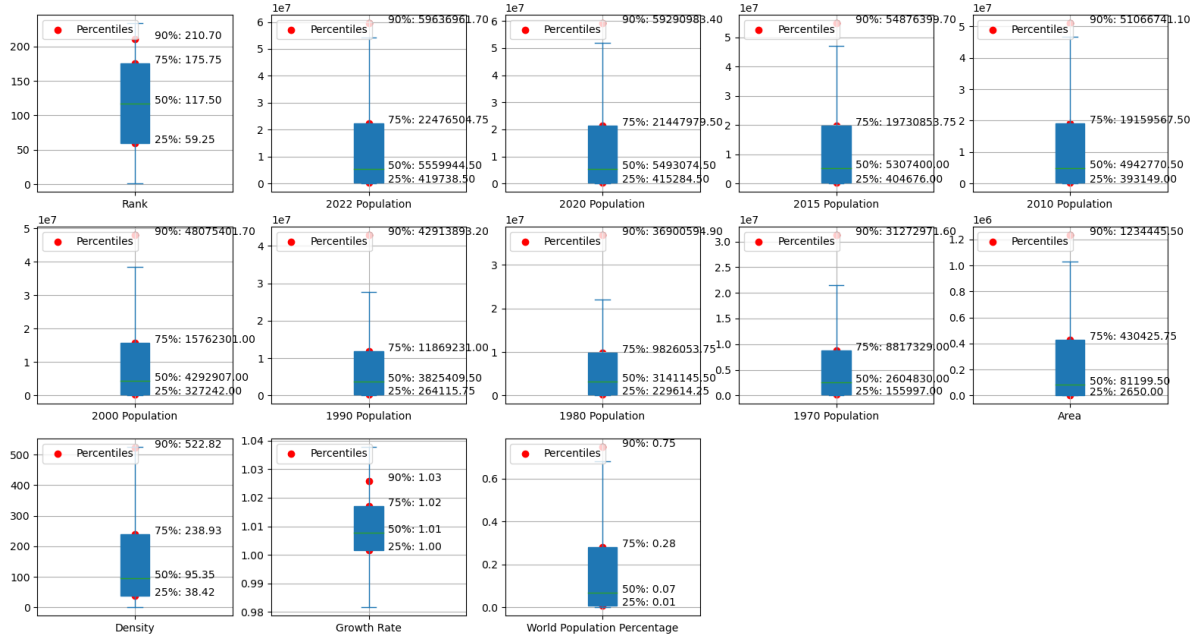
	Rank	CCA3	Country/Territory	Capital	2022 Population	2020 Population	...	1980 Population	1970 Population	Area	Density	Growth Rate	World Population Percentage
Mean	117.5	116.5	116.5	116.5	34074414.709402	33501070.952991	...	18984616.970085	15786908.807692	581449.384615	452.127044	1.009577	0.427051
Mode	[1, 1]	[0, 1]	[0, 1]	[0, 1]	[510, 1]	[520, 1]	...	[733, 1]	[752, 1]	[21, 2]	[0.0261, 1]	[1.0038, 4]	[0.0, 57]
Median	117.5	116.5	116.5	116.5	5559944.5	5493074.5	...	3141145.5	2604830.0	81199.5	95.34675	1.0079	0.07
Standard Deviation	67.694165	67.694165	67.694165	67.694165	136766424.804763	135580876.924439	...	81785186.084201	67795891.643236	1761840.864063	2066.121904	0.013385	1.714977
Variance	4582.5	4582.5	4582.5	4582.5	18705054953876844.0	18384614724384476.0	...	6688816662827466.0	4596174458914722.0	3104083230282.754395	4268859.720555	0.000179	2.941145
Range	233	233	233	233	1425866827	1424929261	...	982371733	822533698	17098241	23172.2406	0.1571	17.88
10%	24.3	23.3	23.3	23.3	48225.2	48225.5	...	32257.4	24994.5	247.4	15.83728	0.9964	0.0
25%	59.25	58.25	58.25	58.25	419738.5	415284.5	...	229614.25	155997.0	2650.0	38.417875	1.001775	0.01
50%	117.5	116.5	116.5	116.5	5559944.5	5493074.5	...	3141145.5	2604830.0	81199.5	95.34675	1.0079	0.07
75%	175.75	174.75	174.75	174.75	22476504.75	21447979.5	...	9826053.75	8817329.0	430425.75	238.93325	1.01695	0.28
90%	210.7	209.7	209.7	209.7	59636961.7	59290983.4	...	36900594.9	31272971.6	1234445.5	522.8226	1.02584	0.747

Mentre la correlazione lineare (calcolata secondo il metodo di Pearson) è in quest'altra tabella.

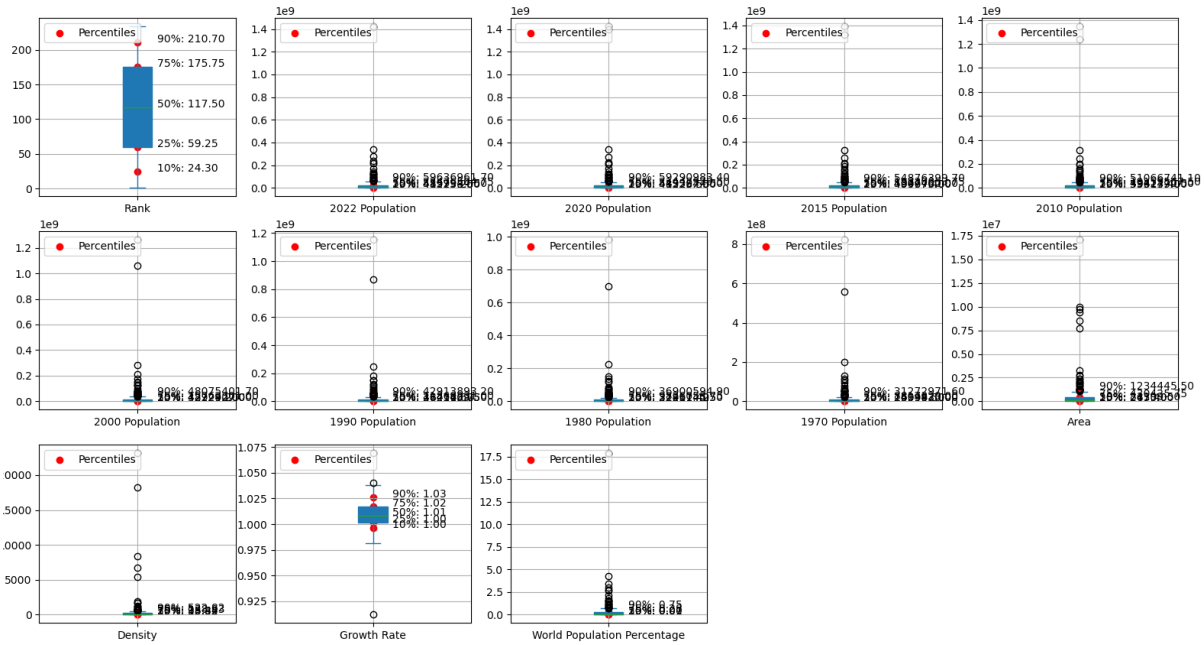
	Rank	CCA3	Country/Territory	Capital	2022 Population	2020 Population	2015 Population	...	1990 Population	1980 Population	1970 Population	Area	Density	Growth Rate	World Population Percentage
Rank	1.000000	-0.006452	-0.042460	0.145126	-0.358361	-0.355854	-0.351222	...	-0.336152	-0.335246	-0.335379	-0.383774	0.129436	-0.224561	-0.358464
CCA3	-0.006452	1.000000	0.836041	0.064403	-0.050542	-0.050842	-0.051307	...	-0.054373	-0.055566	-0.056375	-0.041418	0.013827	0.060015	-0.050665
Country/Territory	-0.042460	0.836041	1.000000	-0.027703	-0.044589	-0.044706	-0.044745	...	-0.045633	-0.044613	-0.043454	-0.051993	0.013591	0.045103	-0.044670
Capital	0.145126	0.064403	-0.027703	1.000000	-0.078016	-0.077126	-0.075373	...	-0.069952	-0.067360	-0.064576	-0.032602	-0.023288	-0.149169	-0.078050
2022 Population	-0.358361	-0.050542	-0.044589	-0.078016	1.000000	0.999946	0.999490	...	0.987228	0.980285	0.973162	0.453411	-0.027618	-0.020863	0.999999
2020 Population	-0.355854	-0.050842	-0.044706	-0.077126	0.999946	1.000000	0.999763	...	0.988724	0.982121	0.975254	0.454093	-0.027358	-0.025116	0.999944
2015 Population	-0.351222	-0.051307	-0.044745	-0.075373	0.999490	0.999763	1.000000	...	0.991594	0.985724	0.979414	0.458240	-0.026857	-0.032154	0.999487
2010 Population	-0.347461	-0.051752	-0.044699	-0.073601	0.998629	0.999105	0.999783	...	0.993929	0.988786	0.983042	0.461936	-0.026505	-0.037983	0.998626
2000 Population	-0.341057	-0.052812	-0.045543	-0.071453	0.994605	0.995583	0.997340	...	0.998336	0.995160	0.990956	0.473933	-0.026139	-0.050515	0.994598
1990 Population	-0.336152	-0.054373	-0.045633	-0.069952	0.987228	0.988724	0.991594	...	1.000000	0.999042	0.996602	0.486764	-0.026224	-0.062397	0.987218
1980 Population	-0.335246	-0.055566	-0.044613	-0.067360	0.980285	0.982121	0.985724	...	0.999042	1.000000	0.999194	0.498166	-0.026587	-0.072349	0.980273
1970 Population	-0.335379	-0.056375	-0.043454	-0.064576	0.973162	0.975254	0.979414	...	0.996602	0.999194	1.000000	0.509940	-0.026881	-0.081313	0.973150
Area	-0.383774	-0.041418	-0.051993	-0.032602	0.453411	0.454993	0.458240	...	0.486764	0.498166	0.509940	1.000000	-0.063128	-0.013970	0.453284
Density	0.129436	0.013827	0.013591	-0.023288	-0.027618	-0.027358	-0.026857	...	-0.026224	-0.026587	-0.026881	-0.063128	1.000000	-0.069753	-0.027646
Growth Rate	-0.224561	0.060015	0.045103	-0.149169	-0.020863	-0.025116	-0.032154	...	-0.062397	-0.072349	-0.081313	-0.013970	-0.069753	1.000000	-0.020930
World Population Percentage	-0.358464	-0.050665	-0.044670	-0.078050	0.999999	0.999944	0.999487	...	0.987218	0.980273	0.973150	0.453284	-0.027646	-0.020930	1.000000

Gli output della funzione *boxplot()*, risultano utili per visualizzare graficamente la distribuzione dei dati. Consento infatti di verificare in maniera rapida la normalità dei dati e la presenza degli outlier. La linea centrale viene usata per rappresentare la mediana, mentre la parte inferiore e superiore della scatola mostrano il 25esimo e 75esimo percentile. La posizione della mediana all'interno della scatola consente di verificare la simmetria dei dati: se è significativamente più vicina ad uno degli estremi della scatola, la distribuzione dell'attributo è asimmetrica. I baffi che partono dalla scatola si estendono per 1.5 volte la lunghezza della scatola, che rappresenta il range interquartile, ed i record che si trovano all'esterno dei baffi vengono detti outlier – in quanto sono al di fuori della variazione attesa. Inoltre nei grafici viene anche rappresentato il 90esimo percentile, identificato come un pallino rosso.

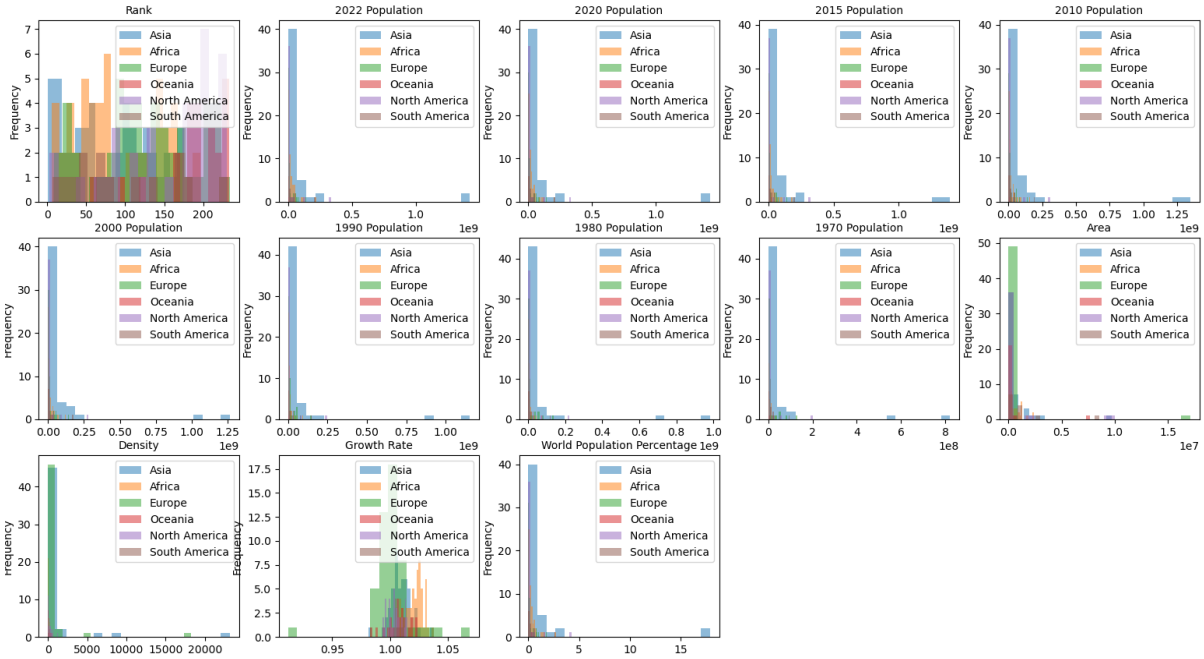
Questa prima figura mostra i box plot senza gli outlier:



Questa seconda figura mostra invece l'effetto degli outlier sugli attributi.



