Report Progetto Data e Web Mining

Niccolo' Pirillo

Matricola: 890361

Analisi Esplorativa dei Dati

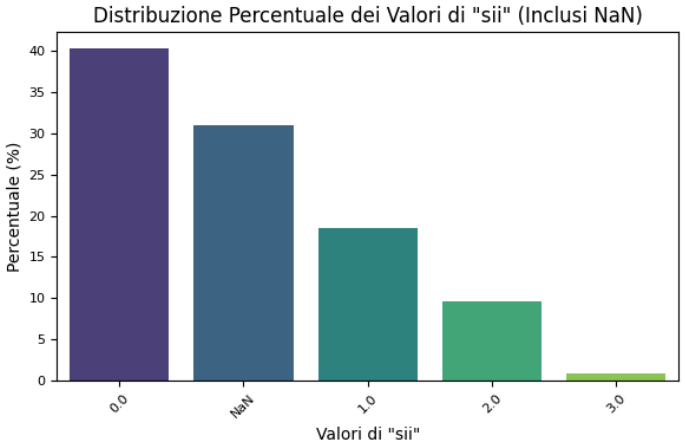
1) Caricamento e Panoramica Preliminare dei Dati

L'esplorazione preliminare di **train** e **test** mi ha permesso immediatamente di comprendere il carattere generale dei dati e di notare alcune **peculiarità** del dataset che saranno esplorate più approfonditamente nelle fasi successive. Tra queste c'è un'elevata **presenza di valori mancanti**, **valori potenzialmente anomali** e le uniche **colonne categoriali** rappresentano esclusivamente le stagioni.

Inoltre il set di **test** presenta molte meno categorie rispetto al train, tra cui il target "sii". Test sara' quindi esclusivamente dedicato alla **competizione Kaggle** e siccome lo **score** restituito non rappresenta i valori effettivi sull'accuratezza del modello, prendero' una **porzione del train** per generare un set di test contenente i valori da predire.

2) Distribuzione del Target (SII)

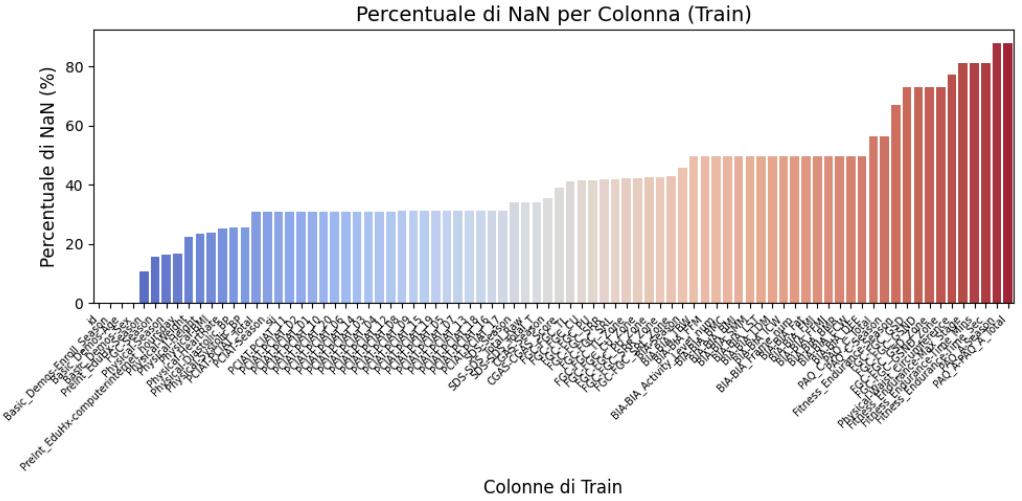
La **distribuzione** del target presenta criticità significative a causa di un'elevata di **valori mancanti** che supera il 30% e un **forte sbilanciamento** delle varie classi che vede dominare la classe "0" sulle altre, fino ad una frequenza minore del 1% per quanto riguarda la classe "3".



3) Valori Mancanti

Il dataset train ha una percentuale **media di dati mancanti** che supera il 40% con alcune colonne che **superano il 70-80%**, e' necessario quindi valutare l'**importanza delle features** per considerare se mantenerle o meno in quanto un'imputazione troppo **complessa** potrebbe portare ad un

sovradattamento ai dati di training mentre una troppo **semplice** potrebbe introdurre **bias**. Inoltre è possibile notare alcuni gruppi di misurazioni che sembrano avere **pattern simili** nei valori mancanti.

