Weather Classification

Università degli Studi di Cagliari
Corso di Laurea in Informatica Applicata e Data Analytics
Machine Learning A.A. 2023/2024

19 Luglio 2024

Roberto A. Usai, Marco Cipollina, Gabriele Demurtas, Chiara Scalas

Abstract

Il progetto esplora l'applicazione di tecniche di machine learning per la classificazione delle condizioni meteorologiche utilizzando un dataset tabellare sintetico. Il dataset, generato artificialmente, include deliberatamente outliers e valori anomali che non rispecchiano perfettamente scenari realistici. L'obiettivo principale è sviluppare un modello predittivo in grado di classificare con precisione quattro tipi di condizioni meteorologiche: sole, nuvolosità, pioggia e neve. Il dataset contiene undici caratteristiche rilevanti, tra cui temperatura, umidità e velocità del vento.

Sono state create altre due trasformazioni del dataset originale tramite campionamento e oversampling, su cui sono state applicate varie tecniche di pre-processing. Cinque algoritmi di classificazione, di cui due custom (uno dei quali multiplo) e tre implementati tramite la libreria scikit-learn, sono stati addestrati e valutati utilizzando diverse combinazioni di dataset e tecniche pre-processing.

Dopo un'analisi approfondita dei risultati è stata creata un’ulteriore combinazione di pre-processing, in cui inizialmente sono stati gestiti i valori anomali e successivamente svolte tecniche di oversampling e standardizzazione.

Le performance dei modelli sono poi state confrontate evidenziando che il miglior risultato, con un'accuratezza pari al 97.6%, è stato ottenuto tramite un classificatore Ensemble customizzato e l'utilizzo del dataset senza outliers, ampliato e standardizzato.

Indice:

1. Introduzione

L'analisi e la previsione delle condizioni meteorologiche rappresentano una sfida significativa nel campo del machine learning, data la complessità e la variabilità intrinseca dei dati meteorologici. Prevedere accuratamente il tempo atmosferico può avere un impatto sostanziale su vari settori, come l'agricoltura, i trasporti e la gestione delle risorse naturali.

Il progetto si propone di sviluppare un modello in grado di classificare quattro tipi di condizioni meteorologiche: sole, nuvolosità, pioggia e neve. Per questo scopo, è stato utilizzato il dataset sintetico *Weather Type Classification*, che include undici caratteristiche rilevanti, tra cui temperatura, umidità e velocità del vento. Il dataset è stato generato artificialmente e contiene volutamente outliers e valori anomali per testare la robustezza dei modelli.

Gli obiettivi principali di questo progetto sono i seguenti:

- Sviluppare e confrontare diversi modelli di classificazione;
- Applicare varie tecniche di pre-processing, come la gestione dei valori anomali e la standardizzazione, per migliorare le performance dei modelli;
- Analizzare l'impatto di diverse combinazioni di tecniche di pre-processing e modelli di classificazione.

Questo studio non solo mira a identificare il miglior modello predittivo, ma anche a fornire una comprensione approfondita delle tecniche di machine learning applicate a dati meteorologici sintetici. I risultati ottenuti offrono spunti preziosi per future applicazioni e miglioramenti nell'area della previsione meteorologica tramite machine learning.

1.1 Librerie Utilizzate

Per la realizzazione di questo progetto è stato utilizzato il linguaggio di programmazione Python (ver. 3.10.4) insieme alle seguenti librerie:

Librerie	Versione
Dython	0.6.7
Imbalanced-learn	0.12.3
Matplotlib	3.5.2
Numpy	1.21.5
Pandas	1.4.4
Scikit-learn	1.1.1
Seaborn	0.13.2

Dython:
Questa libreria contiene diversi strumenti per l'analisi dei dati. Nel progetto è stata utilizzata per l'analisi della matrice di correlazione, inclusiva degli attributi di tipo categorico.

Imbalanced-learn:
Permette l'uso di funzioni per l'elaborazione dei dati, in particolare per il bilanciamento delle classi target nel dataset. Nel nostro caso, è stata utilizzata la tecnica SMOTE per il bilanciamento di tipo Oversampling.

Matplotlib:
Fornisce metodi e strumenti per la rappresentazione grafica dei dati. È stata utilizzata principalmente per la visualizzazione dei diversi aspetti analizzati nel dataset e delle prestazioni dei modelli.

Numpy:
Utilizzata per il calcolo delle operazioni sugli array e per varie operazioni aritmetiche.

Pandas:
Fornisce metodi per la gestione di DataFrame. Nel progetto è stata usata per l'estrazione dei dati dal file .csv contenente il dataset utilizzato.

Scikit-learn:
Questa libreria fornisce numerosi metodi e modelli di apprendimento automatico. È stata utilizzata per le varie funzioni di utilità e per i tre modelli di classificazione: Decision Tree, Naive Bayes e SVM.

Seaborn:
Basata su Matplotlib, questa libreria offre diverse interfacce per la visualizzazione di grafici statistici, come ad esempio i BoxPlot.

1.2 Interfaccia utente

È stata creata un'interfaccia grafica eseguibile su terminale Windows (comando *python main.py* nel path del progetto). Questa interfaccia permette di:

- Visualizzare a schermo vari tipi di analisi del dataset originale;
- Addestrare le diverse combinazioni di classificatori e pre-processing;
- Visualizzare le relative performance dei modelli addestrati;
- Confrontare le performance di più classificatori.

L'interfaccia è stata progettata per essere intuitiva e user-friendly, consentendo agli utenti di interagire facilmente con il sistema.

1.3 Codice sorgente

Il codice sorgente del progetto è interamente commentato per facilitare la comprensione e la manutenzione del software. Ogni funzione e modulo è stato accuratamente documentato per chiarire la logica implementativa e le specifiche tecniche.

Il codice è disponibile nel repository GitHub accessibile a questo link. Nel repository, è possibile trovare anche la documentazione aggiuntiva, inclusi i file README e le istruzioni per l'installazione e l'esecuzione del progetto.

2. Analisi del Dataset

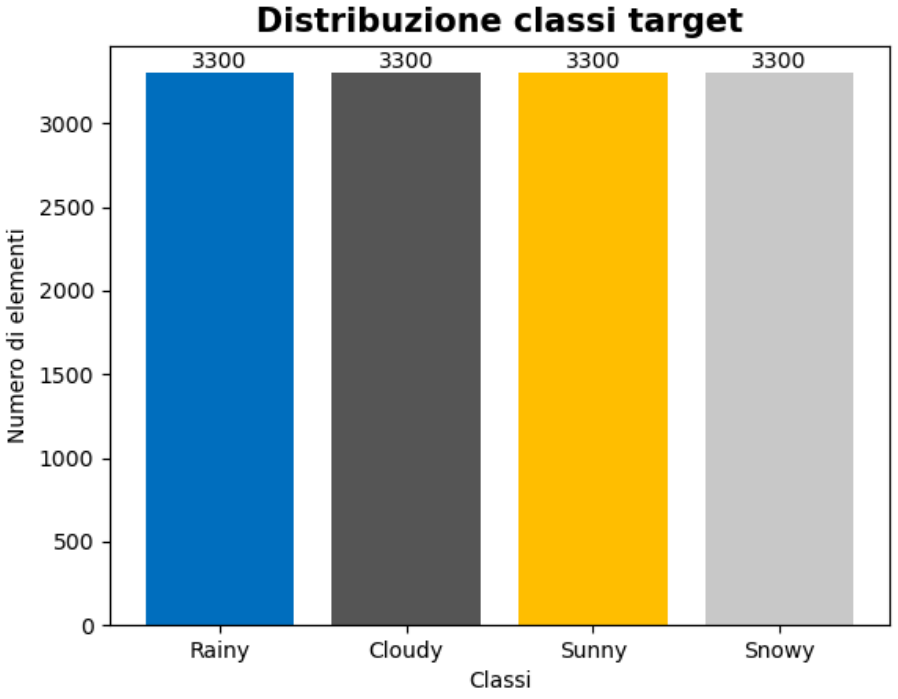
Il dataset scelto per questo progetto è di tipo record data, più precisamente, di tipo data matrix. Si presenta infatti come una tabella composta da 13200 righe (records) e 11 colonne (features), di cui 7 di tipo numerico e 4 di tipo categorico.