Nella figura seguente si vede l'output della funzione *histogram()*, che consente di ottenere una idea della distribuzione dei dati.



2.3 Discussione

Fra gli attributi si possono identificare alcuni gruppi con caratteristiche simili. Vi sono gli attributi categorici ("*CCA3*", "*Capital*", "*Country/Territory*") che insieme con Rank come si può vedere dalla mappa delle correlazioni, sono poco legati agli altri. Le eccezioni sono "*CCA3*" e "*Country/Territory*", che mostrano una forte correlazione tra di loro. Tutti gli attributi in argomento presentano un'alta varianza, dovuta al fatto che non sono presenti valori ripetuti.

Un secondo gruppo è quello composto dagli attributi legati alla popolazione nel corso degli anni. Questi mostrano una correlazione molto forte tra di loro e con la "*World Population Percentage*", ed una correlazione elevata, anche se meno forte rispetto alle precedenti, con l'attributo "*Area*". Risulta interessante notare come quest'ultima correlazione diminuisca negli anni, partendo da uno 0.51 con la popolazione degli anni 70 fino allo 0.45 con la popolazione del 2022. Analizzando i box plot senza outlier relativi agli attributi si nota anche una forte asimmetria nella distribuzione, con la mediana fortemente spostata verso il basso, quindi di tipo positivo. Dal secondo set di box plot si nota anche come gli outlier dominino tutte le misure non robuste alla loro presenza – due valori infatti svettano molto lontani da tutti gli altri, schiacciando i grafici verso il basso. Si nota inoltre come il minore dei due raggiunga l'altro con il tempo. Anche gli istogrammi mostrano questa forte asimmetria a destra e consentono di identificare i valori anomali come appartenenti alla classe "Asia". I valori alti di varianza e deviazione standard mostrano il risultato dell'estrema variazione dei valori all'interno delle classi – come anche il range che supera 1.4 miliardi. Anche i percentili calcolati mostrano questa tendenza, con quartili che nel 2020 vanno dal 415,284.5 al 25% a 21,447,979.5 al 75%

L'attributo "*Area*" si comporta in maniera simile al gruppo delle popolazioni, con una forte asimmetria positiva e un outlier che domina tutti gli altri. Al contrario della popolazione si può distinguere anche un secondo gruppo di valori molto più alti della norma, che formano un gruppo a metà strada tra l'estremo ed il resto dei valori. In questo caso l'istogramma mostra che il massimo appartiene alla classe "*Europe*", mentre il secondo gruppo è composto da record appartenenti ad altre 3 classi: "North America", "*South America*" ed "*Oceania*". Come per il gruppo delle popolazioni la varianza e la deviazione standard sono molto alte, ma analizzando i percentili si nota che gli outlier hanno una forte influenza su questi valori.

La feature "*Density*" si dimostra poco correlata con gli altri attributi, e mostra anch'essa una forte asimmetria positiva con la presenza di due gruppi di valori estremi che si distaccano nettamente dagli altri. In entrambi i gruppi i valori appartengono alle classi di "*Europe*" ed "*Asia*". I dati numerici mostrano una deviazione standard bassa rispetto a quella degli altri attributi, risultato dovuto al minore range dei valori – in particolare quelli che appartengono allo scarto interquartile. Questa minore variabilità si nota

anche nel box plot senza i valori estremi dove, al contrario degli altri attributi discussi in precedenza, il 90esimo percentile si trova sul baffo, seppure al limite.

"*Growth Rate*" mostra correlazione simile all'attributo precedente, ma si ha solo una leggera asimmetria positiva, con outlier che, pur avendo una forte influenza sui risultati di molti dei valori dell'analisi, non si mostrano capaci di dominarla del tutto come nel caso tutti gli attributi precedenti. Il box plot mostra infatti solo 3 valori estremi, ed in questo caso si trovano oltre entrambi i lati dei baffi – distribuzione che può diminuirne l'impatto su alcune misure di posizione. L'istogramma consente inoltre di vedere una distribuzione gaussiana dei valori, sebbene con una "campana" molto stretta. Una minore variabilità che si rispecchia anche in valori bassi sia per la deviazione standard che per la varianza, con uno scarto interquartile molto basso.

Infine, "*World Population Percentage*" si comporta in maniera simile alle popolazioni: fortemente correlato alle popolazioni, una buona correlazione con l'area, asimmetria fortemente positiva, con due outlier che dominano il resto dei valori – entrambi appartenenti alla classe "*Asia*". Si differenzia dalle popolazioni nel caso dei valori assoluti per la natura stessa dell'attributo, ma mostra comunque una alta deviazione standard considerando i valori di partenza ed un range interquartile con 0.01 al 25% e 0.28 al 75%.

In sintesi, si osserva che il dataset è sbilanciato, con una sottorappresentazione di "*North America*", "*South America*" ed "*Oceania*". Un gruppo di attributi è di scarso interesse in quanto composto da valori unici. Gli attributi legati alla popolazione nel corso degli anni ed il "*World Population Percentage*" presentano informazioni con correlazioni così alte da risultare poco significativi. Le caratteristiche degli attributi si riflettono anche sulle distribuzioni: quasi tutte non gaussiane, fortemente asimmetriche e con la presenza di outlier che non possono essere ignorati dallo studio. Si notano anche l'assenza del rumore e la scarsa correlazione lineare tra la maggior parte degli attributi.

3 Classificatori

3.1 Naive Bayes

Per affrontare il problema di classificazione dei paesi in continenti, abbiamo incluso, tra i vari modelli, un classificatore Naive Bayes, e nello specifico la variante ComplementNB di scikit-learn. Il Naive Bayes è un classificatore probabilistico basato sul teorema di Bayes, che permette di calcolare la probabilità a posteriori di una classe C, dati gli attributi (features) osservati $A_1, A_1, \ldots, A_n$:

$$P(C \mid A_1, A_2, \ldots, A_n) = \frac{P(A_1, A_2, \ldots, A_n \mid C) \cdot P(C)}{P(A_1, A_2, \ldots, A_n)} \tag{1}$$

dove:

- $P(C \mid A_1, A_2, \ldots, A_n)$ è la probabilità a posteriori della classe C dati gli attributi.
- $P(A_1, A_2, \ldots, A_n \mid C)$ è la verosimiglianza (likelihood) degli attributi data la classe.
- $P(C)$ è la probabilità a priori della classe.
- $P(A_1, A_2, \ldots, A_n)$ è la probabilità marginale degli attributi (un fattore di normalizzazione).

L'assunzione "naive" di indipendenza condizionata stabilisce che, data la classe, gli attributi sono indipendenti tra loro. Matematicamente, questo si esprime come:

$$P(A_1, A_2, \ldots, A_n \mid C) = \prod_{j=1}^{n} P(A_j \mid C) \tag{2}$$

Questa assunzione semplifica notevolmente il calcolo. Combinando il teorema di Bayes (1) con l'assunzione di indipendenza (2), otteniamo la formula per la probabilità a posteriori nel Naive Bayes:

$$P(C \mid A_1, A_2, \ldots, A_n) = \frac{\prod_{j=1}^{n} P(A_j \mid C) \cdot P(C)}{P(A_1, A_2, \ldots, A_n)} \tag{3}$$

Poiché il denominatore $P(A_1, A_2, \ldots, A_n)$ è costante per tutti i valori $C = c_i$ (dato un insieme di features), la classe predetta è quella che massimizza il numeratore:

$$\prod_{j=1}^{n} P(A_j \mid C = c_i) \cdot P(C = c_i) \tag{4}$$

Abbiamo scelto il Naive Bayes, e in particolare ComplementNB, per le seguenti ragioni:

1. **Baseline Semplice:** Volevamo un modello semplice e veloce da addestrare come **baseline** di performance. Questo ci permette di valutare se modelli più complessi apportano un miglioramento significativo.
2. **Robustezza a Features Irrilevanti:** Il Naive Bayes tende a essere meno sensibile a features irrilevanti rispetto ad altri modelli.
3. **ComplementNB per lo Sbilanciamento:** La variante ComplementNB è stata specificamente progettata per dataset sbilanciati, come il nostro, dove alcuni continenti hanno molti meno paesi rappresentati rispetto ad altri.
4. **Approccio Probabilistico:** Il Naive Bayes fornisce una **probabilità** di appartenenza a ciascuna classe, non solo una predizione secca. Questo può essere utile per valutare l'**incertezza** del modello nelle sue predizioni.

Siamo consapevoli che l'assunzione di indipendenza condizionata è molto forte e probabilmente non valida nel nostro caso. Ad esempio, è ragionevole aspettarsi che la popolazione di un paese in un determinato anno sia fortemente correlata alla popolazione negli anni precedenti, e che l'area di un paese sia correlata alla sua popolazione. Tuttavia, abbiamo deciso di utilizzare comunque il Naive Bayes come punto di partenza, data la sua semplicità e le sue proprietà, pur con la consapevolezza dei suoi limiti.

3.1.1 Risultati sul dataset grezzo

Tuning Iperparametri Il classificatore ComplementNB utilizzato in questo studio ha due iperparametri principali che possono influenzare significativamente le performance:

- **alpha**: È il parametro di smoothing di Laplace (additive smoothing) che previene valori di probabilità zero quando alcune caratteristiche non appaiono nei dati di training per una particolare classe. Un valore più alto di alpha implica uno smoothing maggiore, che può ridurre l'overfitting ma potenzialmente diminuire la precisione di stima.
- **norm**: Parametro booleano che, quando impostato a True, applica una normalizzazione ai vettori delle features, aiutando a compensare le differenze nella lunghezza dei documenti (nel nostro caso, la magnitudine delle caratteristiche demografiche).

Per identificare la combinazione ottimale di questi parametri, è stata implementata una ricerca a griglia (GridSearch) con validazione incrociata a 5 fold, utilizzando i seguenti valori:

- alpha: [0.1, 0.5, 1.0, 2.0, 5.0]
- norm: [True, False]

I risultati del tuning hanno indicato che la combinazione ottimale per il dataset grezzo è: **alpha** = 0.1, **norm** = False.
Il valore ottimale di alpha (0.1) è significativamente inferiore al valore predefinito (1.0), suggerendo che il modello beneficia di un minor smoothing delle probabilità. Questo indica che le caratteristiche demografiche contengono pattern discriminativi significativi che non dovrebbero essere eccessivamente attenuati.
La preferenza per norm=False è particolarmente interessante: per questo specifico dataset demografico, la normalizzazione delle feature può avere un effetto negativo sulle performance del classificatore Naive Bayes (confermato anche successivamente durante i confronti con le tecniche di pre-processing), probabilmente perché la scala originale delle variabili (ad esempio, l'ordine di grandezza della popolazione) contiene informazione discriminativa che viene persa durante la normalizzazione.

Lavorando quindi sul dataset grezzo, rimuovendo soltanto le feature non necessarie ("*CCA3*", "*Country/Territory*", "*Capital*") e applicando il tuning, il classificatore Naive Bayes (ComplementNB) ha raggiunto un'**accuratezza** del 40.43% sul test set, confermata da una cross-validation con media del 47.03%. Questi risultati, evidenziano alcune **limitazioni** significative.
L'analisi delle metriche per classe rileva un comportamento fortemente sbilanciato del classificatore:

```
Accuratezza sul test set: 0.4043
Accuratezza media CV: 0.4703
               precision    recall  f1-score   support

       Africa       0.35      0.73      0.47        11
         Asia       0.22      0.20      0.21        10
       Europe       0.60      0.90      0.72        10
North America       0.00      0.00      0.00         8
      Oceania       0.00      0.00      0.00         5
South America       0.00      0.00      0.00         3

     accuracy                           0.40        47
    macro avg       0.20      0.30      0.23        47
 weighted avg       0.26      0.40      0.31        47
```

- **Europa e Africa**: Il modello mostra buone capacità predittive per queste classi, con recall rispettivamente del 90% e del 73%. Questo suggerisce che le caratteristiche demografiche di questi continenti sono sufficientemente distintive anche senza feature engineering avanzato.
- **Asia**: Con un recall di solo il 20%, il modello fatica molto a riconoscere correttamente i paesi asiatici, probabilmente a causa della grande eterogeneità demografica presente nel continente.
- **Nord America, Oceania e Sud America**: Il modello si dimostra completamente incapace di riconoscere paesi appartenenti a queste classi (recall 0%). Questo rappresenta la limitazione più grave, suggerendo che il dataset grezzo non contiene informazioni a sufficienza per distinguere questi continenti dagli altri.