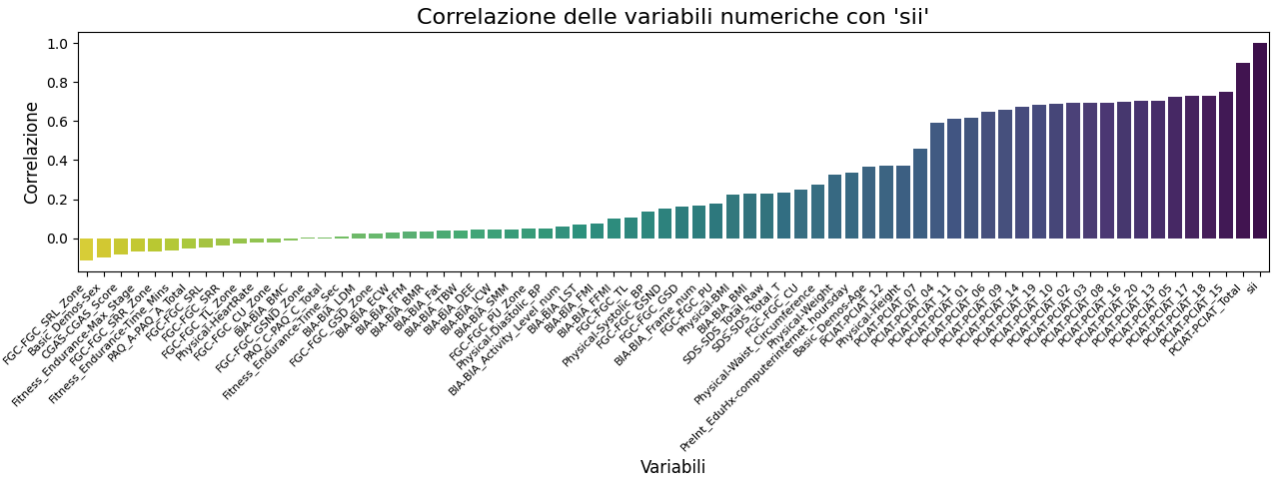


4) Colonne mancanti in Test

Nonostante le predizioni siano basate anche sui valori di **PCIAT-PCIAT_Total** e le relative 20 **colonne PCIAT**, tutte queste colonne **non sono presenti in test**, di conseguenza si può considerare di eliminare queste colonne e concentrarsi sulle altre per le predizioni.

5) Correlazione con 'sii'

Il grafico mostra una **correlazione** delle **colonne numeriche** con la colonna da predire. Alcune mostrano uno score negativo indicando una relazione inversa, altre presentano uno **score elevato** come nelle colonne **PCIAT**



6) Conclusioni e prossimi passi

L'analisi ha evidenziato criticità come **valori mancanti**, **sbilanciamento del target** e **differenze tra train e test**. I prossimi passi includono la gestione dei dati mancanti, il bilanciamento del dataset e la selezione delle caratteristiche più rilevanti per costruire un modello predittivo efficace.

Pulizia e preparazione dei dati

1) **Divisione dei set**

I dataset sono stati divisi in:

- **train_x / train_y** : utilizzati per addestrare e stimare le capacita' del modello
- **test_x :** utilizzato per fare le predizioni sulla competizione Kaggle
- **train_test_x / train_test_y** : utilizzati per valutare il modello

Dopo diversi tentativi, ho osservato che **mantenere le colonne PCIAT** in tutti i dataset e imputarle successivamente portava a **sovrastimare** i modelli predittivi. Questo si traduceva in alte prestazioni nei set di validazione, ma bassi nei test e nella competizione Kaggle. Per questo motivo, ho scelto di **mantenere solo le colonne comuni** tra *train* e *test*, **eliminando le PCIAT**, in linea con la natura della competizione, che fornisce dati senza queste colonne.