Gli attributi sono i seguenti e nessuno di questi contiene valori nulli:

1. *Temperature*: temperatura espressa in gradi Celsius (numerico)
2. *Humidity*: percentuale di umidità (numerico)
3. *Wind Speed*: velocità del vento in km/h (numerico)
4. *Precipitation (%)*: percentuale di precipitazioni (numerico)
5. *Cloud Cover*: descrizione della copertura delle nuvole (categorico)
6. *Atmospheric Pressure*: pressione atmosferica (numerico)
7. *UV Index*: indice UV indicante la forza dei raggi atmosferici (numerico)
8. *Season*: stagione in cui è stata effettuata la rilevazione (categorico)
9. *Visibility (km)*: visibilità espressa in km (numerico)
10. *Location*: tipo di posizione in cui è stata effettuata la rilevazione (categorico)
11. *Weather Type*: tipo di tempo atmosferico, valore da predire (categorico)

2.1 Distribuzione dei record

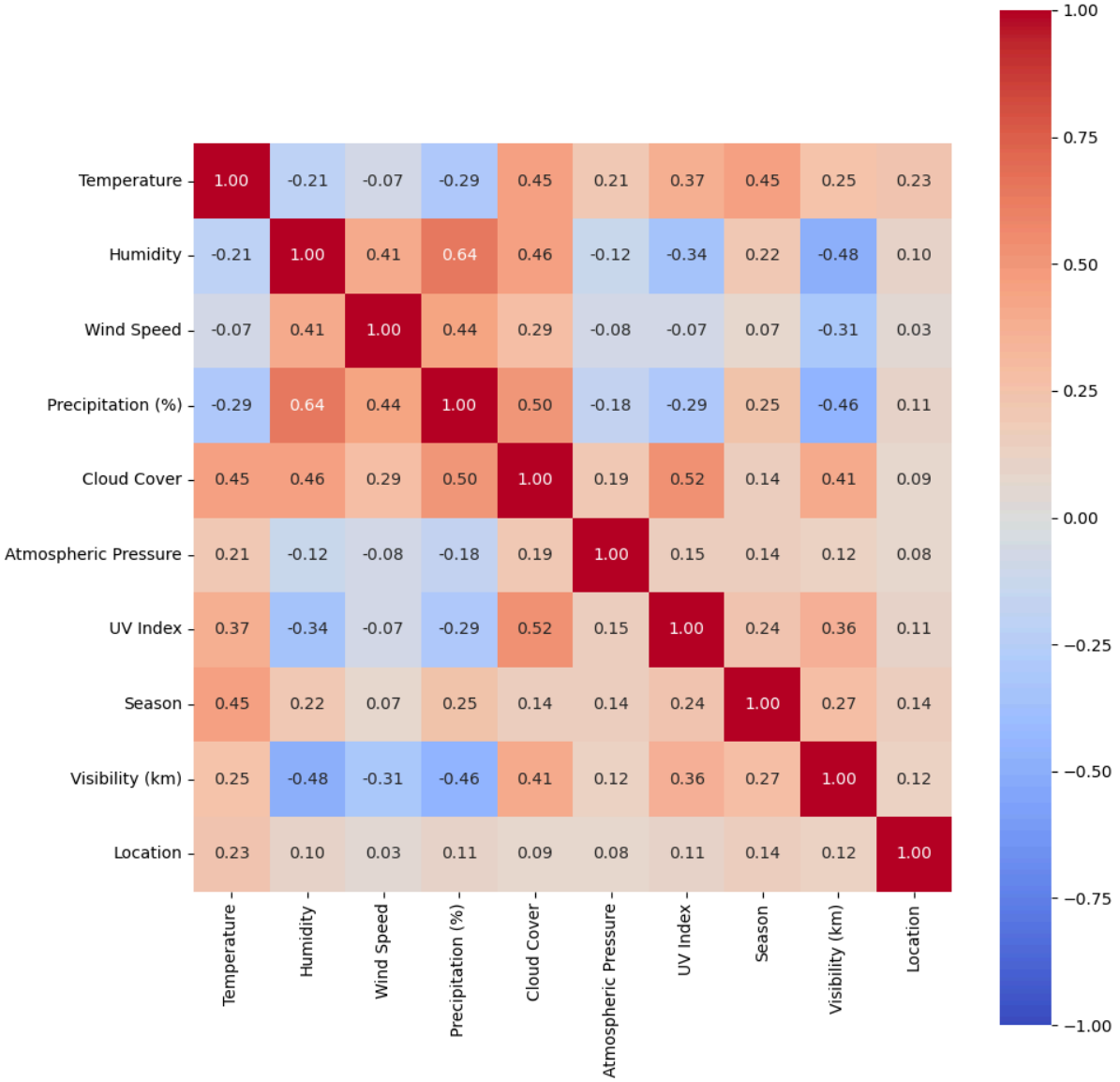
Vediamo a seguito la distribuzione dei records tra le quattro classi target:



Come si evince dal grafico, le quattro classi contengono 3300 records ciascuna, risultando perfettamente bilanciate e rendendo non necessarie operazioni di bilanciamento. Tuttavia, verranno proposte due varianti del dataset. La prima variante contiene un numero ridotto di record per classe, mentre la seconda variante include un numero aumentato di record, mantenendo comunque il bilanciamento tra le classi. Questo approccio permetterà di osservare il variare delle prestazioni al diminuire o all'aumentare del numero di campioni per classe.

2.2 Matrice di correlazione

La matrice di correlazione visualizzata di seguito mostra le correlazioni tra le diverse caratteristiche del dataset. La correlazione tra due variabili è rappresentata da un valore compreso tra -1 e 1, dove 1 indica una correlazione positiva perfetta, -1 indica una correlazione negativa (o inversa) perfetta e 0 indica nessuna correlazione.



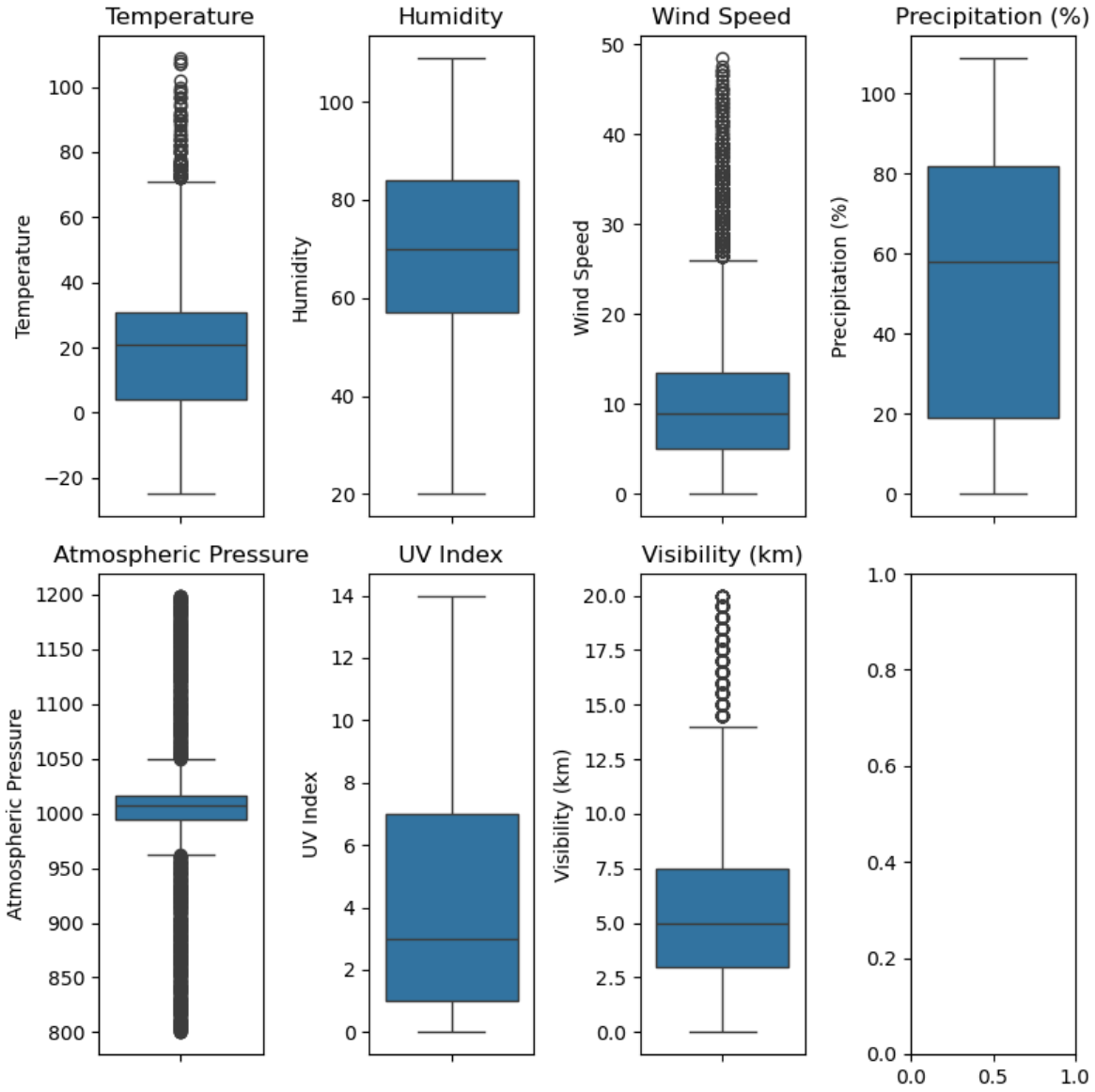
Dall'analisi della matrice di correlazione possiamo fare le seguenti osservazioni:

- Humidity e Precipitation (%): Esiste una correlazione positiva significativa (+0.64) tra l'umidità e le precipitazioni, suggerendo che un aumento dell'umidità è fortemente associato a un aumento delle precipitazioni.
- Humidity e Cloud Cover: Esiste una correlazione positiva moderata (+0.46) tra l'umidità e la copertura nuvolosa, il che indica che giornate con maggiore umidità tendono ad avere anche una maggiore copertura nuvolosa.
- Humidity e Visibility (km): Esiste una correlazione negativa significativa (-0.48) tra l'umidità e la visibilità, il che è comprensibile poiché un'umidità più alta spesso riduce la visibilità.
- Precipitation (%) e Wind Speed: Esiste una correlazione positiva moderata (+0.44) tra le precipitazioni e la velocità del vento, suggerendo che giornate con maggiori precipitazioni tendono ad avere venti più forti.
- Precipitation (%) e Cloud Cover: Esiste una correlazione positiva significativa (+0.50) tra le precipitazioni e la copertura nuvolosa, il che indica che una maggiore copertura nuvolosa è associata a un aumento delle precipitazioni.
- Precipitation (%) e Visibility (km): Esiste una correlazione negativa moderata (-0.46) tra le precipitazioni e la visibilità, suggerendo che maggiori precipitazioni riducono la visibilità.
- Cloud Cover e UV Index: Esiste una correlazione positiva moderata (+0.52) tra la copertura nuvolosa e l'indice UV, il che è interessante poiché ci si potrebbe aspettare una correlazione negativa; questo potrebbe indicare che la copertura nuvolosa rilevata non riduce significativamente l'esposizione ai raggi UV.