Limitazioni del modello La distribuzione sbilanciata delle classi nel dataset (con Africa e Asia sovrarappresentate) probabilmente contribuisce alla tendenza del modello a favorire queste classi nelle sue previsioni. Inoltre, l'assenza di normalizzazione dei dati potrebbe penalizzare il classificatore Bayesiano, particolarmente sensibile alla scala delle variabili.
La bassa precisione complessiva (macro-average 0.20) indica che il modello spesso assegna erroneamente le classi, mentre il moderato F1-score ponderato (0.31) riflette il compromesso tra precisione e recall. Questi risultati forniscono una **baseline** importante per valutare l'impatto delle tecniche di miglioramento che saranno applicate successivamente: feature selection, normalizzazione dei dati e bilanciamento delle classi.

3.2 Support Vector Machine

Il Support Vector Machine (SVM) è un algoritmo di classificazione supervisionato che cerca di individuare un iperpiano ottimale per separare le classi massimizzando il margine tra i punti appartenenti a categorie diverse. Questo modello è particolarmente efficace per problemi con dati complessi e spazi di alta dimensionalità. Nel nostro caso, l'obiettivo è classificare gli stati nei vari continenti in base alle loro caratteristiche numeriche. SVM è una buona scelta per questo compito per diversi motivi:

- **Gestione di dati multidimensionali**: Il modello può lavorare con molte feature senza soffrire troppo di overfitting.
- **Separabilità non lineare**: Grazie all'uso di kernel come RBF, il modello può individuare confini di decisione complessi.
- **Buona generalizzazione**: SVM funziona bene su dataset con un numero moderato di esempi rispetto alle feature disponibili.

Tuttavia, ci sono alcune sfide:

- La distribuzione delle classi è altamente sbilanciata.
- Non tutte le feature sono rilevanti per la classificazione.
- La scelta degli iperparametri può avere un forte impatto sulle prestazioni del modello.

Per ottenere le migliori prestazioni, abbiamo applicato diverse tecniche di preprocessing e abbiamo ottimizzato gli iperparametri con GridSearch.

3.2.1 Risultati sul dataset grezzo

Tuning iperparametri Per trovare la configurazione ottimale del modello SVM, abbiamo eseguito una ricerca degli iperparametri con **GridSearchCV**, testando diverse combinazioni:

- **Kernel**: Sono stati valutati linear, rbf, poly e sigmoid, con **RBF** che si è rivelato il più performante.
- **Parametro C:** Regola il compromesso tra margine e accuratezza. Abbiamo testato i valori [0.01, 0.1, 1, 10, 100], con il miglior risultato ottenuto per **C = 100.**
- **Gamma**: Controlla l'influenza dei singoli punti nei kernel non lineari. Tra i valori testati [1, 0.1, 0.01, 0.001], il migliore è risultato **1**.

Nel nostro progetto, abbiamo riscontrato che senza la standardizzazione nel preprocessing, il tuning degli iperparametri di SVM si blocca. Questo problema si verifica perché, durante l'ottimizzazione di Gamma e C, il modello può avere difficoltà a convergere, specialmente con kernel RBF o Polinomiale. Per risolvere questa criticità, abbiamo implementato la funzione TuningIperparametri(), che utilizza dati standardizzati per trovare i migliori valori di Kernel, Gamma e C. Una volta individuati, questi parametri sono stati direttamente inseriti nel codice. Questo approccio evita interruzioni durante l'addestramento e permette di valutare l'accuratezza dell'SVM anche con i dati grezzi senza problemi di convergenza.

Con l'utilizzo dei migliori iperparametri trovati, abbiamo ottenuto le seguenti prestazioni sul dataset grezzo:

```
Accuratezza sul test set: 0.2340
Score della cross-validation: [0.23684211 0.23684211 0.24324324 0.27027027 0.24324324]
Accuratezza media CV: 0.2461
Accuratezza del modello: 0.25
```

I risultati ottenuti sono pessimi, sia sul test set che con il cross validation, ma crediamo che il preprocessing possa portare a dei risultati decisamente migliori.

3.3 Decision Tree

Gli alberi decisionali sono una delle tecniche di classificazione maggiormente utilizzate. Hanno una **struttura gerarchica** costituita da un insieme di nodi, correlati da archi orientati e etichettati, che permette di rappresentare un insieme di **regole di classificazione**. Perciò per classificare un nuovo record in questi tipo di classificatori è necessario semplicemente seguire la regola di classificazione giusta e quindi il giusto percorso partendo dalla radice fino ad arrivare ad una foglia.
Abbiamo scelto questo tipo di classificatori perché:

- Sono classificatori **molto semplici da capire e spiegare** inoltre la loro struttura permette chiaramente di mostrare come si sviluppa il processo decisionale.
- Non richiede normalizzazione o standardizzazione dei dati poiché le divisioni sono basate su soglie specifiche delle caratteristiche.
- Possono catturare relazioni non lineari tra gli attributi e la classe dei record.

Ovviamente sempre consapevoli dei contro:

- Decision boundary di limitata espressività.
- Rischio di overfitting a causa di un troppo adattamento ai dati di training.
- Rischio di frammentazione dei dati.

Sarà quindi necessario determinare i migliori iperparametri proprio per evitare i problemi appena citati.

Realizzazione e test del classificatore Per la sua realizzazione e ottimizzazione si sono utilizzate le funzioni offerte da scikit-learn mentre per quanto riguarda la scelta degli iperparametri si è deciso di ottimizzare la **massima profondità** dell'albero e il **massimo numero di nodi foglia**. Questi ultimi se ottimizzati evitano che l'albero si adattarsi troppo ai dati o di raggiunga una complessità eccessiva che ridurebbe la sua capacità di generalizzazione. Per quanto riguarda la misura della purezza dei nodi, si è stabilito a priori che sarò calcolata con l'**indice di GINI**.
In pratica quindi una volta creato l'albero, per il tuning si è scelto di utilizzare la funzione di scikit_learn 'GridSearch' la quale permette di ottenere i migliori iperparametri a seguito di una procedura di ricerca esaustiva e molto efficace la quale utilizza, per ridurre eventuali rischi di overfitting, una tecnica di cross-validation per la verifica delle performance.
Infine, abbiamo addestrato un nuovo modello con i migliori iperparametri individuati ed effettuato le predizioni sul test set calcolandone l'accuratezza. Un ulteriore verifica delle performance è stata poi effettuata sul training set attraverso una cross validation a 5 fold.
Così facendo otteniamo sia il risultato effettivo sul test set, utile per capire quanto è efficace il modello su dati nuovi, e sia una verifica sul training set che ci permette anche di capire se l'errore non sia l'overfitting ma magari il modello in sé che non riesce a adattarsi ai dati.

3.3.1 Risultati sul dataset grezzo

```
Migliori parametri trovati: {'max_depth': 16, 'max_leaf_nodes': 24}
Miglior score in CV: 0.4657
Accuratezza sul test set: 0.4681
Score della cross-validation: [0.52631579 0.31578947 0.43243243 0.51351351 0.48648649]
Accuratezza media CV: 0.4549
Accuratezza del modello: 0.45
```

Dai risultati quindi gli iperparametri migliori sembrano essere una profondità massima di 16 e un numero massimo di nodi foglia di 24.
Il risultato sul test set è abbastanza pessimo e ciò, ricordando che si sta utilizzando il dataset grezzo senza alcun preprocessing, può essere dato da diversi fattori tra cui:

1. Il dataset essendo sbilanciato favorisce le classi con maggior numero di record, il che vuoldire che l'influenza delle classi maggioritarie porta le regole di classificazione a pendere a loro favore a discapito delle classi con meno record con un conseguente aumento nella difficoltà di classificare adeguatamente proprio i record di quest'ultime.

2. Il dataset avendo attributi estremamente correlati tra loro può portare l'albero a crescere in complessità o nel numero di nodi senza però un effettivo guadagno nelle performance.

L'unica nota positiva sembra essere la resistenza degli alberi decisionali agli attributi con scale diverse (nel nostro dataset è presente una grossa differenza di scale di valori tra gli attributi).
Dai risultati si nota poi che il risultato ottenuto nel training set è molto simile a quello ottenuto sul test set, si ha quindi un **underfitting** che in parte può essere dato dai problemi del dataset ma in parte probabilmente è dato dal fatto che il modello non riesce a adattarsi ai dati e, di conseguenza, non riesce dividere adeguatamente i record nelle loro classi.
Una possibilità è che l'albero decisionale non sia adatto a questo tipo di problema; noi però crediamo che la problematica sia, più che il classificatore, il fatto che gli attributi non trasmettano abbastanza informazioni ai fini della predizione.
Tutto ciò, come fanno notare anche gli iperparametri migliori individuati, porta l'albero a cercare di compensare questo problema crescendo in numero di nodi e soprattutto in profondità (16 è una profondità esagerata per un problema di questo tipo) portando infine a un maggiore overfitting. Vedremo se con il preprocessing sarà possibile migliorare le performance di questo classificatore.

3.4 KNN Custom

Come classificatore semplice custom abbiamo deciso di utilizzare il classificatore KNearestNeighbour. Il **KNN** fa parte dei classificatori **Instance-based** (noti anche come Lazy Learners) i quali sfruttano un apprendimento basato sulla memorizzazione dei dati di addestramento forniti, utilizzandoli come riferimento e confronto per la classificazione di nuove istanze.
Questo confronto utilizza una particolare **misura di prossimità**: usano la distanza o similarità tra i vari esempi.

Per creare e utilizzare un modello come il KNN per prima cosa è necessario memorizzare al suo interno l'intero training set (questa in pratica è la sua fase di addestramento) dopodiché per ciascun nuovo record da classificare viene calcolata la sua **distanza** da ciascun oggetto del training set e infine si individuano i k record più vicini: al nuovo record è assegnata la classe prevalente tra questi ultimi.

Abbiamo deciso di utilizzare questo classificatore perché:

- Non richiede l'induzione di un modello, con conseguente risparmio di risorse e **semplicità**.
- Non fa ipotesi sui dati (visto che si basa solo sulla misura di prossimità), di conseguenza è possibile utilizzarlo in un'ampia gamma di problemi, anche quando il dataset non è divisibile in maniera lineare(quindi è molto **flessibile**)
- Su dataset piccoli può comunque mantenere una certa accuratezza, che altri algoritmi invece fanno fatica a mantenere.

Ovviamente teniamo in considerazione anche i suoi lati negativi ovvero:

- Ogni volta occorre calcolare la prossimità tra il nuovo record e gli esempi del training set e ciò comporta che richieda più risorse computazionali rispetto ad altri modelli.
- Richiede molta memoria per memorizzare tutti i dati all'interno.
- Potrebbe ignorare o considerare meno attributi con una scala di valori inferiore di conseguenza è necessario quindi una normalizzazione o standardizzazione

Il punto cruciale quando si vuole utilizzare un classificatore di questo tipo è quindi effettuare un adeguata normalizzazione degli attributi e individuare il valore di k e la misura di prossimità più adatti.

Realizzazione e test del classificatore Nella pratica tutto ciò è stato implementato in codice python senza l'utilizzo di alcuna funzione esterna e per gli iperparametri è stato scelto di ottimizzare il numero di elementi più vicini per determare la classe 'k', il tipo di distanza da utilizzare durante il calcolo delle distanze e infine se pesare o no le distanze (quindi il tipo di peso da utilizzare).
Abbiamo creato la classe '**knnCustom**' che accetta in ingresso il training set, gli iperparametri del classificatore e il numero di fold con cui fare una cross validation. Dopodichè all'interno abbiamo:

- La funzione '**calcolo_distanza**' che accetta il training set, il test set e la distanza con cui fare il calcolo; restituisce una lista di liste con all'interno il valore della distanza tra ogni record del training set e quello del test set corrispondente.
- La funzione '**knn**' che accetta il training set, i record del test set e gli iperparametri scelti; restituisce una lista con la predizione della classe per ogni record del test set.
- La funzione '**calcolo_accuratezza**' che accetta le predizioni del classificatore e le effettive classi dei record del test set; restituisce l'accuratezza del modello.
- La funzione '**tuning_con_cross_validation**' non riceve niente in ingresso e in pratica restituisce la media dell'accuratezza ottenuta durante un cross validation effettuata sul training set con i parametri ottimali individuati.

Quindi in pratica dopo aver creato il classificatore abbiamo effettuato il calcolo dell'accuratezza media, sul training set, ottenuta da un cross validation che utilizza i migliori iperparametri (con l'apposita funzione citata) dopodichè abbiamo creato un nuovo classificatore utilizzando proprio questi ultimi e effettuato le predizioni sul test set calcolandone l'accuratezza.