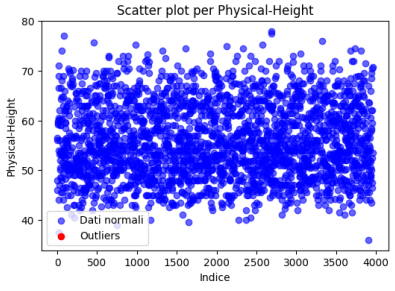
2) **Codifica delle colonne categoriali**

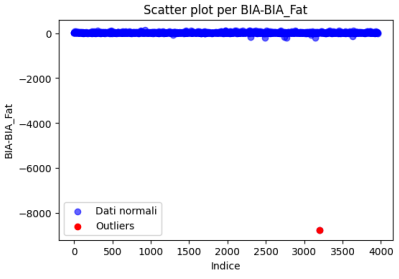
Il primo approccio e' stato la **One Hot Encoding**, richiedeva però che i dati non contenessero valori mancanti, ma allo stesso tempo, i procedimenti successivi richiedevano solo valori numerici. Allora ho optato inizialmente per una **mappatura numerica alle variabili categoriali** legate alle stagioni. Il **tentativo di applicare la OHE** dopo l'imputazione ha invece peggiorato le performance dei modelli aumentando in modo significativo il numero di colonne e complessità nei dati.

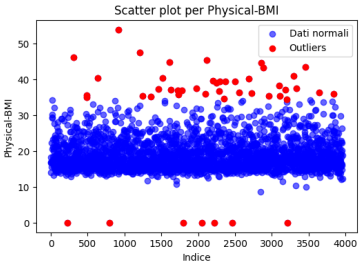
3) **Outliers e nan**

Inizialmente sono state **eliminate** le colonne che contenevano più dell'**80% di valori mancanti** in quanto non apportavano informazioni significative.

Nell'analisi esplorativa dei dati sono emersi **possibili valori anomali** riguardanti l'altezza, peso, BMI, pressione arteriosa e frequenza cardiaca e **valori negativi**. Per indagare su queste feature ho deciso di utilizzare lo **Z-score** che permette di vedere quanto un dato si discosta dalla media. Alcune features si sono mostrate regolari, altre presentano anomalie. Dopo un'analisi degli outliers rilevati, accompagnata ad altre colonne per comprenderne meglio il contesto, ho deciso di **trasformare solo i valori più estremi o chiaramente sbagliati in nan** che verranno imputati nella fase successiva.







4) **Imputazione X/Y separata**

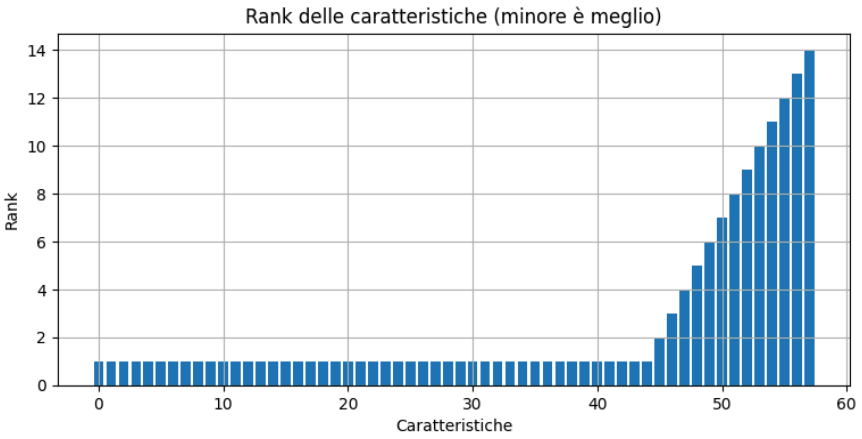
Data l'alta percentuale di istanze prive del valore 'sii' nel set di training ho deciso di non eliminarle ma **imputare strategicamente i dati**. Dopo vari test ho osservato che una imputazione unica portava a sovrastimare le prestazioni sul validation set rispetto al set di test. Per mitigare questa discrepanza ho deciso di **imputare separatamente** train_X da train_Y con 2 modelli diversi in modo da **migliorare la generalizzazione** e **bilanciare le etichette di train_y** attraverso l'utilizzo di Naive

Bayes che fornisce delle stime di probabilità di appartenenza a ciascuna classe, **riducendo il rischio di bias verso la classe maggioritaria**.

Inoltre **train_test_x, test_x e train_x** sono stati imputati da **KNNimputer addestrato solo su train_x** al fine di **non coinvolgere informazioni dai set di test** che non dovrebbero influenzare il processo di addestramento.

5) **Features Subset Selection**

Per selezionare le caratteristiche piu' rilevanti da train_x ho utilizzato la **Recursive Feature Elimination Cross-Validation,** che mi ha permesso di eliminare le feature meno significative, riducendo il rischio di overfitting.