2.3 BoxPlot

I boxplot sono strumenti grafici utilizzati in statistica per visualizzare la distribuzione di un insieme di dati. Rappresentano visivamente la mediana, il primo e il terzo quartile, e le estensioni dei dati che indicano la variabilità fuori dai quartili. Includono anche eventuali valori anomali.

I boxplot riportati di seguito mostrano la distribuzione dei valori delle sette variabili numeriche presenti nel dataset. Analizziamo ogni variabile individualmente, evidenziando eventuali valori anomali o surreali.



- Temperature: La temperatura mostra una distribuzione ampia con valori compresi tra −25° C e 109° C. La presenza di outlier è evidente nella parte superiore della scala, dove alcune temperature sembrano irrealisticamente alte.
- Humidity: L'umidità dovrebbe essere espressa come percentuale e quindi compresa tra 0% e 100%. Tuttavia, nel boxplot sono presenti valori che superano il 100%, indicando chiaramente dati surreali.
- Wind Speed: La velocità del vento presenta un'ampia gamma di valori con diversi outlier nella parte superiore. La distribuzione sembra essere asimmetrica, con la maggior parte dei valori concentrati tra 0 e 25 km/s.
- Precipitation (%): Anche le precipitazioni sono rappresentate in percentuale, eppure i valori superano il 100%, indicando nuovamente dati surreali.
- Atmospheric Pressure: La pressione atmosferica mostra valori estremamente elevati, con molti outlier che superano i 1050 hPa, un valore irrealisticamente alto.

- UV Index: L'indice UV varia tra 0 e circa 12, con alcuni outlier. La distribuzione sembra ragionevole e non presenta anomalie evidenti.
- Visibility (km): La visibilità mostra una gamma di valori compresi tra 0 e circa 20 km, con diversi outlier. La distribuzione è relativamente ampia ma non presenta valori surreali evidenti.

La presenza di valori surreali in vari attributi numerici (come umidità e precipitazioni superiori al 100%, temperature eccessivamente alte e pressione atmosferica molto al di sopra dei 1050 hPa) indica che il dataset necessita di una pulizia accurata. Questi errori derivano da imprecisioni nella generazione dei dati sintetici. Durante il pre-processing, sarà cruciale identificare e correggere questi valori per migliorare l'accuratezza dei modelli predittivi.

3. Metriche di valutazione

Per valutare le performance dei modelli abbiamo utilizzato diverse metriche. La metrica principale scelta è stata l'accuratezza, per diverse ragioni:

- Bilanciamento del Dataset: Il dataset utilizzato è perfettamente bilanciato, con un numero uguale di campioni per ciascuna classe. In un contesto di dati bilanciati, l'accuratezza è una metrica affidabile che riflette bene le performance dei modelli.
- Task Meteorologico: Il task di classificazione delle condizioni meteorologiche non determina specifici rischi in caso di erronea classificazione, rendendo l'accuratezza una scelta appropriata. Essa fornisce una misura chiara e intuitiva della proporzione di previsioni corrette effettuate dai modelli.

Oltre all'accuratezza, l'interfaccia utente sviluppata permette comunque di ottenere report dettagliati utilizzando altre metriche importanti, offrendo una visione più completa delle performance dei modelli:

- Precision: Misura la proporzione di veri positivi tra tutti i campioni che sono stati classificati come positivi.
- Recall: Misura la proporzione di veri positivi tra tutti i campioni che sono realmente positivi.
- F1 Score: È la media armonica della precision e della recall. È particolarmente utile in scenari con classi sbilanciate, e anche se in questo caso il dataset è bilanciato, offre comunque un'indicazione della qualità complessiva delle previsioni, bilanciando tra precision e recall.
- Matrice di Confusione: Fornisce una rappresentazione dettagliata delle performance del modello, mostrando il numero di veri positivi, falsi positivi, veri negativi e falsi negativi. La matrice di confusione permette di identificare specifici errori di classificazione e di comprendere meglio come i modelli si comportano per ciascuna classe.

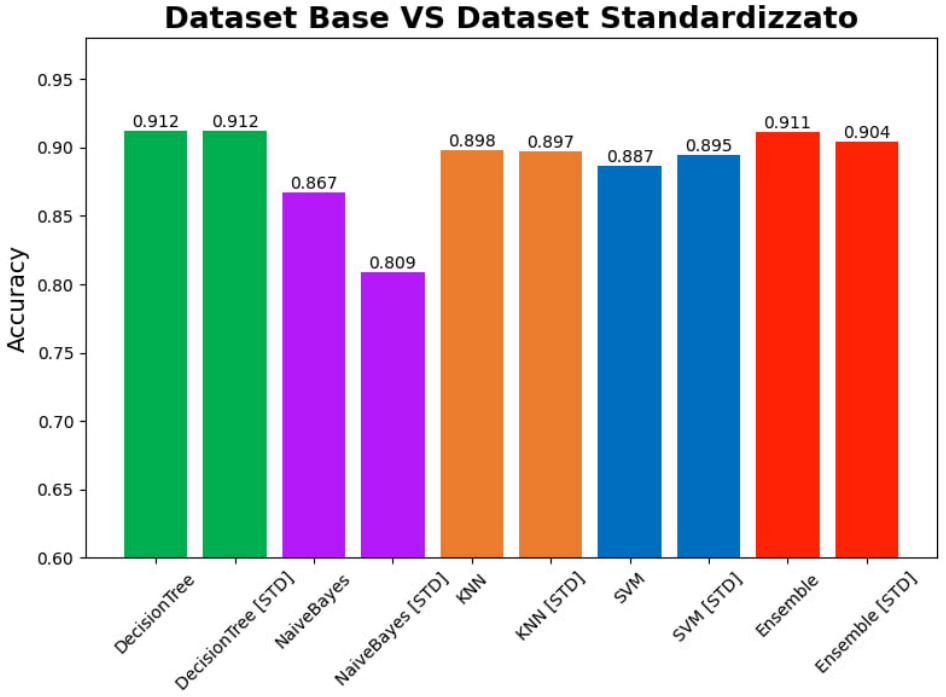
4. Pre-Processing

Le tecniche di pre-processing sono fondamentali per preparare i dati in modo che i modelli possano apprendere efficacemente. Per garantire che i dati siano nella forma ottimale sono state applicate le seguenti tecniche:

- Standardizzazione dei Dati: Uniformare le scale delle variabili numeriche per migliorare la performance dei modelli.
- Campionamento: Creare un dataset ridotto, mantenendo il bilanciamento, per testare le prestazioni dei modelli con meno dati.
- Oversampling: Aumentare il numero di campioni per ciascuna classe, mantenendo il bilanciamento, per osservare l'impatto su modelli con dataset più corposi.
- Selezione delle Feature: Ridurre il numero di attributi per diminuire la complessità dei modelli e il carico computazionale.
- Rimozione degli Outlier e dei Valori Anomali: Pulire il dataset eliminando valori anomali e surreali, garantendo dati più accurati e affidabili.

4.1 Standardizzazione

Per effettuare lo scaling dei dati del dataset, è stata utilizzata la classe StandardScaler della libreria scikit-learn. La standardizzazione consiste nel centrare i dati attorno a una media zero e scalare in modo che abbiano una deviazione standard unitaria. Questo processo è stato applicato esclusivamente agli attributi di tipo numerico.



Il grafico sopra riportato mostra le performance ottenute dai 5 modelli, prima sul dataset base, privo di pre-processing, e poi sul dataset *STD* (standardizzato).

- Decision Tree: le performance sono rimaste invariate, mantenendo un'accuratezza del 91.2%.
- Naive Bayes: ha subito un peggioramento delle prestazioni, passando da un'accuratezza dell'86.7% a una dell'80.9%.
- KNN: ha mostrato un leggerissimo calo di performance, con un'accuratezza che è scesa dal 89.8% al 89.7%.
- SVM: è l'unico che ha beneficiato della standardizzazione, migliorando non solo l'accuratezza, dal 88.7% al 89.5%, ma anche la velocità d'addestramento.
- Ensemble: ha mostrato una leggera diminuzione delle prestazioni, passando da un'accuratezza del 91.1% al 90.4%.

Questi risultati suggeriscono che la standardizzazione ha un impatto variabile sui diversi modelli. In particolare, modelli come SVM traggono beneficio dalla standardizzazione a causa della loro sensibilità alle scale dei dati, mentre altri modelli come il Decision Tree non mostrano miglioramenti significativi. È quindi cruciale considerare la natura del modello e delle variabili del dataset quando si applicano tecniche di pre-processing.

4.2 Variazione del numero di campioni

L'oversampling e l'undersampling sono tecniche utilizzate principalmente per affrontare il problema dello sbilanciamento dei dati nelle classi del dataset. I dataset sbilanciati potrebbero portare gli algoritmi di apprendimento automatico a specializzarsi sulle classi maggioritarie, trascurando le classi minoritarie.

Nel nostro caso, disponendo di un dataset perfettamente bilanciato, queste due tecniche sono state utilizzate con l'obiettivo di ampliare e ridurre il dataset originale per testare come il variare del numero di campioni influisce sulle prestazioni dei modelli.

4.2.1 Undersampling

Per ridurre il numero di campioni del dataset originale, mantenendo solo metà dei record, è stato applicato il campionamento stratificato. Questo metodo consente di mantenere la rappresentatività delle classi nel dataset ridotto, fondamentale per evitare distorsioni durante le fasi di addestramento e valutazione dei modelli di apprendimento.