3.4.1 Risultati sul dataset grezzo

```
Accuratezza Media sui Fold con i Migliori Parametri: 0.5785169785169786
Accuratezza sul test set: 0.3191489361702128
```

A quanto pare, i risultati sul test set risultano pessimi mentre quelli sul training set, nonostante non siano comunque ottimali, sembrano essere decisamente migliori. Il modello, quindi, non riesce a adattarsi completamente ai dati e decisamente fa fatica a predire le classi di nuovi record.
I migliori iperparametri durante le predizioni sono **k=1**, distanza utilizzata = **distanza euclidea** e peso utilizzato = **peso uniforme**. Quindi questo classificatore sembra performare meglio quando valuta un solo elemento vicino. Questo però, come si può anche osservare poi nell'accuratezza sul test set, porta a **overfitting** perché il classificatore si è adattato troppo ai dati e non ha di conseguenza una buona capacità di generalizzazione.
Ricordiamo poi che il modello è addestrato su **dati non standardizzati** è ciò può aver diminuito considerevolmente le capacità predittive di questo tipo di classificatori. Tra gli attributi, infatti, c'è una grande differenza di scala di valori ed essendo il KNN un classificatore che si basa sulle distanze tra i punti dati ciò comporta che tali attributi dominino, in modo eccessivo, il calcolo delle distanze con conseguente diminuzione di importanza (nella fase predittiva) di attributi con scale inferiori. Quindi se questi ultimi attributi sono effettivamente efficaci nel discriminare i record, ora come ora non siamo in grado di sfruttare queste informazioni.
Un altro problema evidente è lo **sbilanciamento del dataset** che comporta appunto classi con pochi record. Il nostro classificatore potrebbe aver assegnato il record alla classe errata data proprio questa disparità di elementi. Comunque, avendo un k molto piccolo, dovremmo aver perlomeno diminuito tale possibilità. Sarà comunque necessario quindi un successivo bilanciamento del dataset.
Per quanto riguarda il numero di attributi, di sicuro con 12 attributi la dimensionality curse incide sulla performance, e oltre a quello c'è da contare che che molti di tali attributi sono estremamente correlati tra loro il che comporta che insieme abbiano un peso maggiore poi nella effettiva predizione (rispetto agli attributi non correlati).

Si suppone infine che le caratteristiche utilizzate potrebbero non essere sufficientemente distintive e quindi il classificatore non riesca a dividere i record adeguatamente (problematica che potrebbe comunque rimanere dopo tutte le fasi di preprocessing). Vedremo quindi se con il preprocessing tutte queste problematiche potranno essere risolte o perlomeno gestite così da ottenere una performance migliore

3.5 Classificatore multiplo custom

Per creare il classificatore multiplo custom si è creata la classe *Ensemble_classifier* per definire il tipo di oggetto. La classe contiene i metodi *__init__()*, *fit()*, *base_pred()* e *hard_voting()*.

Il metodo *__init__()* viene utilizzato per creare i classificatori di base che verranno usati dall'oggetto. In quanto metodo costruttore, viene eseguito in maniera automatica quando viene definito un oggetto appartenente a questa classe. I modelli scelti sono: il **Decision Tree Classifier**, il **Support Vector Classifier**, ed il **Naive Bayes Classifier**. Il primo è stato scelto in quanto si tratta di un modello rapido e poco costoso, capace di gestire dati sia numerici che categorici. Inoltre, su un dataset semplice, come quello preso in analisi, ha delle prestazioni paragonabili a metodi più complessi. Lo SVC è stato scelto a causa della sua flessibilità nel gestire dataset con outlier sfruttando le variabili di slack per ottenere un soft margin che sia in grado di tollerarli. Infine, il Naive Bayes è poco influenzato sia dagli outlier che dagli attributi poco significativi per l'analisi, il che lo rende una buona opzione per il dataset. Con questi 3 modelli si crede che il classificatore multiplo riesca ad avere una precisione più alta dei singoli classificatori.
Il metodo *fit()* è usato per indurre i classificatori sul training set che riceve in ingresso, *base_pred()* invece è usato per ottenere le predizioni dei singoli classificatori. *hard_voting()*, infine, effettua la previsione finale del classificatore multiplo.

3.5.1 Risultati sul dataset grezzo

Nella tabella seguente sono riportate le prestazioni dei classificatori singoli e del multiplo.

```
Decision Tree Accuracy:  0.46808510638297873
Decision Tree Precision:  0.40946969696969693
Decision Tree Recall:  0.46414141414141413
Decision Tree F1:  0.4113871635610766

Support Vector Accuracy:  0.23404255319148937
Support Vector Precision:  0.03900709219858156
Support Vector Recall:  0.16666666666666666
Support Vector F1:  0.0632183908045977

Naive Bayes Accuracy:  0.1702127659574468
Naive Bayes Precision:  0.14168391994478952
Naive Bayes Recall:  0.20113636363636367
Naive Bayes F1:  0.12722832722832722

Hard Voting Accuracy:  0.425531914893617
Hard Voting Precision:  0.3718045112781954
Hard Voting Recall:  0.353030303030303
Hard Voting F1:  0.346813725490196
```

La precisione ottenuta è maggiore rispetto a quella dei modelli capaci di gestire modelli più complessi, ma non raggiunge quella dell'albero decisionale. Sui dati grezzi, i risultati non giustificano l'utilizzo del classificatore multiplo. Si osserverà se il pre-processing sia capace di migliorarne le prestazioni.

4 Tecniche di Pre-processing utilizzate

4.1 Bilanciamento

Poiché il dataset si è rivelato sbilanciato, occorre utilizzare delle tecniche di bilanciamento per ovviare al problema. Si nota infatti che le classi "South America" e "Oceania" sono sottorappresentate rispetto alle altre 4 – i loro dati risultano quindi più importanti per il corretto addestramento degli algoritmi e non conviene ridurli.

Le tecniche utilizzate sono 3: lo SMOTE oversampling, il random oversampling ed il random undersampling. Fra le tecniche di oversampling la prima risulta più precisa, e viene paragonata a quella casuale per verificare la sua capacità di creare una decision boundary più robusta. Poiché il numero di dati appartenenti alle classi minoritarie è ridotto si è preferito concentrarsi sulle tecniche di oversampling, quindi l'unica tecnica di undersampling valutata è il random undersampling.

4.1.1 Ensemble classifier

```
Decision Tree Accuracy:  0.46808510638297873
Decision Tree Precision:  0.40946969696969693
Decision Tree Recall:  0.4641414141414413
Decision Tree F1:  0.4113871635610766

Support Vector Accuracy:  0.23404255319148937
Support Vector Precision:  0.03900709219858156
Support Vector Recall:  0.16666666666666666
Support Vector F1:  0.0632183908045977

Naive Bayes Accuracy:  0.1702127659574468
Naive Bayes Precision:  0.14168391994478952
Naive Bayes Recall:  0.20113636363636367
Naive Bayes F1:  0.12722832722832722

Hard Voting Accuracy:  0.425531914893617
Hard Voting Precision:  0.3718045112781954
Hard Voting Recall:  0.353030303030303
Hard Voting F1:  0.346813725490196
```

Ensemble con dati grezzi

SMOTE Si può notare un impatto non trascurabile sulle prestazioni del modello, che subiscono una influenza negativa. In particolare, viene colpito il Support Vector Classifier che crolla dallo 0.234 di precisione fino allo 0.064. Questo, insieme ad una minore ma comunque significativa riduzione nella accuratezza del Decision Tree porta a risultati peggiori anche il classificatore multiplo.
I dati ottenuti indicano quindi che la tecnica SMOTE non sia adatta al classificatore, e che il tentativo di generare più record su classi minoritarie porti ad una maggiore varianza. Questa peggiora le predizioni del modello, ed in realtà riesce solo a complicare il dataset.

```
Decision Tree Accuracy:  0.40425531914893614
Decision Tree Precision:  0.3828282828282828
Decision Tree Recall:  0.4161616161616162
Decision Tree F1:  0.3480080921257392

Support Vector Accuracy:  0.06382978723404255
Support Vector Precision:  0.010638297872340425
Support Vector Recall:  0.16666666666666666
Support Vector F1:  0.02

Naive Bayes Accuracy:  0.1702127659574468
Naive Bayes Precision:  0.12779503105590062
Naive Bayes Recall:  0.20113636363636367
Naive Bayes F1:  0.1253968253968254

Hard Voting Accuracy:  0.3829787234042553
Hard Voting Precision:  0.3731060606060606
Hard Voting Recall:  0.3994949494949495
Hard Voting F1:  0.3349248790425261
```

Ensemble con SMOTE oversampling

Random oversampling Le affermazioni fatte per quanto riguarda la tecnica SMOTE vengono confermate dall'uso del metodo random, che mostra comunque un deterioramento delle capacità di classificazione maggiore rispetto a quella precedente, ma con una minore perdita di prestazioni per i classificatori base.

```
Decision Tree Accuracy:  0.425531914893617
Decision Tree Precision:  0.2770600744284955
Decision Tree Recall:  0.42676767676767685
Decision Tree F1:  0.32759781725298964

Support Vector Accuracy:  0.23404255319148937
Support Vector Precision:  0.03900709219858156
Support Vector Recall:  0.16666666666666666
Support Vector F1:  0.063218390804597

Naive Bayes Accuracy:  0.1702127659574468
Naive Bayes Precision:  0.12852254428341384
Naive Bayes Recall:  0.20113636363636367
Naive Bayes F1:  0.125952380952381

Hard Voting Accuracy:  0.3617021276595745
Hard Voting Precision:  0.20555555555555557
Hard Voting Recall:  0.3378787878787879
Hard Voting F1:  0.25432098765432 1
```

Ensemble con random oversampling

Random undersampling Questa tecnica porta anch'essa ad un peggioramento delle prestazioni, riducendo significativamente l'accuratezza sia del Decitsion Tree che del Support Vector Classifier. In compenso il Naive Bayes guadagna molto da questo approccio, consentendo al metodo di ottenere risultati paragonabili a quelli del random oversampling – anche se con precisione e recall migliori.
La causa di questo deterioramento è probabilmente la riduzione dei dati disponibili per l'addestramento dei classificatori, che si rivela troppo significativa su un dataset di dimensioni ridotte come quello analizzato.

```
Decision Tree Accuracy:  0.3829787234042553
Decision Tree Precision:  0.4009259259259259
Decision Tree Recall:  0.3911616161616162
Decision Tree F1:  0.36905458089668614

Support Vector Accuracy:  0.06382978723404255
Support Vector Precision:  0.010638297872340425
Support Vector Recall:  0.16666666666666666
Support Vector F1:  0.02

Naive Bayes Accuracy:  0.2978723404255319
Naive Bayes Precision:  0.32474747474747473
Naive Bayes Recall:  0.3037878787878788
Naive Bayes F1:  0.25921686338353006

Hard Voting Accuracy:  0.3617021276595745
Hard Voting Precision:  0.3860449735449736
Hard Voting Recall:  0.3744949494949495
Hard Voting F1:  0.34890468148982073
```

Ensemble con random undersampling

4.1.2 Decision Tree

```
Miglior score in CV: 0.4552
Accuratezza sul test set: 0.4894
Score della cross-validation: [0.42105263 0.31578947 0.43243243 0.48648649 0.48648649]
Accuratezza media CV: 0.4284
Accuratezza del modello: 0.43
```

Decision tree con dati grezzi

SMOTE Gli effetti della tecnica di oversampling sul classificatore risultano particolarmente interessanti, in quanto sembrano indicare un caso di overfitting. Nel corso delle cross validation il classificatore addestrato sulle versioni bilanciate mostra delle performance notevolmente migliori, fatto che influenza poi la sua accuratezza media – notevolmente superiore a quella del classificatore che lavora sui dati grezzi. Un'analisi dei risultati ottenuti sul test set mostra invece dei risultati opposti: un crollo di circa 12 punti percentuali nelle prestazioni su quest'ultimo.
Questi dati indicano che la tecnica di SMOTE introduce troppa variabilità nei dati sintetici, peggiorando sensibilmente le capacità del classificatore – si rivela dunque inadatta

```
Miglior score in CV: 0.5871
Accuratezza sul test set: 0.3617
Score della cross-validation: [0.51785714 0.50909091 0.54545455 0.67272727 0.65454545]
Accuratezza media CV: 0.5799
Accuratezza del modello: 0.58
```

Decision tree con SMOTE

Random oversampling I risultati ottenuti con il metodo random confermano il problema del precedente. I risultati ottenuti sui set di validazione sono molto migliori rispetto a quelli con SMOTE, e anche quelli sul test set si mostrano superiori. Questi ultimi rimangono tuttavia peggiori di quelli sui dati grezzi.
I risultati sembrano confermare che per gestire un dataset così sbilanciato le tecniche di oversampling siano inadatte.

```
Miglior score in CV: 0.6451
Accuratezza sul test set: 0.4255
Score della cross-validation: [0.60714286 0.65454545 0.63636364 0.67272727 0.63636364]
Accuratezza media CV: 0.6414
Accuratezza del modello: 0.64
```