Modelli Predittivi

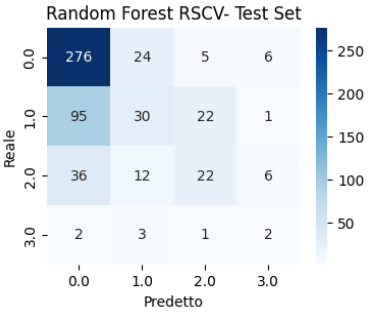
Per scegliere i due modelli richiesti dal progetto ho fatto dei **test su su una serie di modelli** per vedere **in fase preliminare** quale di questi si comportava meglio sui dati di validazione e test. Tra i vari metodi testati ci sono KNN, Random Forest e AdaBoost, XGboost, CatBoost e LightGBM. **Tutti i metodi hanno prodotto risultati simili** sia **sul validation set che nel test** set mostrando una buona **capacità di generalizzazione** e che non si sta adattando troppo ai dati di training.

1) **Random Forest**

Tra i metodi di Scikit-Learn, Random Forest e' quello che si comportava meglio sia sui set di test, validation che nella competizione Kaggle. Inizialmente ho provato a usare il classificatore con dei valori standard, notando immediatamente un leggero **aumento di prestazioni** con il parametro **class_weight = 'balanced'** che **attutisce lo sbilanciamento delle classi del dataset** assegnando un peso maggiore alle classi minoritarie, migliorando F1-score e recall. Per massimizzare le prestazioni del modello ho effettuato il **tuning dei parametri** con una **random search con cross validation** che ha portato un leggero aumento di accuratezza nel test set (da 59.9% a 60.8%) e un piccolo miglioramento nella classificazione della classe 2.

Tuttavia il modello soffre molto lo **sbilanciamento delle classi minoritarie**, in particolare **la gestione della classe 3 rimane critica**, con una recall del 25% nel set di test, il che significa che il modello continua a ignorarla in molti casi.

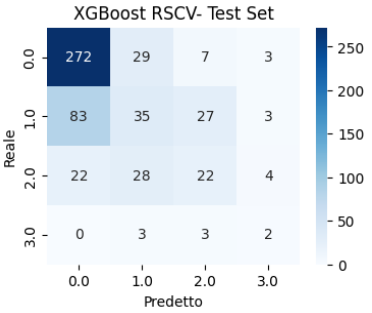
	Validation Accuracy	Test Accuracy
Random Forest	64.03%	59.85%
Random Forest RSCV	63.56%	60.77%



2) XGBoost

In contrasto con modello di bagging scelto nel punto precedente ho scelto XGboost per vedere come i dati si comportavano con un modello di boosting. Il modello ottimizzato ha mostrato un **miglioramento dell'accuratezza** rispetto alla versione base, ma continua a presentare **difficoltà nel distinguere le classi 1 e 2**, che vengono spesso confuse. L'analisi della confusion matrix evidenzia che la **classe 0 viene predetta con maggiore precisione**, mentre la **classe 3 soffre per la scarsità di dati**.

	Validation Accuracy	Test Accuracy
XGBoost	65.14%	59.48%
XGBoost RSCV	67.03%	60.09%



3) Analisi delle istanze e differenze

Nell'analisi delle **istanze correttamente classificate** con maggiore confidenza da parte di **Random Forest**, è emerso che la maggior parte dei casi riguarda bambini tra i 5 e i 7 anni. Questo potrebbe suggerire che il modello sia **più accurato nel classificare bambini più piccoli**. Le istanze erroneamente classificate, invece, evidenziano una **difficoltà del modello** nel classificare casi con **punteggi CGAS bassi** o **misure fisiche non standard**.

Per quanto riguarda **XGBoost**, anche in questo caso il modello ha una buona **capacità di classificare soggetti di età più piccola**, ma mostra buone performance anche nei **soggetti con punteggi CGAS ridotti**. Le istanze erroneamente classificate sembrano invece essere legate a una **difficoltà nel classificare i soggetti di sesso maschile**.

In generale, le performance di Random Forest e XGBoost sono risultate simili, con **XGBoost leggermente migliore** sul validation set, ma le prestazioni sui test set sono state quasi identiche. Entrambi i modelli gestiscono bene la classe maggioritaria, ma **faticano a classificare correttamente le classi minoritarie**, probabilmente a causa dello sbilanciamento dei dati.

Nella **competizione Kaggle**, tuttavia, la differenza tra i due modelli è più marcata, con XGBoost che ottiene uno score di 0.408 rispetto a 0.377 per Random Forest.