Questa versione ridotta è stata utilizzata per verificare le prestazioni dei modelli di apprendimento, sia in combinazione con altre tecniche di pre-processing sia senza ulteriori trasformazioni.

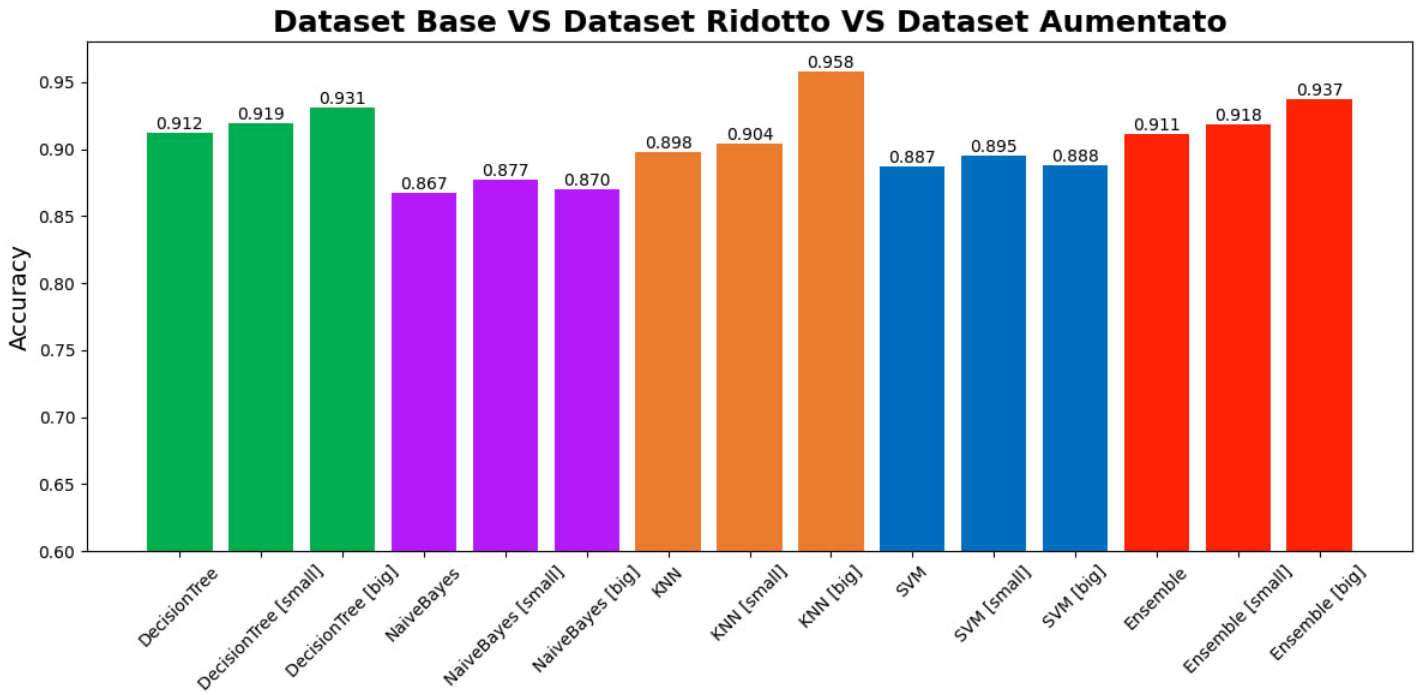
I risultati delle performance dei modelli addestrati sul dataset *small* sono stati confrontati con quelli ottenuti utilizzando il dataset originale. Questa analisi ha permesso di valutare l'efficacia dei modelli in condizioni di dati limitati, fornendo indicazioni utili per scenari reali dove la raccolta di dati può essere costosa o difficoltosa.

4.2.2 Oversampling

Per raddoppiare il numero di record è stata utilizzata la tecnica SMOTE (Synthetic Minority Over-sampling Technique). Questa seleziona i *k* più prossimi a ogni campione e genera nuove istanze lungo le linee che collegano il campione originale con i suoi vicini.

Il dataset *big*, ottenuto tramite oversampling, è stato poi utilizzato per verificare le prestazioni dei modelli di apprendimento. Sono state valutate le performance dei modelli sia in combinazione con altre tecniche di pre-processing sia senza ulteriori trasformazioni.

Questa analisi ha fornito indicazioni utili su come la disponibilità di un numero maggiore di campioni possa influire sull'accuratezza e sulla robustezza dei modelli di apprendimento, offrendo spunti preziosi per applicazioni reali dove l'espansione del dataset può essere una strategia vantaggiosa.



Confrontando le prestazioni dei singoli modelli al variare del dataset, possiamo osservare le seguenti tendenze:

- Decision Tree: mostra un leggero miglioramento con il dataset ampliato, raggiungendo un'accuratezza del 93.1%. Le prestazioni rimangono sostanzialmente invariate tra il dataset base e il dataset ridotto.
- Naive Bayes: mostra un leggero miglioramento con il dataset ridotto, raggiungendo un'accuratezza del 87.7%. Questo potrebbe essere dovuto al fatto che Naive Bayes assume indipendenza tra le caratteristiche. Un dataset ridotto potrebbe ridurre la varianza e rendere queste assunzioni più vicine alla realtà, migliorando le prestazioni del modello.
- KNN: ottiene le migliori performance con il dataset ampliato, raggiungendo un'accuratezza del 95.8%. Questo rappresenta un miglioramento significativo rispetto al dataset base e al dataset ridotto, che mostrano rispettivamente un'accuratezza di 89.8% e 90.4%. L'aumento del numero di campioni aiuta il KNN a trovare vicini più rilevanti, migliorando così la sua capacità di classificazione.
- SVM: ottiene prestazioni migliori con il dataset ridotto, raggiungendo un'accuratezza del 89.5%, leggermente superiore al dataset base. Questo potrebbe essere dovuto al fatto che l'SVM con un dataset ridotto può favorire una migliore generalizzazione, riducendo il rischio di overfitting.
- Ensemble: mostra un miglioramento con il dataset ampliato, raggiungendo un'accuratezza del 93.7%. Le performance con il dataset ridotto sono leggermente superiori rispetto al dataset base, con un'accuratezza del 91.8% rispetto al 91.1%.

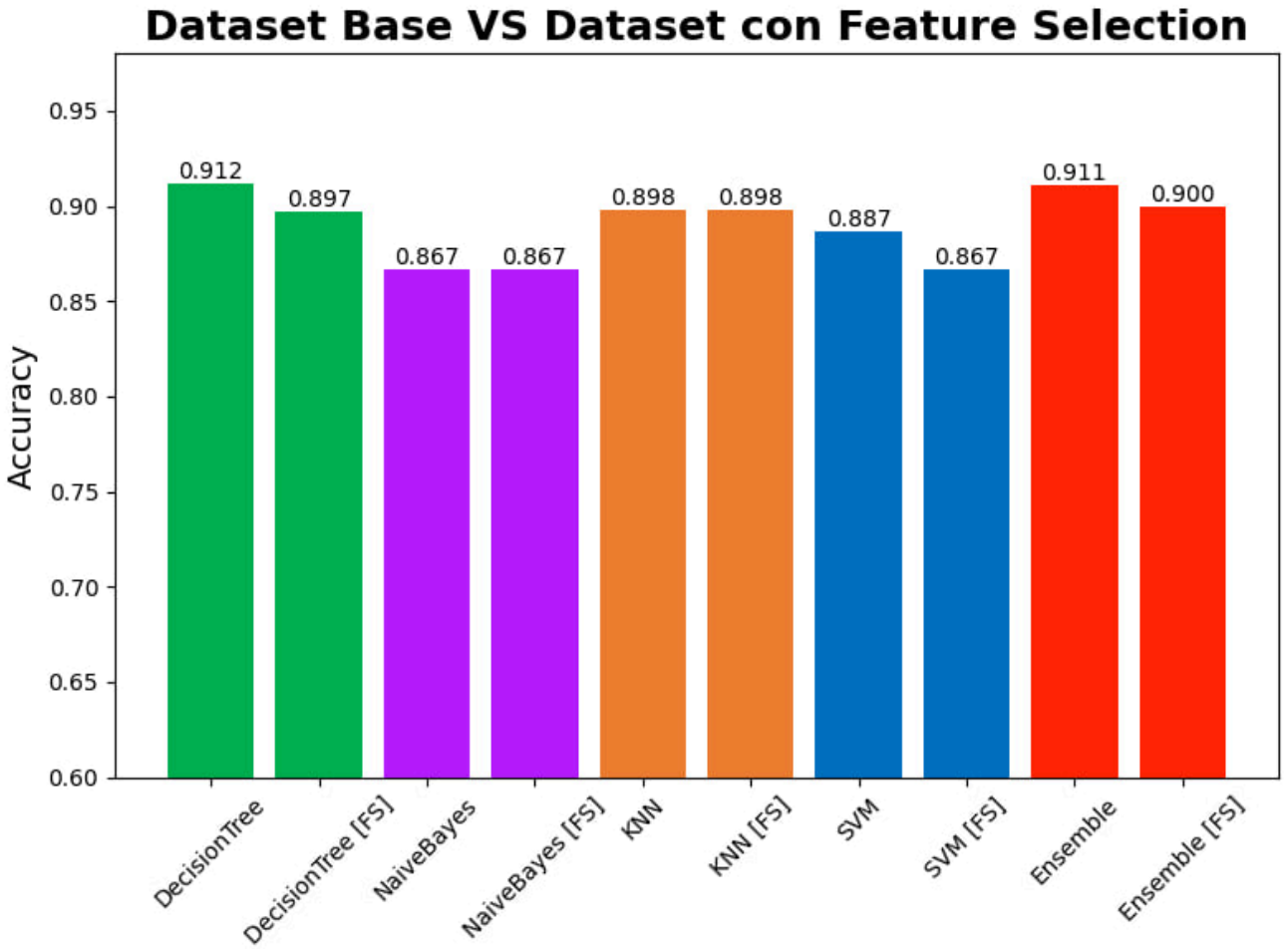
Queste osservazioni suggeriscono che il dataset ampliato migliora le performance dei modelli Decision Tree, KNN ed Ensemble, mentre i modelli Naive Bayes e SVM beneficiano maggiormente del dataset ridotto. La scelta del dataset ottimale dipende quindi dal modello specifico e dalle caratteristiche dei dati. Il Naive Bayes e l'SVM sembrano beneficiare di dataset ridotti probabilmente a causa della riduzione della varianza e del rischio di overfitting, mentre gli altri modelli traggono vantaggio dal maggior numero di campioni per migliorare la loro capacità di generalizzazione e robustezza.