Decision tree con Random oversampling

Random undersampling Il metodo di undersampling porta anch'esso ad una accuratezza notevolmente peggiore sul test set, ma in questo caso non migliora nemmeno quelle sul set di validazione. Risulta un metodo inadatto, in parte a causa della eccessiva riduzione dei valori per le classi maggioritarie necessaria per bilanciarle.

```
Miglior score in CV: 0.4253
Accuratezza sul test set: 0.3830
Score della cross-validation: [0.35714286 0.46153846 0.46153846 0.46153846 0.30769231]
Accuratezza media CV: 0.4099
Accuratezza del modello: 0.41
```

Decision tree con random undersampling

4.1.3 Naive Bayes

```
Miglior score in CV: 0.4703
Accuratezza sul test set: 0.4043
Score della cross-validation: [0.5        0.5        0.37837838 0.54054054 0.43243243]
Accuratezza media CV: 0.4703
Accuratezza del modello: 0.47
```

Naive bayes con dati grezzi

SMOTE Al contrario dell'albero decisionale con SMOTE si ottengono risultati peggiori nei 5 set di validazione generati dalla cross-validation, anche se il peggioramento è moderato rispetto ad altri analizzati in precedenza. Nel caso del test set si ha invece un leggero miglioramento, dovuto alla robustezza dei classificatori bayesiani rispetto al rumore generato dal metodo di generazione di dati sintetici. Questo risultato porta a pensare che i metodi di bilanciamento possano avere degli effetti positivi su questo classificatore.

```
Miglior score in CV: 0.4170
Accuratezza sul test set: 0.4255
Score della cross-validation: [0.32142857 0.43636364 0.41818182 0.47272727 0.43636364]
Accuratezza media CV: 0.4170
Accuratezza del modello: 0.42
```

Naive Bayes con SMOTE

Random oversampling I risultati ottenuti con il random oversampling confermano l'affermazione del paragrafo precedente. Gli effetti sono infatti gli stessi, ma il metodo porta a risultati più estremi – con un guadagno significativo di accuratezza nel test set ed una perdita altrettanto importante in quelli di validazione.
Si dimostra il metodo ottimale per questo classificatore, ma la variabilità dei risultati ottenibili dovuta alla sua natura casuale lo rende poco affidabile come metodo di miglioramento del modello.

```
Miglior score in CV: 0.3769
Accuratezza sul test set: 0.4681
Score della cross-validation: [0.33928571 0.4        0.30909091 0.4        0.43636364]
Accuratezza media CV: 0.3769
Accuratezza del modello: 0.38
```

Naive Bayes con random oversampling

Random undersampling Il random undersampling porta, con questo classificatore, ad ottenere dei risultati intermedi rispetto a quelli delle due tecniche viste in precedenza. Si ha infatti un miglioramento delle prestazioni nel set di test, accompagnato da un peggioramento di quelle nei set di validazione. In entrambi i casi, per quanto la differenza con i risultati ottenuti sul dataset grezzo non sia trascurabile, non raggiunge nemmeno gli estremi del metodo precedente.
I risultati ottenuti indicano che le tecniche di bilanciamento sono adatte al classificatore bayesiano, e che nonostante un apparente peggioramento nella fase di validazione, ci si possa aspettare un moderato incremento dell'accuratezza nella classificazione.

```
Miglior score in CV: 0.4099
Accuratezza sul test set: 0.4468
Score della cross-validation: [0.35714286 0.38461538 0.53846154 0.38461538 0.38461538]
Accuratezza media CV: 0.4099
Accuratezza del modello: 0.41
```

Naive Bayes con random undersampling

4.1.4 Support Vector Classifier

```
Accuratezza sul test set: 0.2340
Score della cross-validation: [0.23684211 0.23684211 0.24324324 0.27027027 0.24324324]
Accuratezza media CV: 0.2461
Accuratezza del modello: 0.25
```

SVC con dati grezzi

SMOTE La tecnica SMOTE si rivela del tutto inadatta per l'uso con lo SVC. Si ottiene infatti un crollo dell'accuratezza sul test set di oltre 15 punti percentuali, e le sue prestazioni sui set di validazione – per quanto migliori – risultano comunque notevolmente peggiorate rispetto a quelle sui dati grezzi.
Il classificatore si dimostra particolarmente vulnerabile alla variabilità introdotta nel dataset dai dati sintetici, il che indica che tecniche che ne fanno uso siano inadatte.

```
Accuratezza sul test set: 0.0638
Score della cross-validation: [0.16071429 0.16363636 0.16363636 0.16363636 0.16363636]
Accuratezza media CV: 0.1631
Accuratezza del modello: 0.16
```

SVC con SMOTE

Random oversampling I risultati ottenuti applicando questa tecnica risultano interessanti, in quanto non si ha un vero guadagno in termini di capacità di classificazione sui dati del set di prova. Tuttavia, è possibile notare un marcato miglioramento su quelli dei set di validazione.
In questo caso, i risultati ottenuti non sembrano indicare un problema di overfitting, quanto una incapacità del metodo di bilanciamento di influenzare in maniera positiva i risultati – rivelandosi così inefficace.

```
Accuratezza sul test set: 0.2340
Score della cross-validation: [0.55357143 0.50909091 0.65454545 0.67272727 0.76363636]
Accuratezza media CV: 0.6307
Accuratezza del modello: 0.63
```

SVC con random oversampling

Random undersampling Poiché il classificatore si basa sulla massimizzazione del margine tra le classi, ridurre troppo i dati che utilizza per calcolare il confine tra di esse gli risulta fatale. Si ottengono infatti risultati paragonabili a quelli ottenuti con il metodo SMOTE. Si dimostra quindi anch'esso inadatto al classificatore preso in esame.

```
Accuratezza sul test set: 0.0638
Score della cross-validation: [0.14285714 0.15384615 0.15384615 0.15384615 0.15384615]
Accuratezza media CV: 0.1516
Accuratezza del modello: 0.15
```

SVC con random undersampling

4.1.5 KNN Custom

```
Accuratezza Media sui Fold con i Migliori Parametri: 0.5785169785169786
Accuratezza sul test set: 0.3191489361702128
```

KNN con dati grezzi

SMOTE L'uso della tecnica SMOTE non sembra sortire effetti di alcun tipo sulla precisione del classificatore KNN, che mantiene prestazioni simili a quelle ottenute con i dati grezzi sia per quanto riguarda i set di validazione, sia per quelle relative al set di prova. È probabile che il metodo per il bilanciamento, in quanto basato sulla creazione di valore sintetici e partendo da classi fortemente sottorappresentate, crei questi nuovi record in punti che non si differenziano abbastanza da quelli precedenti. Questo fa si che i record ai quali risultano più vicini i valori di test non siano diversi – spiegando i risultati ottenuti e rivelandosi inefficace nel migliorare la precisione del classificatore.

```
Accuratezza Media sui Fold con i Migliori Parametri: 0.565974025974026
Accuratezza sul test set: 0.3191489361702128
```

KNN con SMOTE

Random oversampling L'affermazione del paragrafo precedente viene dimostrata dai risultati ottenuti con il random oversampling: la duplicazione dei record delle classi sottorappresentate porta a dei risultati migliori nel set di validazione, in quanto si trovano in posizioni identiche. Per quanto riguarda il set di prova, il risultato invariato mostra che il classificatore risulta insensibile alle tecniche di bilanciamento che fanno uso di oversampling quando si devono gestire classi con differenze di popolazione importante.

```
Accuratezza Media sui Fold con i Migliori Parametri: 0.7236363636363636
Accuratezza sul test set: 0.3191489361702128
```

KNN con random oversampling

Random undersampling La tecnica di random undersampling si mostra invece l'unica con degli effetti sulle effettive capacità del classificatore. Porta infatti un moderato aumento della accuratezza nel test set, in cambio di una forte riduzione di capacità in quelli di validazione.
Questo risultato è dovuto alla forte diminuzione dei dati disponibili in ciascuna classe: gli esempi necessari per un modello di rote learning come il KNN sono ulteriormente ridotti quando si fa uso dei set di validazione, portando a performance minori. Nel caso invece del test di prova la rimozione di elementi che potevano portare a risultati errati si dimostra essere un beneficio ed il modello si rivela più prestante.

```
Accuratezza Media sui Fold con i Migliori Parametri: 0.18351648351648353
Accuratezza sul test set: 0.3404255319148936
```

KNN con random undersampling

4.2 Scaler

In quanto il dataset presenta caratteristiche con scale significativamente diverse, è necessario applicare delle tecniche di scaling per garantire un trattamento equo di tutte le feature durante l'addestramento dei modelli utilizzati. Una caratteristica con valori numerici elevati potrebbe dominare il processo di apprendimento rispetto a caratteristiche con valori più piccoli, anche se queste ultime potrebbero essere più informative.
Le tecniche utilizzate sono 2: la normalizzazione e la standardizzazione La prima riscala i valori nell'intervallo [0, 1], mentre la seconda trasforma i dati in modo che abbiano media zero e deviazione standard unitaria.
La standardizzazione è particolarmente utile quando i dati seguono approssimativamente una distribuzione normale e risulta efficace per algoritmi come SVM e k-NN che utilizzano metriche di distanza. La normalizzazione, d'altra parte, è spesso preferibile quando si hanno distribuzioni non gaussiane o quando il vero range dei dati è sconosciuto, ed è particolarmente adatta per algoritmi come Naive Bayes che assumono distribuzioni di probabilità specifiche. Inoltre avendo scelto ComplementNB() come classificatore Bayesiano, il quale non accetta dei valori negativi, è utilizzabile solo la normalizzazione, in quanto la standardizzazione potrebbe produrre anche valori negativi. Per gli altri classificatori useremo la standardizzazione.
Data la natura abbastanza eterogenea del dataset in analisi, ci aspettiamo che l'applicazione di tecniche di scaling migliori le performance di classificazione, soprattutto per gli algoritmi che usano metriche di distanza o sono sensibili alla scala dei dati.

4.2.1 Ensemble classifier

```
Decision Tree Accuracy:  0.40425531914893614
Decision Tree Precision:  0.3828282828282828
Decision Tree Recall:  0.4161616161616162
Decision Tree F1:  0.3480080921257392

Support Vector Accuracy:  0.06382978723404255
Support Vector Precision:  0.010638297872340425
Support Vector Recall:  0.16666666666666666
Support Vector F1:  0.02

Naive Bayes Accuracy:  0.1702127659574468
Naive Bayes Precision:  0.12779503105590062
Naive Bayes Recall:  0.20113636363636367
Naive Bayes F1:  0.1253968253968254

Hard Voting Accuracy:  0.3829787234042553
Hard Voting Precision:  0.3731060606060606
Hard Voting Recall:  0.3994949494949495
Hard Voting F1:  0.3349248790425261
```

Ensemble con dati grezzi

```
Decision Tree Accuracy:  0.46808510638297873
Decision Tree Precision:  0.40946969696969693
Decision Tree Recall:  0.46414141414141413
Decision Tree F1:  0.4113871635610766

Support Vector Accuracy:  0.425531914893617
Support Vector Precision:  0.22894736842105265
Support Vector Recall:  0.3196969696969697
Support Vector F1:  0.2513888888888889

Naive Bayes Accuracy:  0.1702127659574468
Naive Bayes Precision:  0.14141414141414144
Naive Bayes Recall:  0.20113636363636367
Naive Bayes F1:  0.1279649181609966

Hard Voting Accuracy:  0.46808510638297873
Hard Voting Precision:  0.4142857142857143
Hard Voting Recall:  0.46414141414141413
Hard Voting F1:  0.4098039215686275
```

Ensemble con standardizzazione

Standardizzazione L'applicazione della standardizzazione all'Ensemble Classifier ha prodotto un discreto miglioramento nelle performance del modello. Il risultato conferma che anche in un sistema di hard voting, la standardizzazione dei dati rimane cruciale. L'ensemble trae vantaggio dal miglioramento dei singoli classificatori, e nel nostro caso specifico, la standardizzazione ha permesso di sfruttare al meglio le capacità dell'SVM all'interno dell'ensemble. È interessante notare che dopo la standardizzazione, il voting dell'ensemble produce risultati identici al Decision Tree individuale. Questo suggerisce che con dati standardizzati, il Decision Tree e l'SVM tendono ad essere d'accordo più frequentemente, portando a un voto maggioritario che riflette le loro predizioni condivise.

4.2.2 Decision tree

```
Miglior score in CV: 0.4552
Accuratezza sul test set: 0.4894
Score della cross-validation: [0.42105263 0.31578947 0.43243243 0.48648649 0.48648649]
Accuratezza media CV: 0.4284
Accuratezza del modello: 0.43
```

Decision tree con dati grezzi

Standardizzazione I risultati confermano che il Decision Tree è piuttosto insensibile alla standardizzazione dei dati. Questo comportamento è giustificato dalla teoria degli alberi decisionali, che costruiscono regole di split basate sull'ordine relativo dei valori e non sulla loro scala assoluta.
Possiamo notare che l'accuratezza sul test set mostra un calo abbastanza significativo con i dati standardizzati. Questo potrebbe essere dovuto alla variabilità statistica, in quanto il test set rappresenta solo il 20% dei dati, e la variabilità naturale tra i diversi campionamenti può influenzare notevolmente i risultati.
Per un'applicazione pratica, considerando che la standardizzazione non migliora le performance e potrebbe in alcuni casi peggiorarle, è consigliabile utilizzare il Decision Tree sui dati grezzi.

```
Miglior score in CV: 0.4555
Accuratezza sul test set: 0.3617
Score della cross-validation: [0.42105263 0.31578947 0.43243243 0.48648649 0.51351351]
Accuratezza media CV: 0.4339
Accuratezza del modello: 0.43
Modello addestrato
```

Decision tree con standardizzazione

4.2.3 Naive Bayes

```
Miglior score in CV: 0.4703
Accuratezza sul test set: 0.4043
Score della cross-validation: [0.5        0.5        0.37837838 0.54054054 0.43243243]
Accuratezza media CV: 0.4703
Accuratezza del modello: 0.47
```

Naive Bayes con dati grezzi

Normalizzazione I risultati dimostrano che, contrariamente alle aspetttative teoriche, la normalizzazione ha un effetto negativo sulle performance del classificatore ComplementNB per questo specifico dataset demografico.
Questi risultati suggeriscono che, nonostante ComplementNB richieda dati non negativi, la struttura e le scale originali del dataset, contengono informazioni che vengono perse durante il processo di normalizzazione.
Per l'applicazione pratica è dunque consigliabile utilizzare ComplementNB sui dati grezzi, assicurandosi che non contengano valori negativi.

```
Miglior score in CV: 0.3690
Accuratezza sul test set: 0.2128
Score della cross-validation: [0.36842105 0.36842105 0.37837838 0.40540541 0.32432432]
Accuratezza media CV: 0.3690
Accuratezza del modello: 0.37
```

Naive Bayes con normalizzazione

4.2.4 Support Vector Classifier

```
Accuratezza sul test set: 0.2340
Score della cross-validation: [0.23684211 0.23684211 0.24324324 0.27027027 0.24324324]
Accuratezza media CV: 0.2461
Accuratezza del modello: 0.25
```

SVM con dati grezzi

Standardizzazione I risultati mostrano un miglioramento drammatico delle performance di SVM copo l'applicazione della standardizzazione. L'accuratezza media in cross validation è praticamente raddoppiata; anche l'accuratezza sul test set ha mostrato un incremento notevole.
Questo comportamento è perfettamente in linea con i fondamenti teorici di SVM, che ci dicono che questo algoritmo è fortemente dipendente dalla scala dei dati, poichè utilizza metriche di distanza per trovare l'iperpiano ottimale. La standardizzazione permette a tutte le feature di contribuire equamente alla decisione, indipendentemente dalla loro scala originale.
Questo esperimento conferma l'importanza cruciale della standardizzazione come prerequisito essenziale per ottenere performance ottimali con SVM, soprattutto quando si lavora con dataset che presentano feature a scale molto diverse, come in questo caso.

```
Accuratezza sul test set: 0.4255
Score della cross-validation: [0.52631579 0.5        0.40540541 0.54054054 0.48648649]
Accuratezza media CV: 0.4917
Accuratezza del modello: 0.49
```

SVM con dati standardizzati

4.2.5 KNN Custom

```
Accuratezza Media sui Fold con i Migliori Parametri: 0.5785169785169786
Accuratezza sul test set: 0.3191489361702128
```

KNN Custom con dati grezzi

Standardizzazione Contrariamente alle aspettative teoriche, che ci dicono che gli algoritmi basati su distanza dovrebbero beneficiare della standardizzazione, in questo caso quest'ultima ha prodotto un leggero calo delle performance del KNN Custom. Questo risultato può dipendere da diversi motivi, come le specifiche dell'implementazione del classificatore custom e la natura particolare del dataset. È possibile che per KNN Custom, in questo specifico contesto, le differenze di scala tra le feature contengano informazioni utili per la classificazione che vengono parzialmente perse durante la standardizzazione.

```
Accuratezza Media sui Fold con i Migliori Parametri: 0.5625779625779626
```

KNN Custom con standardizzazione

4.3 Feature Selection

Dai risultati e dalle considerazioni fatte in precedenza risulta certa la presenza nel dataset di più attributi troppo correlati tra loro (le informazioni che contengono sono troppo simili, es: la popolazione del 2022 con quella del 2020) e soprattutto di attributi poco utili al fine della predizione (non permettono di suddividere abbastanza efficacemente le classi). Perciò ora, attraverso le opportune tecniche feature selection, osserveremo come cambiano le prestazioni a seguito della applicazione di questa tecnica di pre-processing.
Le tecniche che abbiamo scelto di utilizzare sono tutte basate su SelectKBest il quale esegue uno scoring di tutti le features e permette di selezionare le prime k migliori. Tale scoring è stato eseguito su 3 test statistici:

- Con **mutual_info**: una volta fornitogli il set di record da analizzare, 'mutual_info_classif' permette di misurare la quantità di informazione condivisa tra ogni feature e la variabile target. Si scelgono infine le prime k features che hanno la quantità di informazione condivisa maggiore.
- Con **test chi2**: una volta fornitogli il set di record da analizzare, il test chi-quadro (chi2) misura l'indipendenza tra ogni feature e la variabile target. Calcola la differenza tra le frequenze osservate e quelle attese per ciascuna feature. Le prime k features con i valori di chi-quadro più alti, che indicano una maggiore dipendenza dalla variabile target, vengono selezionate.
- Con test **ANOVA F-value**: una volta fornitogli il set di record da analizzare, il test (f_classif) calcola il rapporto tra la varianza tra le classi e la varianza all'interno delle classi per ogni feature. Un valore F più alto indica una maggiore capacità della feature di separare le classi in modo significativo. Infine, vengono selezionate le prime k feature con i valori F più elevati, poiché risultano più efficaci nella discriminazione tra le classi.

Queste tre tecniche di riduzione sono state selezionate perché:

- La riduzione tramite mutual info è utile perché in grado di catturare relazioni non lineari tra le caratteristiche e la variabile target. Inoltre, è abbastanza robusto e discreto su una diversa varietà di distribuzioni dei dati e può essere applicato sia a caratteristiche continue che discrete. Misurando quanto una caratteristica riduce l'incertezza sulla variabile target, seleziona le caratteristiche più informative, che è ottimo per ottenere migliori performance.
- La riduzione tramite test chi2 risulta molto utile invece per individuare tutti quegli attributi che hanno una forte relazione statistica con la variabile target, e che quindi permettono sicuramente in maniera più efficace di distinguere le classi durante la predizione.
- La riduzione tramite test ANOVA F-value è utile perché, essendo un test basato sull'analisi della varianza, permette di selezionare le caratteristiche che, mostrando una maggiore differenza di varianza tra le classi, aiutano a distinguerle più efficacemente.

4.3.1 Ensemble Classifier

```
Decision Tree Accuracy:  0.46808510638297873
Decision Tree Precision:  0.40946969696969693
Decision Tree Recall:  0.46414141414141413
Decision Tree F1:  0.4113871635610766

Support Vector Accuracy:  0.23404255319148937
Support Vector Precision:  0.03900709219858156
Support Vector Recall:  0.16666666666666666
Support Vector F1:  0.0632183908045977

Naive Bayes Accuracy:  0.1702127659574468
Naive Bayes Precision:  0.14168391994478952
Naive Bayes Recall:  0.20113636363636367
Naive Bayes F1:  0.12722832722832722

Hard Voting Accuracy:  0.425531914893617
Hard Voting Precision:  0.3718045112781954
Hard Voting Recall:  0.353030303030303
Hard Voting F1:  0.3468137254901961
```

Ensemble con dati grezzi

Test chi2 A seguito della selezione di feature mediante test chi2 è evidente che sia il Nayve Bayes che l'SVM in pratica non percepiscono alcun miglioramento nelle performance, da ciò si può intuire che questa tecnica non sembra particolarmente efficace con questi classificatori. La stessa cosa non può essere detta per l'albero decisionale che vede calare la sua accuratezza e le altre statistiche di più del 10% (quindi un deciso calo di performance). Più o meno la stessa perdita di performance la osserviamo poi nella predizione finale del classificatore ensemble e ciò sottolinea che la riduzione tramite test chi2, a causa dell'inefficacia sull'albero decisionale, non sia ottimale con il nostro classificatore composto.

```
Decision Tree Accuracy:  0.2978723404255319
Decision Tree Precision:  0.226010101010101
Decision Tree Recall:  0.23674242424242423
Decision Tree F1:  0.20056925248576 78

Support Vector Accuracy:  0.23404255319148937
Support Vector Precision:  0.03900709219858156
Support Vector Recall:  0.16666666666666666
Support Vector F1:  0.0632183908045977

Naive Bayes Accuracy:  0.1702127659574468
Naive Bayes Precision:  0.14047619047619048
Naive Bayes Recall:  0.20113636363636367
Naive Bayes F1:  0.12558629110353248

Hard Voting Accuracy:  0.2978723404255319
Hard Voting Precision:  0.22348484848484848
Hard Voting Recall:  0.23522727272727273
Hard Voting F1:  0.19657097288676237
```

Ensemble con test chi2

Mutual info Mediante selezione di feature tramite mutual info notiamo che le predizioni dell'albero decisionale questa volta vengono peggiorate in maniera inferiore rispetto al caso precedente e al contrario il Nayve Bayes sembra aver tratto un leggero beneficio da questa tecnica di riduzione. I risultati del classificatore ensemble in generale peggiorano, soprattutto nella precisione, di conseguenza anche questa tecnica non è adatta per il nostro classificatore.

```
Decision Tree Accuracy:  0.44680851063829785
Decision Tree Precision:  0.3453159041394336
Decision Tree Recall:  0.3893939393939394
Decision Tree F1:  0.3550867910517033

Support Vector Accuracy:  0.23404255319148937
Support Vector Precision:  0.03900709219858156
Support Vector Recall:  0.16666666666666666
Support Vector F1:  0.0632183908045977

Naive Bayes Accuracy:  0.19148936170212766
Naive Bayes Precision:  0.15622895622895625
Naive Bayes Recall:  0.221969696969697
Naive Bayes F1:  0.1459854597109499

Hard Voting Accuracy:  0.40425531914893614
Hard Voting Precision:  0.2989267676767677
Hard Voting Recall:  0.356060606060606
Hard Voting F1:  0.31657848324514987
```

Ensemble con Mutual info

```
Decision Tree Accuracy:  0.46808510638297873
Decision Tree Precision:  0.3680555555555556
Decision Tree Recall:  0.40606060606060607
Decision Tree F1:  0.37092731829573933

Support Vector Accuracy:  0.23404255319148937
Support Vector Precision:  0.03900709219858156
Support Vector Recall:  0.16666666666666666
Support Vector F1:  0.0632183908045977

Naive Bayes Accuracy:  0.2127659574468085
Naive Bayes Precision:  0.1625180375180375
Naive Bayes Recall:  0.2371212121212215
Naive Bayes F1:  0.15968552520276658

Hard Voting Accuracy:  0.44680851063829785
Hard Voting Precision:  0.3244949494949495
Hard Voting Recall:  0.38787878787878793
Hard Voting F1:  0.3447971781305114
```

Ensemble con F-value

F-value Attraverso quest'ultima tecnica di selezione di feature, i risultati sembrano essere molto simili a quelli ottenuti precedentemente, infatti, l'albero decisionale cala leggermente nelle performance mentre il Nayve bayes trae un leggero vantaggio da questa riduzione dimensionale. Tutto ciò porta a un'accuratezza finale leggermente maggiore al prezzo però di un decremento nella precisione del classificatore.