4.3 Feature Selection

La selezione delle feature è un processo fondamentale nel machine learning che mira a ridurre la dimensionalità dei dati eliminando attributi ridondanti o non informativi. Questo processo può migliorare le performance dei modelli, ridurre il carico computazionale e prevenire il problema dell'overfitting. Tuttavia, nel nostro caso, il dataset contiene solo 11 attributi, il che non rappresenta un onere computazionale significativo. Nonostante ciò, abbiamo deciso di testare il trade-off tra la riduzione delle dimensionalità e le performance ottenute.

Per determinare quali feature rimuovere ci si è basati sull'analisi della matrice di correlazione (descritta nella relativa sezione). Basandosi su queste correlazioni, è stato deciso di rimuovere le feature *Humidity* e *Cloud Cover*, fortemente correlate con altre presenti nel dataset, in quanto potrebbero risultare ridondanti.

Il seguente grafico mostra il confronto tra le performance ottenute dai modelli sul dataset originale e sul dataset *FS* (Feature Selection):



- Decision Tree: mostra un leggero calo delle performance dopo la rimozione delle feature, con un'accuratezza che scende dal 91.2% al 89.7%.

- Naive Bayes: le performance rimangono invariate, con un'accuratezza del 86.7% sia prima che dopo la selezione delle feature.
- KNN: mostra che la rimozione delle feature non ha avuto un impatto significativo.
- SVM: subisce un calo delle performance, con un'accuratezza che scende dal 88.7% al 86.7%.
- Ensemble: mostra una diminuzione delle performance, con un'accuratezza che scende dal 91.1% al 90.0%.

Questi risultati indicano che la selezione delle feature non ha apportato benefici in termini di rapporto performance/costo computazionale, peggiorando leggermente le performance. Ciò potrebbe essere dovuto al fatto che, con un numero già ridotto di feature, la rimozione di attributi informativi può ridurre la capacità dei modelli di apprendere correttamente.

4.4 Rimozione dei valori anomali e degli outlier

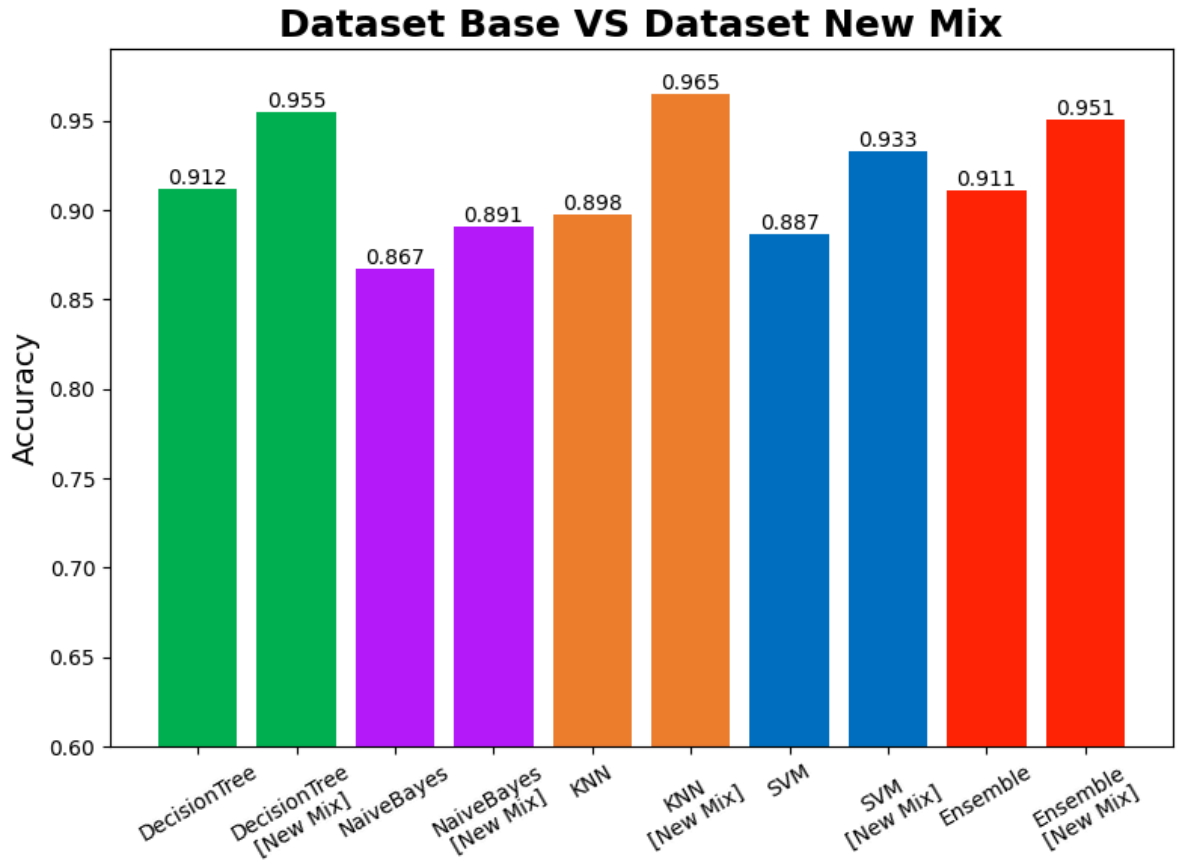
La rimozione degli outlier e dei valori anomali è un passaggio cruciale nel pre-processing dei dati, in quanto permette di migliorare la qualità del dataset e la robustezza dei modelli di apprendimento. Gli outlier possono distorcere i risultati dei modelli, influenzando negativamente le prestazioni e portando a conclusioni errate. Eliminando i valori anomali, si garantisce che i dati utilizzati per l'addestramento siano più rappresentativi e affidabili.

Nel nostro dataset, sono state applicate le seguenti rimozioni di valori anomali, basate sull'analisi del boxplot svolta nella relativa sezione:

- Temperature: sono stati rimossi i valori superiori a 50°C, poiché anomali e non rappresentativi di condizioni atmosferiche realistiche. I valori originali risultano inclusi in un range tra -25° C e 109° C.
- Humidity: sono stati eliminati i valori che superano il 100%, in quanto l'umidità è espressa come percentuale e quindi compresa tra 0% e 100%.
- Precipitation (%): sono stati rimossi i valori superiori al 100% poiché indicativi di errori di inserimento o dati surreali.
- Atmospheric Pressure: sono stati eliminati i valori superiori a 1050 hPa, considerati surreali nell'atmosfera terrestre.

Questi interventi hanno permesso di ottenere un dataset più pulito e realistico, riducendo l'influenza di valori estremi che potrebbero distorcere i risultati.

Visti gli ottimi risultati trovati precedentemente con la standardizzazione e l'ampliamento del dataset, è stata creata direttamente la migliore combinazione possibile tra le tecniche di pre-processing analizzate, combinando la rimozione dei valori anomali e degli outlier alle due sopracitate.



I risultati sopra mostrati indicano che questa combinazione delle tecniche di pre-processing ha portato a un miglioramento generale delle performance dei modelli, confermando l'importanza di un'accurata preparazione dei dati per ottenere risultati ottimali nei modelli di machine learning.

5. Classificatori

È stato scelto di implementare diversi classificatori per sfruttare i loro differenti approcci, ognuno con caratteristiche uniche che possono essere più o meno efficaci a seconda del tipo di dati.

Per garantire la coerenza e la facilità d'uso tra i vari classificatori, sono stati definiti alcuni parametri comuni che saranno utilizzati in tutte le implementazioni:

- dataset: tupla contenente i dati di training e di test già splittati.
- votazione: tipologia di votazione da utilizzare:
  - none: il modello non fa parte di un ensemble (default);
  - hard: il modello fa parte di un ensemble e restituisce le sue predizioni;
  - soft: il modello fa parte di un ensemble e restituisce le probabilità delle sue predizioni.
- show_results: permette di scegliere se stampare i risultati del classificatore.

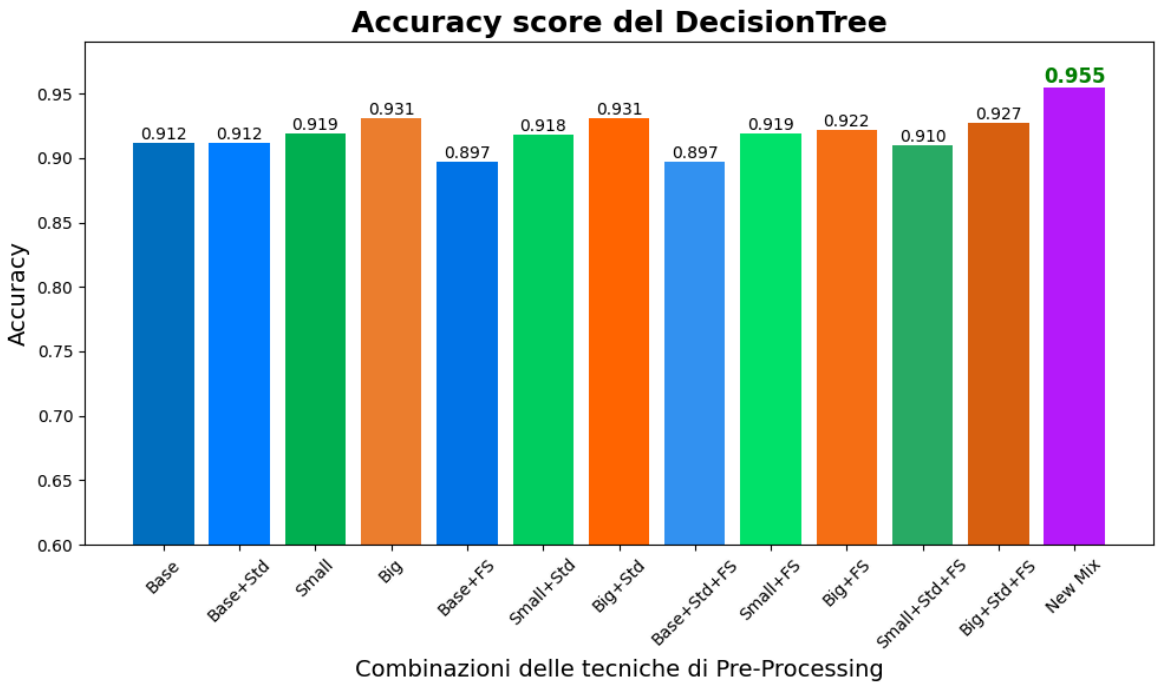
Questi parametri permettono di configurare i classificatori e di integrare i loro risultati nell'ensemble, garantendo al contempo una flessibilità sufficiente per eseguire diverse modalità di votazione o visualizzare i report finali.

5.1 Decision Tree

È stato selezionato per avere un approccio di apprendimento basato su regole. Esso suddivide i dati in base a caratteristiche e soglie specifiche per prendere decisioni. Ogni nodo rappresenta una caratteristica e ogni ramo una decisione basata su quella caratteristica. Le foglie dell'albero rappresentano le classificazioni finali, ovvero le etichette delle classi.

Gli iperparametri principali su cui verte il DecisionTree sono il criterio di split (Entropia di default) e la profondità massima dell'albero (7 di default), entrambi scelti opportunamente tramite tuning.

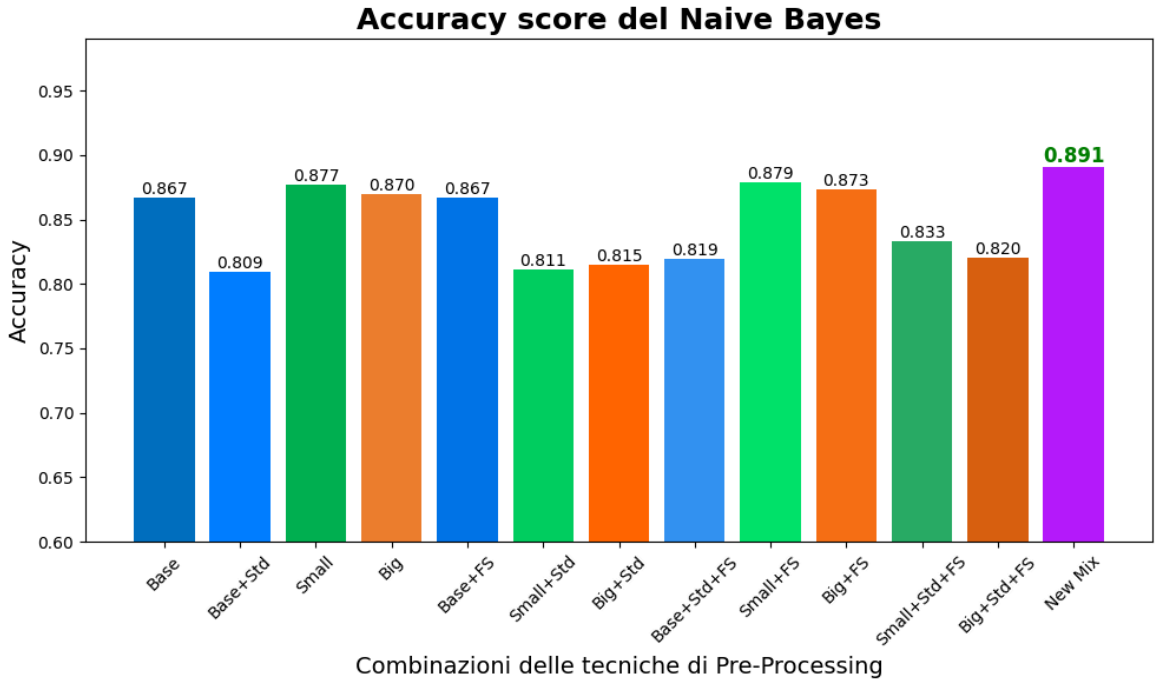
New Mix è il dataset in cui vengono raggiunte le performance più alte. Questo può essere attribuito alla maggiore quantità di campioni, che consente al modello di identificare pattern più definiti, e soprattutto alla pulizia dei dati, che riduce il rumore e le anomalie. È possibile osservare i risultati nel grafico sotto riportato:



5.2 Naive Bayes

L'utilizzo del Naive Bayes è stato scelto per includere un approccio probabilistico nel nostro set di classificatori. Questo si basa sul teorema di Bayes, assumendo che le caratteristiche siano indipendenti tra loro (assunzione "naive"). Il Naive Bayes è stato testato vista la sua semplicità di implementazione e velocità computazionale, nonostante l'incoerenza con le caratteristiche dipendenti del dataset.

L'iperparametro su cui si basa il Naive Bayes è *var_smoothing* (0.001 di default), che permette di determinare la varianza da sommare ad ogni istanza.



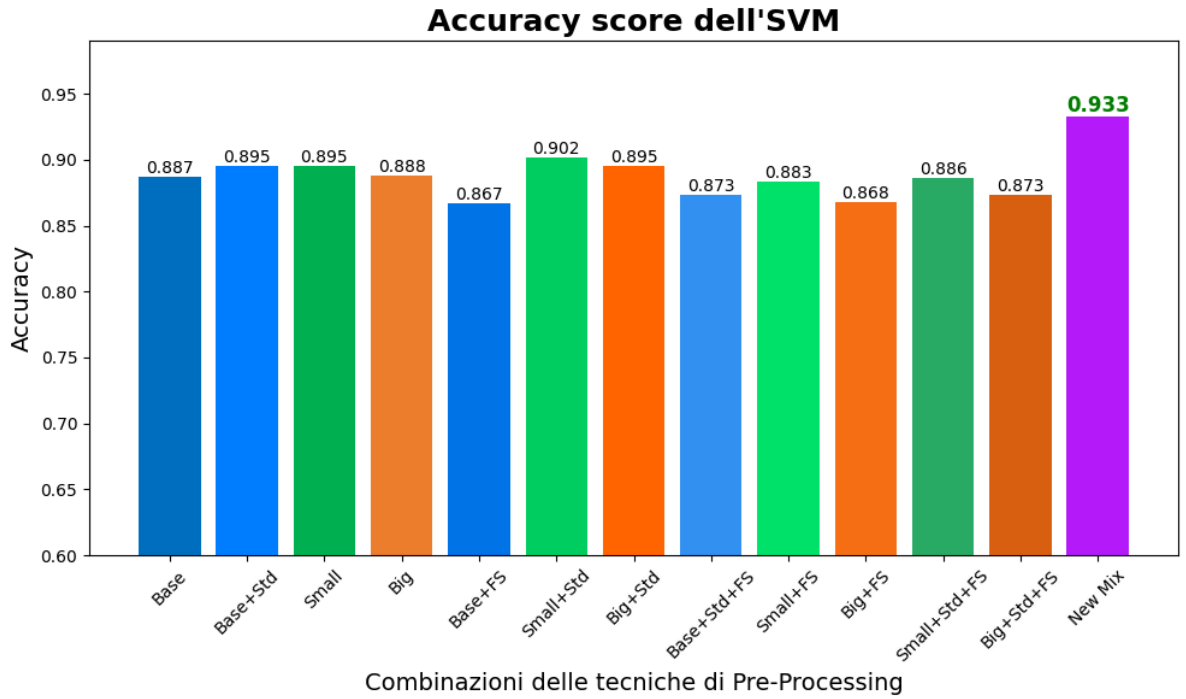
La standardizzazione del dataset ha generalmente ridotto significativamente l'accuratezza. Questo può essere dovuto all'incoerenza della distribuzione dei dati rispetto all'assunzione di indipendenza del Naive Bayes. L'unica tecnica di pre-processing rilevata utile è stata la rimozione degli outliers.

Coerentemente col fatto che essendo un task di classificazione su attributi meteorologici, che perciò non risultano essere indipendenti tra di loro, è risultato il modello meno performante, con una accuratezza finale ben più bassa rispetto agli altri classificatori adoperati. Tuttavia, la sua semplicità e velocità computazionale lo rendono un utile punto di confronto.

5.3 Support Vector Machine

La Support Vector Machine è rientrata nella nostra scelta per via del suo approccio geometrico basato su margini, essa cerca di trovare un iperpiano che separa le classi con il margine più ampio possibile.