4.3.2 Decision tree

```
Miglior score in CV: 0.4602
Accuratezza sul test set: 0.4681
Score della cross-validation: [0.44736842 0.34210526 0.43243243 0.45945946 0.48648649]
Accuratezza media CV: 0.4336
Accuratezza del modello: 0.43
```

Decision tree con dati grezzi

Test chi2 Con questa tecnica l'accuratezza sul test set crolla quasi del 50% rispetto a quella che si ottiene con il dataset grezzo mentre, durante la cross-validation sul training set, sembra ci sia solo un 'leggero' calo di performance. Questi risultati non ci sorprendono dato che abbiamo già osservato, nel classificatore ensemble, che l'albero decisionale non reagisce bene alla riduzione con test chi2. Possiamo dedurre quindi che si sia adattato troppo ai record di training diminuendo così la sua capacità di generalizzare (nel training set il calo di performance potrebbe essere invece dato dalla perdita di informazioni durante la riduzione).

```
Miglior score in CV: 0.4225
Accuratezza sul test set: 0.2553
Score della cross-validation: [0.39473684 0.42105263 0.43243243 0.40540541 0.37837838]
Accuratezza media CV: 0.4064
Accuratezza del modello: 0.41
```

Decision tree con test chi2

Mutual info A seguito della selezione tramite mutual info l'accuratezza sul test set è leggermente diminuita, così come quella risultante dalla media delle accuratezze ottenute durante la cross-validation. La variabilità nei risultati sul validation set mostra una minore stabilità del classificatore (potrebbe essere dovuta agli attributi selezionati che non permettono un adeguata divisione tra le classi e, come nella tecnica precedente, alla perdita di informazioni utili per l'induzione dell'albero).

```
Miglior score in CV: 0.4331
Accuratezza sul test set: 0.4255
Score della cross-validation: [0.42105263 0.44736842 0.48648649 0.51351351 0.2972973 ]
Accuratezza media CV: 0.4331
Accuratezza del modello: 0.43
```

Decision tree con Mutual info

F-value Anche con la selezione tramite F-value i risultati sia sul test set (anche se in maniera maggiore rispetto al caso precedente) sia durante la cross-validation sembrano leggermente peggiori. Quindi anche questa tecnica peggiora le performance di questo classificatore e di conseguenza possiamo constatare che quest'ultimo non beneficia da nessuna delle tre tecniche di riduzione (dovuto come già detto ad attributi non ottimi per l'induzione dell'albero e la perdita di informazioni durante la selezione).

```
Miglior score in CV: 0.4331
Accuratezza sul test set: 0.4043
Score della cross-validation: [0.42105263 0.44736842 0.48648649 0.51351351 0.2972973 ]
Accuratezza media CV: 0.4331
Accuratezza del modello: 0.43
```

Decision tree con F-value

4.3.3 Naive Bayes

```
Miglior score in CV: 0.4703
Accuratezza sul test set: 0.4043
Score della cross-validation: [0.5        0.5        0.37837838 0.54054054 0.43243243]
Accuratezza media CV: 0.4703
Accuratezza del modello: 0.47
```

Naive Bayes con dati grezzi

Test chi2 A seguito della selezione di feature effettuata sulla base del test chi2 si nota un grosso peggioramento sia nelle predizioni sul test set sia su quelle del training set ottenute durante cross-validation, osservando in entrambi i casi un calo prestazionale di circa 15%. Tra i precedenti classificatori, insieme all'albero decisionale dell'ensemble, sembra che sia tra i modelli di classificazione meno adatti per questa tecnica di riduzione.
Per questo classificatore la selezione di feature dovrebbe risultare molto utile, anche perché elimina potenziali features condizionalelmente dipendenti tra loro (e che quindi violano la sua assunzione di dipendenza condizionale tra gli attributi). Il forte calo però sembra sottolineare invece che gli attributi eliminati erano effettivamente utili al classificatore e che quindi una loro eliminazione porti a perdere informazioni significative.

```
Miglior score in CV: 0.3158
Accuratezza sul test set: 0.2553
Score della cross-validation: [0.26315789 0.31578947 0.40540541 0.2972973  0.2972973 ]
Accuratezza media CV: 0.3158
Accuratezza del modello: 0.32
```

Naive Bayes con test chi2

Mutual info Rispetto al caso precedente, la selezione mediante mutual info sembra migliorare le predizioni sul test set di un 2% ma comunque portare a un sostanzioso peggioramento per quelle effettuate sul validation set. Su quest'ultimo si nota però che con questa tecnica i risultati sono comunque meno variabili (rispetto alle predizioni ottenute con le altre tecniche di riduzione) e si attestano più o meno tutte sul 40% di accuratezza. Questo, insieme all'accuratezza superiore, sottolinea che scegliere gli attributi in base all'informazione mutua sia sicuramente il metodo più efficace per il nostro classificatore (tra quelli osservati).

```
Miglior score in CV: 0.3905
Accuratezza sul test set: 0.4255
Score della cross-validation: [0.39473684 0.36842105 0.40540541 0.35135135 0.43243243]
Accuratezza media CV: 0.3905
Accuratezza del modello: 0.39
```

Naive Bayes con Mutual info

F-value Con questa tecnica si può osservare una performance risultante molto simile a quella che si ottiene su dataset grezzo, con un leggerissimo decremento delle prestazioni durante la cross-validation e un decremento del 4% sul test set. Questa tecnica, quindi, non è adatta al nostro classificatore.

```
Miglior score in CV: 0.4600
Accuratezza sul test set: 0.3617
Score della cross-validation: [0.42105263 0.47368421 0.48648649 0.54054054 0.37837838]
Accuratezza media CV: 0.4600
Accuratezza del modello: 0.46
```

Naive Bayes con F-value

4.3.4 Support Vector Machine

```
Accuratezza sul test set: 0.2340
Score della cross-validation: [0.23684211 0.23684211 0.24324324 0.27027027 0.24324324]
Accuratezza media CV: 0.2461
Accuratezza del modello: 0.25
```

SVM con dati grezzi

Test chi2 La selezione di feature tramite test chi2 non ha comportato neanche la minima differenza di performance su questo classificatore. La riduzione di dimensionalità dovrebbe portare all'SVM una maggiore stabilita e la possibilità di concentrarsi sulle caratteristiche più utili ai fini della predizione. Si deduce quindi che gli attributi eliminati non erano per niente significativi per questo modello e che già l'SVM li ignorava durante le predizioni.

```
Accuratezza sul test set: 0.2340
Score della cross-validation: [0.23684211 0.23684211 0.24324324 0.27027027 0.24324324]
Accuratezza media CV: 0.2461
Accuratezza del modello: 0.25
```

SVM con test chi2

Mutual info Anche dopo questa tecnica di selezione il risultato è il medesimo ottenuto precedentemente. Neanche scegliere gli attributi che forniscono le maggiori informazioni sulle classi è utile per incrementare la sua accuratezza.

```
Accuratezza sul test set: 0.2340
Score della cross-validation: [0.23684211 0.23684211 0.24324324 0.27027027 0.24324324]
Accuratezza media CV: 0.2461
Accuratezza del modello: 0.25
```

SVM con mutual info

F-value Infine, con la selezione mediante F-value, osservando che i risultati sono anche in questo caso identici a quelli ottenuti precedentemente, possiamo quindi asserire che nessuna delle 3 tecniche di riduzione di dimensionalità sia efficace per questo classificatore.

```
Accuratezza sul test set: 0.2340
Score della cross-validation: [0.23684211 0.23684211 0.24324324 0.27027027 0.24324324]
Accuratezza media CV: 0.2461
Accuratezza del modello: 0.25
```

SVM con F-value

4.3.5 KNN Custom

Test chi2 La tecnica di selezione di feature tramite test chi2 non sembra aver dato dei buoni risultati, al contrario sembra che il KNN reagisca male a questa tecnica visto questo decremento delle performance sia nel training set con cross validation sia poi nel test set.
Come avviene per altri tipi di classificatori, la selezione di feature dovrebbe teoricamente apportare benefici anche a questo tipo di modelli. Quindi, riteniamo che il calo nelle prestazioni possa essere attribuito alla mancanza di informazioni utili a fini predittivi, derivante dalla selezione degli attributi. In altre parole, la riduzione potrebbe aver eliminato variabili che, pur essendo considerate meno significative, risultavano comunque utili per una corretta separazione delle classi.

```
Accuratezza Media sui Fold con i Migliori Parametri: 0.5785169785169786
Accuratezza sul test set: 0.3191489361702128
```

KNN Custom con dati grezzi

```
Accuratezza Media sui Fold con i Migliori Parametri: 0.5250173250173249
Accuratezza sul test set: 0.23404255319148937
```

KNN Custom con test chi2

Mutual info Anche la selezione di feature tramite mutual info sembra riconfermare che questo classificatore non stia ottenendo alcun beneficio da queste tecniche di preprocessing. Il calo prestazionale è deciso e anche molto simile al caso precedente, soprattutto sul test set. Per quanto riguarda il validation set, si ha anche in questo caso un peggioramento di performance e ciò ci conferma che probabilmente durante la selezione si sono perse anche informazioni utili per la predizione.

```
Accuratezza Media sui Fold con i Migliori Parametri: 0.5355509355509357
Accuratezza sul test set: 0.2553191489361702
```

KNN Custom con mutual info

F-value La tecnica con test ANOVA dimostra di ottenere risultati ancora peggiori delle tecniche precedenti. Questa quindi fra le tre tecniche utilizzate è quella meno adatta al nostro classificatore e conferma infine che nessuna delle tecniche di selezione di feature permette al classificatore di ottenere una performance migliore rispetto al dataset grezzo. In quest'ultimo caso poi si ha un netto peggioramento nelle performance sul validation set che sottolinea che il modello non è riuscito a adattarsi bene agli attributi selezionati e, come conseguenza, questo sicuramente ha peggiorato ulteriormente le predizioni sul test set.

```
Accuratezza Media sui Fold con i Migliori Parametri: 0.4228690228690229
Accuratezza sul test set: 0.2127659574468085
```

KNN Custom con F-value

5 Ricerca della combinazione di pre-processing ottimale

5.1 Ensemble classifier

Selezione Attributi	Bilanciamento	Standardizzazione	Rimozione Outlier	Accuratezza
No	No	Non standardizzato	No	0.43
Chi-Squared	SMOTE	Standardizzato	Sì	0.19
Chi-Squared	SMOTE	Standardizzato	No	0.21
Chi-Squared	Random Over Sampling	Standardizzato	Sì	0.19
Chi-Squared	Random Over Sampling	Standardizzato	No	0.21
Chi-Squared	Random Under Sampling	Standardizzato	Sì	0.19
Chi-Squared	Random Under Sampling	Standardizzato	No	0.12
Mutual Information	SMOTE	Standardizzato	Sì	**0.47**
Mutual Information	SMOTE	Standardizzato	No	0.43
Mutual Information	Random Over Sampling	Standardizzato	Sì	0.38
Mutual Information	Random Over Sampling	Standardizzato	No	0.40
Mutual Information	Random Under Sampling	Standardizzato	Sì	0.28
Mutual Information	Random Under Sampling	Standardizzato	No	0.32
F - Classifier	SMOTE	Standardizzato	Sì	0.45
F - Classifier	SMOTE	Standardizzato	No	0.43
F - Classifier	Random Over Sampling	Standardizzato	Sì	0.36
F - Classifier	Random Over Sampling	Standardizzato	No	0.40
F - Classifier	Random Under Sampling	Standardizzato	Sì	0.29
F - Classifier	Random Under Sampling	Standardizzato	No	0.28

Dalla tabella, prodotto della ricerca per la combinazione ottimale dei metodi di preprocessing per il classificatore multiplo, si possono notare delle tendenze interessanti.
La prima, e più evidente, è il peggioramento notevole delle prestazioni del classificatore con la selezione degli attributi effettuata con il metodo Chi-Squared. Poiché questo metodo sceglie gli attributi che risultano più legati alla classe, e nel dataset nessuno degli attributi mostra forti correlazioni con essa, è ragionevole ipotizzare che abbia effettuato una selezione eccessiva del numero di attributi, portando ad una riduzione dell'accuratezza del classificatore.
La seconda è che, gli altri due metodi portano a risultati simili, largamente determinati dal metodo di bilanciamento. Si può notare che SMOTE risulti la tecnica più adatta per il classificatore multiplo, poiché la variabilità che introduce nel dataset generando dati sintetici per classi fortemente sottorappresentate viene mediata grazie ai 3 classificatori base, rendendolo più robusto. Il random oversampling mostra performance simili, anche se leggermente ridotte – confermando l'ipotesi che il sovracampionamento sia il metodo da preferire per questo dataset. Il sottocampionamento random, come evidenziato nella sezione dedicata al bilanciamento, risulta inadatto in quanto rimuove troppi elementi per poter addestrare correttamente il modello.
Infine, si può notare come generalmente la rimozione dei valori estremi porti a una maggiore capacità di classificazione – con 2 eccezioni. La prima è legata al selettore Chi-Squared. In quel caso avere delle informazioni in più, anche se di bassa qualità per la classificazione, risulta positivo e porta a guadagni marginali. L'altra è dovuta al random undersampling: anche in questo caso la penuria di dati nel set di addestramento è fa si che l'aggiunta dei valori estremi sia un beneficio.

Non risulta sorprendente, quindi, che il risultato ottimale sia ottenuto usando il selettore con Mutual Information – capace di identificare dipendenze non lineari fra gli attributi per lasciare solo quelli che aggiungono la maggiore quantità di informazione – combinato con il metodo SMOTE e l'eliminazione dei valori estremi. Questo porta ad un moderato guadagno di prestazioni rispetto al dataset grezzo, da 0.43 a 0.47, giustificando lo sforzo per migliorare il dataset.