Gli iperparametri che determinano le performance del modello sono il tipo di kernel (linear di default) e il parametro di regolarizzazione C (1 di default), ambedue scelti coerentemente tramite tuning.



Il grafico sopra riportato mostra che l'SVM ha raggiunto performance più alte con la standardizzazione e la rimozione degli outlier. In particolare, la rimozione degli outlier ha avuto un impatto positivo, migliorando l'accuratezza complessiva del modello. Questi risultati suggeriscono che, per ottenere il massimo dalle capacità dell'SVM, è essenziale applicare tecniche di pre-processing che puliscano e bilancino il dataset

5.4 Custom K-Nearest Neighbor

Abbiamo scelto di utilizzare l'algoritmo K-Nearest Neighbor (KNN) per sfruttare il suo approccio geometrico basato sulla distanza. L'implementazione è stata realizzata tramite la classe custom denominata *CustomKNN*, che eredita dalla classe *BaseEstimator* funzionalità utili per la compatibilità con scikit-learn.

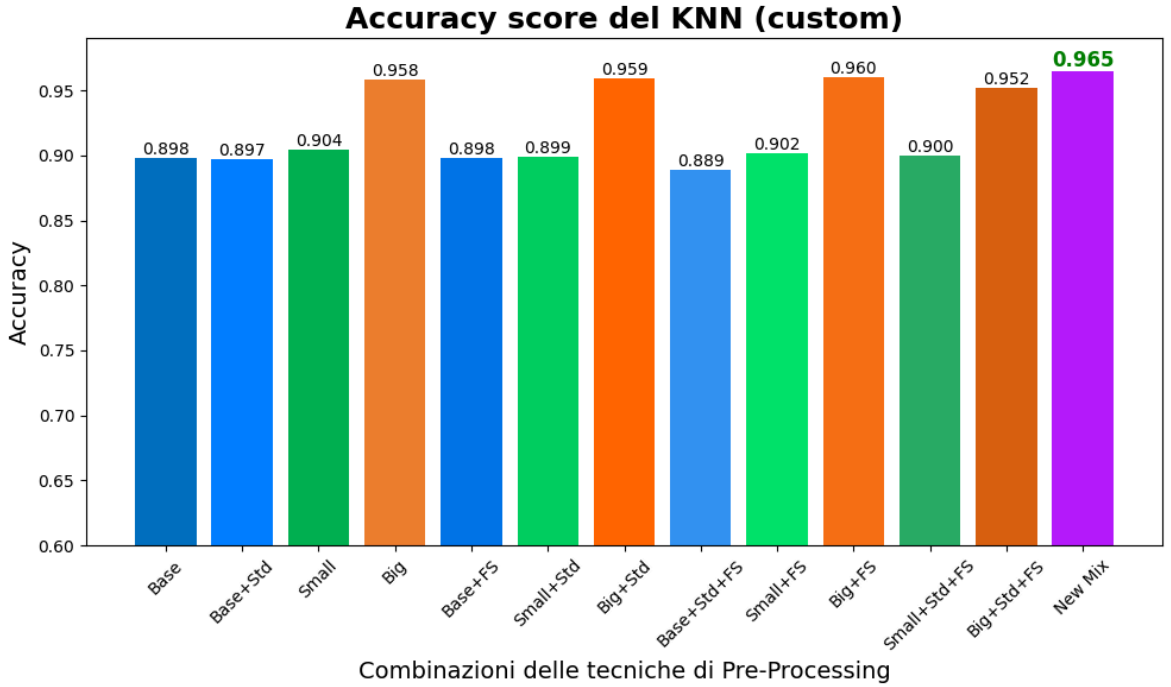
La classe *CustomKNN* è costruita con due parametri principali:

1. *k*: numero di vicini da analizzare per la classificazione di una nuova istanza.
2. *misura_distanza*: stringa che rappresenta la misura della distanza da adottare, può essere "Euclidea", in quanto misura la distanza diretta tra i punti nello spazio multidimensionale, o "Manhattan", scelta per la sua maggiore robustezza rispetto ai valori anomali presenti nel dataset.

Questi rappresentano gli iperparametri del KNN, impostati successivamente di default rispettivamente con 7 e "Manhattan".

CustomKNN fornisce i seguenti metodi:

- *fit*(): Addestra il modello memorizzando il training set e calcolando il numero di classi presenti.
- *predict*(): Effettua previsioni calcolando la distanza tra il test set e il training set, selezionando i *k* vicini più prossimi e pesando i loro contributi. Supporta il tipo di predizione hard e soft.
- *score*(): Calcola l'accuratezza delle previsioni utilizzando accuracy_score().



Il grafico sopra riportato mostra le performance del modello addestrato su diverse combinazioni di pre-processing del dataset, evidenziando come le diverse tecniche influenzino l'accuratezza del modello.

Il modello KNN customizzato ha mostrato in generale performance migliori con il dataset ampliato. Questo è dovuto al fatto che un maggior numero di campioni permette al KNN di trovare vicini più rappresentativi per ciascuna istanza da classificare, migliorando così l'accuratezza complessiva. La rimozione degli outlier, combinata con l'oversampling e la standardizzazione, ha ulteriormente migliorato le performance. Queste osservazioni suggeriscono che, per ottenere il massimo dalle capacità del KNN, è cruciale utilizzare un dataset ampio e pulito.

5.5 Custom Ensemble

Il classificatore *CustomEnsemble* rappresenta un approccio efficace per combinare le previsioni di diversi modelli e per sfruttare i punti di forza dei vari approcci di classificazione. Il classificatore composto permette di migliorare la robustezza e l'accuratezza complessiva del modello, riducendo l'impatto degli errori commessi dai singoli classificatori.

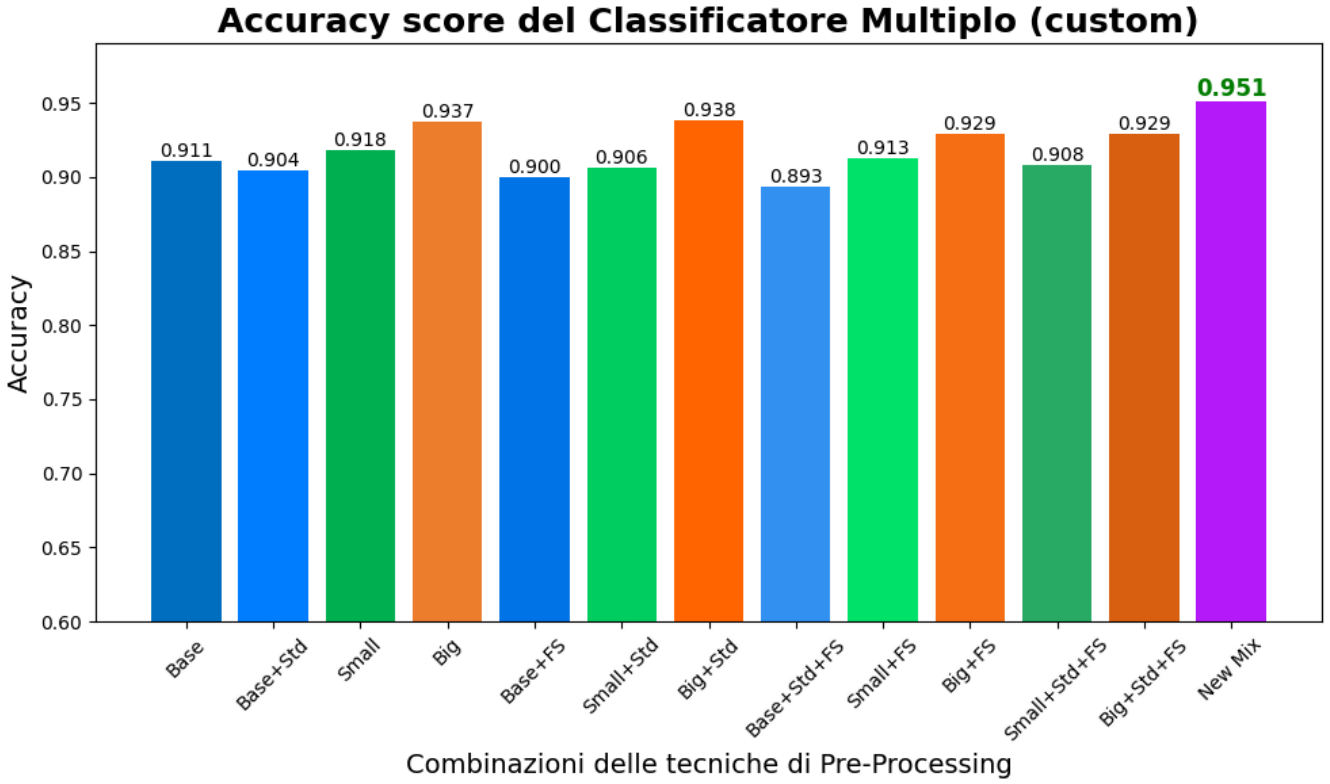
La funzione *ensemble_main*() permette di ottenere diverse tipologie di votazione a seconda dei parametri presi in input:

- Hard Uniforme: somma i voti dei singoli classificatori e determina la classificazione maggioritaria.
- Hard Pesata: somma i voti dei singoli modelli, li moltiplica per i rispettivi pesi e determina la classe maggioritaria per ogni istanza.
- Soft Uniforme: somma le probabilità delle singole classi di ogni classificatore e determina la classe più votata.
- Soft Pesata: moltiplica per i rispettivi pesi le probabilità delle singole classi di ogni classificatore, le somma e determina la classe maggioritaria.

L'uso della votazione pesata, basata sull'accuratezza (moltiplicata per 100 ed elevandola al quadrato, in modo da enfatizzare i divari tra i contributi) dei singoli modelli, garantisce che i contributi più affidabili abbiano un peso maggiore nella decisione finale.