5.2 Decision Tree

Selezione Attributi	Bilanciamento	Standardizzazione	Rimozione Outlier	Accuratezza
No	No	Non standardizzato	No	**0.46**
Chi-Squared	SMOTE	Standardizzato	Sì	0.15
Chi-Squared	SMOTE	Standardizzato	No	0.13
Chi-Squared	Random Over Sampling	Standardizzato	Sì	0.26
Chi-Squared	Random Over Sampling	Standardizzato	No	0.26
Chi-Squared	Random Under Sampling	Standardizzato	Sì	0.26
Chi-Squared	Random Under Sampling	Standardizzato	No	0.26
Mutual Information	SMOTE	Standardizzato	Sì	0.40
Mutual Information	SMOTE	Standardizzato	No	0.26
Mutual Information	Random Over Sampling	Standardizzato	Sì	0.36
Mutual Information	Random Over Sampling	Standardizzato	No	0.40
Mutual Information	Random Under Sampling	Standardizzato	Sì	0.26
Mutual Information	Random Under Sampling	Standardizzato	No	0.19
F - Classifier	SMOTE	Standardizzato	Sì	0.43
F - Classifier	SMOTE	Standardizzato	No	0.38
F - Classifier	Random Over Sampling	Standardizzato	Sì	0.36
F - Classifier	Random Over Sampling	Standardizzato	No	0.43
F - Classifier	Random Under Sampling	Standardizzato	Sì	0.17
F - Classifier	Random Under Sampling	Standardizzato	No	0.28

Nel caso della ricerca per l'albero decisionale gli andamenti si dimostrano simili a quelli visti in precedenza. Il selettore Chi-Squared rimuove troppi attributi ed impatta negativamente la performance, mentre gli altri ottengono risultati simili. Il sottocampionamento ha anch'esso un'influenza negativa sia sul selettore Mutual Information che sul F-Classifier.
Si separa dal precedente poiché non si trovano combinazioni che portino ad un risultato migliore del dataset grezzo – indicando che le tecniche non siano adatte al classificatore. Da notare il fatto che per quanto riguarda l'albero decisionale l'effetto della rimozione dei valori estremi è molto meno prevedibile, migliorando o peggiorando i risultati in maniera significativa. Le eccezioni in questo caso sono quelle legate al primo selettore di attributi: in quei casi l'effetto è nullo o particolarmente ridotto.

Per questo classificatore i risultati migliori si ottengono a partire dai dati grezzi, poiché la rimozione di informazioni dovuta alla selezione degli attributi o al sottocampionamento e l'aggiunta di variabilità dovuta all'oversampling – non mediate tramite un meccanismo di voto – portano ad una complicazione dei dati per l'albero decisionale. In questo caso, lo sforzo si rivela inutile.

5.3 Naive Bayes

Selezione Attributi	Bilanciamento	Normalizzazione	Rimozione Outlier	Accuratezza
No	No	Non normalizzato	No	0.40
Chi-Squared	SMOTE	Normalizzato	Sì	0.38
Chi-Squared	SMOTE	Normalizzato	No	0.40
Chi-Squared	Random Over Sampling	Normalizzato	Sì	0.40
Chi-Squared	Random Over Sampling	Normalizzato	No	**0.42**
Chi-Squared	Random Under Sampling	Normalizzato	Sì	0.40
Chi-Squared	Random Under Sampling	Normalizzato	No	0.36
Mutual Information	SMOTE	Normalizzato	Sì	0.19
Mutual Information	SMOTE	Normalizzato	No	0.32
Mutual Information	Random Over Sampling	Normalizzato	Sì	0.19
Mutual Information	Random Over Sampling	Normalizzato	No	0.17
Mutual Information	Random Under Sampling	Normalizzato	Sì	0.15
Mutual Information	Random Under Sampling	Normalizzato	No	0.15
F - Classifier	SMOTE	Normalizzato	Sì	0.19
F - Classifier	SMOTE	Normalizzato	No	0.17
F - Classifier	Random Over Sampling	Normalizzato	Sì	0.19
F - Classifier	Random Over Sampling	Normalizzato	No	0.17
F - Classifier	Random Under Sampling	Normalizzato	Sì	0.17
F - Classifier	Random Under Sampling	Normalizzato	No	0.17

Questo classificatore richiede l'uso della normalizzazione al posto della standardizzazione, in quanto l'algoritmo richiede valori esclusivamente positivi per funzionare, il che influenza le prestazioni dei metodi rispetto a quanto visto nel caso degli altri classificatori.
Per il Naïve Bayes si nota una inversione di tendenza rispetto a quelli visti in precedenza. In questo caso la forte riduzione di attributi che si ottiene utilizzando il selettore Chi-Squared risulta ottimale, mentre gli altri metodi portano ad una forte riduzione di prestazioni, quasi sempre dimezzando il risultato ottenuto sul dataset grezzo.
Applicato il selettore, i metodi di bilanciamento hanno degli effetti minori sull'accuratezza del classificatore bayesiano, senza poter distinguere un metodo che risulti particolarmente più adatto. I dati migliori si ottengono con il sovracampionamento random, ma il guadagno è sufficientemente ridotto da poter essere attribuibile ad una buona, seppur casuale, scelta dei valori da duplicare.
L'eliminazione dei valori estremi ha anch'essa una influenza minore: poiché il classificatore è basato sulla probabilità gli outlier hanno una influenza ridotta, e difficilmente prevedibile.

Si conclude quindi che la combinazione ottimale di tecniche per questo classificatore è una selezione Chi-Squared con random oversampling e senza la rimozione dei valori estremi.

5.4 Support Vector Machine

Selezione Attributi	Bilanciamento	Standardizzazione	Rimozione Outlier	Accuratezza
No	No	Non standardizzato	No	0.25
Chi-Squared	SMOTE	Standardizzato	Sì	0.50
Chi-Squared	SMOTE	Standardizzato	No	0.42
Chi-Squared	Random Over Sampling	Standardizzato	Sì	0.46
Chi-Squared	Random Over Sampling	Standardizzato	No	0.41
Chi-Squared	Random Under Sampling	Standardizzato	Sì	0.33
Chi-Squared	Random Under Sampling	Standardizzato	No	0.34
Mutual Information	SMOTE	Standardizzato	Sì	0.55
Mutual Information	SMOTE	Standardizzato	No	0.52
Mutual Information	Random Over Sampling	Standardizzato	Sì	**0.66**
Mutual Information	Random Over Sampling	Standardizzato	No	0.62
Mutual Information	Random Under Sampling	Standardizzato	Sì	0.33
Mutual Information	Random Under Sampling	Standardizzato	No	0.38
F - Classifier	SMOTE	Standardizzato	Sì	0.53
F - Classifier	SMOTE	Standardizzato	No	0.50
F - Classifier	Random Over Sampling	Standardizzato	Sì	0.63
F - Classifier	Random Over Sampling	Standardizzato	No	0.62
F - Classifier	Random Under Sampling	Standardizzato	Sì	0.33
F - Classifier	Random Under Sampling	Standardizzato	No	0.36

La combinazione di Pre-Processing migliore per l'SVM è quindi:

- Selezione Features: Mutual Information
- Standardizzazione
- Bilanciamento: Random Over Sampling
- Rimozione degli Outliers attiva

La selezione delle feature tramite Mutual Information permette di mantenere solo le variabili più rilevanti per la classificazione, eliminando quelle ridondanti o poco informative, cosa particolarmente utile in modelli come SVM che possono risentire di feature irrilevanti.
La standardizzazione è essenziale perché il dataset contiene feature con scale molto diverse, come la popolazione espressa in milioni e la densità in valori pi'u piccoli. Poichè SVM utilizza il prodotto scalare per costruire il margine di separazione tra le classi, è fondamentale che tutte le feature abbiano la stessa scala per evitare che quelle con valori più grandi abbiano un impatto sproporzionato.
Il bilanciamento con Random Over Sampling è necessario quando le classi del dataset non sono equilibrate. In questo caso, invece di eliminare dati come farebbe l'undersampling, vengono duplicati alcuni campioni della classe minoritaria per garantire che l'SVM non favorisca eccessivamente la classe dominante.
Infine, la rimozione degli outlier è attivata perchè SVM è molto sensibile a valori anomali, che possono influenzare il posizionamento del margine di separazione e ridurre la capacità del modello di generalizzare sui nuovi dati. Eliminando gli outlier, si ottiene un modello più stabile e affidabile.

5.5 KNN Custom

Selezione Attributi	Bilanciamento	Standardizzazione	Rimozione Outlier	Accuratezza
No	No	Non standardizzato	No	0.32
Chi-Squared	SMOTE	Standardizzato	Sì	0.21
Chi-Squared	SMOTE	Standardizzato	No	0.21
Chi-Squared	Random Over Sampling	Standardizzato	Sì	0.23
Chi-Squared	Random Over Sampling	Standardizzato	No	0.23
Chi-Squared	Random Under Sampling	Standardizzato	Sì	0.29
Chi-Squared	Random Under Sampling	Standardizzato	No	0.19
Mutual Information	SMOTE	Standardizzato	Sì	0.28
Mutual Information	SMOTE	Standardizzato	No	0.28
Mutual Information	Random Over Sampling	Standardizzato	Sì	0.32
Mutual Information	Random Over Sampling	Standardizzato	No	0.36
Mutual Information	Random Under Sampling	Standardizzato	Sì	0.26
Mutual Information	Random Under Sampling	Standardizzato	No	0.26
F - Classifier	SMOTE	Standardizzato	Sì	0.30
F - Classifier	SMOTE	Standardizzato	No	0.27
F - Classifier	Random Over Sampling	Standardizzato	Sì	0.34
F - Classifier	Random Over Sampling	Standardizzato	No	**0.36**
F - Classifier	Random Under Sampling	Standardizzato	Sì	0.30
F - Classifier	Random Under Sampling	Standardizzato	No	0.36

Per quanto riguarda il KNN si può notare che il metodo di selezione degli attributi che porta ai migliori risultati è lo F-Classifier. Questo sistema, che valuta la varianza dell'attributo all'interno di una classe e tra le classi, consente di mantenere nel dataset quelli che sono meglio in grado di separare le classi – e quindi la cui distanza ha il maggiore effetto sulla classificazione. Gli altri metodi portano a risultati moderatamente peggiori, ma paragonabili.
Tra i metodi di bilanciamento quello che risulta più efficace è il sovracampionamento random, che solo considerando il selettore Chi-Squared non porta ai risultati migliori. In quel caso, considerando la penuria di attributi sui quale calcolare le distanze, il modello risulta meno in grado di gestire la variabilità aggiunta dai metodi di oversampling, e si ottengono risultati migliori mantenendo il maggior numero possibile di record veri senza generarne di nuovi.
L'eliminazione degli outlier ha degli effetti moderati nel caso del KNN: poiché questi valori sono così lontani dagli altri, hanno una limitata capacità di influenzare le classi che vengono assegnate ai nuovi.

Per questo modello, i risultati migliori si ottengono con lo F-Classifier il sovracampionamento random e senza togliere i valori estremi, il che porta ad un moderato guadagno di 0.04 in termini di capacità di classificazione per il modello.

6 Conclusioni

In conclusione, il progetto ha esplorato diverse tecniche di machine learning per la classificazione di nazioni in base al continente di appartenenza, utilizzando un dataset contenente informazioni demografiche e geografiche. L'analisi dei dati ha rivelato uno sbilanciamento tra le classi e la presenza di outlier, che hanno influenzato le prestazioni dei classificatori
Sono stati implementati e testati diversi classificatori, tra cui Naive Bayes, Support Vector Machine (SVM), Decision Tree e un classificatore multiplo custom (Ensemble Classifier), sia sul dataset grezzo che dopo l'applicazione di tecniche di pre-processing
Le tecniche di pre-processing includevano bilanciamento dei dati (SMOTE, random oversampling, random undersampling), scaling (normalizzazione e standardizzazione) e feature selection (basata su test Chi-quadro, mutual information e test ANOVA F-value)
La ricerca della combinazione ottimale di pre-processing ha portato a risultati diversi a seconda del classificatore utilizzato:

- **Ensemble Classifier**: La combinazione migliore è risultata essere la selezione di attributi con Mutual Information, il metodo SMOTE per il bilanciamento e la rimozione degli outlier, portando ad un'accuratezza di 0.47
- **Decision Tree**: Nessuna combinazione di pre-processing ha superato le prestazioni ottenute con il dataset grezzo, con un'accuratezza di 0.46, indicando che le tecniche utilizzate non sono adatte a questo classificatore
- **Naive Bayes**: La combinazione ottimale è stata la selezione Chi-Squared con random oversampling e senza la rimozione degli outlier, portando un'accuratezza di 0.42
- **Support Vector Machine**: La combinazione migliore ha incluso la selezione di feature tramite Mutual Information, standardizzazione, bilanciamento con Random Over Sampling e rimozione degli outlier, raggiungendo un'accuratezza di 0.66
- **KNN Custom**: La combinazione più efficace è risultata essere l'utilizzo del selettore F-Classifier, il sovracampionamento random e la non rimozione degli outlier, raggiungendo un'accuratezza di 0.36

In generale, i risultati ottenuti evidenziano l'importanza del pre-processing dei dati per migliorare le prestazioni dei classificatori, ma anche la necessità di scegliere le tecniche più appropriate in base alle caratteristiche del dataset e all'algoritmo utilizzato In particolare, la **Support Vector Machine** ha beneficiato maggiormente delle tecniche di pre-processing, raggiungendo l'accuratezza più elevata grazie alla combinazione ottimale di selezione delle feature con Mutual information, standardizzazione, bilanciamento con Random over sampling e rimozione degli outlier.