Infine, l'accuratezza del modello viene calcolata utilizzando la funzione *accuracy_score()*, confrontando le predizioni finali con le etichette del test set.

Il seguente grafico mostra le performance del modello Custom Ensemble addestrato su diverse combinazioni di pre-processing del dataset.



II modello ha mostrato in generale performance migliori con il dataset ampliato. Questo è dovuto al fatto che un maggior numero di campioni permettono al modello di combinare meglio le previsioni dei singoli classificatori, migliorando così l'accuratezza complessiva. L'utilizzo della combinazione *New Mix* ha ulteriormente migliorato le performance, evidenziando l'importanza di un pre-processing accurato dei dati. Queste osservazioni suggeriscono che, per ottenere il massimo dalle capacità del modello Custom Ensemble, è cruciale utilizzare un dataset ampio e pulito.

In conclusione, l'implementazione del CustomEnsemble ha dimostrato di essere una strategia vincente per migliorare le performance dei modelli di machine learning, rendendo i risultati più stabili e accurati.

6. Tuning degli iperparametri

Ogni modello dispone di una sua funzione per il tuning degli iperparametri. Questa non compare nei menu consultabili all'interno del programma, ma è facilmente eseguibile lanciando singolarmente i file dei modelli. Per esempio, supponendo di voler eseguire il tuning degli iperparametri del DecisionTree, è possibile eseguire la funzione col comando *python DecisionTree.py*.

I modelli menzionati in precedenza sono tutti preimpostati con le loro configurazioni ottimali ottenute sul dataset privo di pre-processing.

Per assicurare che i valori ottimali siano robusti e che non dipendano da una specifica suddivisione dei dati, tutti i modelli sono stati valutati seguendo l'approccio della cross validation.

6.1 Naive Bayes

L'unico iperparametro del Naive Bayes è il valore di smoothing, ovvero il valore usato per aggiungere una piccola quantità di varianza di ciascuna feature per garantire la stabilità dei calcoli. Serve per migliorare le prestazioni del modello evitando divisioni per valori molto piccoli durante il calcolo delle probabilità, e viene proposto con un range di valori molto differenti, da molto piccoli a molto grandi (da 1e-10 a 1e-1). Un ampio range di valori di smoothing permette di trovare il valore ottimale che funziona bene con le diverse scale di varianza presenti nel dataset. Segue l'output dell'esecuzione del tuning con tutte le combinazioni possibili:

```
Smoothing: 1e-10 - Accuratezza: 0.78343
Smoothing: 1e-09 - Accuratezza: 0.78400
Smoothing: 1e-08 - Accuratezza: 0.78911
Smoothing: 1e-07 - Accuratezza: 0.79801
Smoothing: 1e-06 - Accuratezza: 0.81089
Smoothing: 1e-05 - Accuratezza: 0.82964
Smoothing: 0.0001 - Accuratezza: 0.85133
Smoothing: 0.001 - Accuratezza: 0.85748
Smoothing: 0.01 - Accuratezza: 0.84460
Smoothing: 0.1 - Accuratezza: 0.81534
Smoothing: 1 - Accuratezza: 0.80824
Smoothing: 10 - Accuratezza: 0.76989
Smoothing: 100 - Accuratezza: 0.52614

Miglior Accuratezza: 0.85748 (Usando smoothing: 0.001)
```

6.2 Custom KNN

Gli iperparametri fondamentali di un KNN sono il valore di *k*, che determina il numero di vicini da considerare, e la misura della distanza utilizzata per calcolare la somiglianza tra i punti. Per effettuare il tuning è stato proposto un range di k che va da 1 a 10, mentre per le distanze sono state proposte la distanza Euclidea e la distanza di Manhattan.

Segue l'output dell'esecuzione del tuning con tutte le combinazioni possibili:

```
Distanza: "Euclidea", k = 1 - Accuratezza: 0.88333
Distanza: "Euclidea", k = 2 - Accuratezza: 0.88333
Distanza: "Euclidea", k = 3 - Accuratezza: 0.88504
Distanza: "Euclidea", k = 4 - Accuratezza: 0.88542
Distanza: "Euclidea", k = 5 - Accuratezza: 0.88589
Distanza: "Euclidea", k = 6 - Accuratezza: 0.88873
Distanza: "Euclidea", k = 7 - Accuratezza: 0.88807
Distanza: "Euclidea", k = 8 - Accuratezza: 0.88788
Distanza: "Euclidea", k = 9 - Accuratezza: 0.88864
Distanza: "Euclidea", k = 10 - Accuratezza: 0.88854


Distanza: "Manhattan", k = 1 - Accuratezza: 0.88892
Distanza: "Manhattan", k = 2 - Accuratezza: 0.88892
Distanza: "Manhattan", k = 3 - Accuratezza: 0.89053
Distanza: "Manhattan", k = 4 - Accuratezza: 0.89072
Distanza: "Manhattan", k = 5 - Accuratezza: 0.89072
Distanza: "Manhattan", k = 6 - Accuratezza: 0.89356
Distanza: "Manhattan", k = 7 - Accuratezza: 0.89413
Distanza: "Manhattan", k = 8 - Accuratezza: 0.89394
Distanza: "Manhattan", k = 9 - Accuratezza: 0.89290
Distanza: "Manhattan", k = 10 - Accuratezza: 0.89366


Miglior Accuratezza: 0.89413 (Usando distanza di "Manhattan" e con k = 7)
```

6.3 DecisionTree

Come criteri di split sono stati proposti l'Entropia e il coefficiente di Gini, in quanto entrambi permettono di gestire sia attributi categorici che numerici e massimizzano la purezza delle classi risultanti dalla suddivisione, separando chiaramente le condizioni meteorologiche in categorie distinte.

Per la profondità dell'albero è stato scelto un ampio range di valori, in modo da testare il modello con vari livelli di profondità. Le migliori prestazioni sono state ottenute con profondità pari a 10 ed Entropia come criterio di split.

Nonostante il divario delle prestazioni con gli altri valori non fosse ampio, il modello così impostato ha dimostrato di essere sufficientemente complesso da adattarsi ai dati di addestramento, ma ancora in grado di generalizzare bene sui dati di test, senza entrare in overfitting.

Segue l'output dell'esecuzione del tuning con tutte le combinazioni possibili:

```
Criterio: "gini", profondità: 1 - Accuratezza: 0.47699
Criterio: "gini", profondità: 2 - Accuratezza: 0.67453
Criterio: "gini", profondità: 3 - Accuratezza: 0.86572
Criterio: "gini", profondità: 4 - Accuratezza: 0.87907
Criterio: "gini", profondità: 5 - Accuratezza: 0.89943
Criterio: "gini", profondità: 6 - Accuratezza: 0.90597
Criterio: "gini", profondità: 7 - Accuratezza: 0.90795
Criterio: "gini", profondità: 8 - Accuratezza: 0.90786
Criterio: "gini", profondità: 9 - Accuratezza: 0.90852
Criterio: "gini", profondità: 10 - Accuratezza: 0.90682
Criterio: "gini", profondità: 11 - Accuratezza: 0.90720
Criterio: "gini", profondità: 12 - Accuratezza: 0.90455
Criterio: "gini", profondità: 13 - Accuratezza: 0.90653
Criterio: "gini", profondità: 14 - Accuratezza: 0.90653
Criterio: "gini", profondità: 15 - Accuratezza: 0.90710
Criterio: "gini", profondità: 16 - Accuratezza: 0.90587
Criterio: "gini", profondità: 17 - Accuratezza: 0.90606
Criterio: "gini", profondità: 18 - Accuratezza: 0.90606
Criterio: "gini", profondità: 19 - Accuratezza: 0.90549
Criterio: "gini", profondità: 20 - Accuratezza: 0.90540


Criterio: "entropy", profondità: 1 - Accuratezza: 0.47680
Criterio: "entropy", profondità: 2 - Accuratezza: 0.67860
Criterio: "entropy", profondità: 3 - Accuratezza: 0.87121
Criterio: "entropy", profondità: 4 - Accuratezza: 0.88362
Criterio: "entropy", profondità: 5 - Accuratezza: 0.90104
Criterio: "entropy", profondità: 6 - Accuratezza: 0.90663
Criterio: "entropy", profondità: 7 - Accuratezza: 0.90578
Criterio: "entropy", profondità: 8 - Accuratezza: 0.90616
Criterio: "entropy", profondità: 9 - Accuratezza: 0.90710
Criterio: "entropy", profondità: 10 - Accuratezza: 0.90956
Criterio: "entropy", profondità: 11 - Accuratezza: 0.90663
Criterio: "entropy", profondità: 12 - Accuratezza: 0.90568
Criterio: "entropy", profondità: 13 - Accuratezza: 0.90426
Criterio: "entropy", profondità: 14 - Accuratezza: 0.90407
Criterio: "entropy", profondità: 15 - Accuratezza: 0.90350
Criterio: "entropy", profondità: 16 - Accuratezza: 0.90407
Criterio: "entropy", profondità: 17 - Accuratezza: 0.90218
Criterio: "entropy", profondità: 18 - Accuratezza: 0.90312
Criterio: "entropy", profondità: 19 - Accuratezza: 0.90360
Criterio: "entropy", profondità: 20 - Accuratezza: 0.90303


Miglior Accuratezza: 0.90956 (Usando criterio "entropy" e profondita' "10")
```

6.4 SVM

Gli iperparametri chiave per un classificatore SVM sono sostanzialmente due: la tipologia di kernel e il parametro di regolarizzazione C.

I kernel proposti sono il kernel lineare (*linear*) per i dati linearmente separabili, il kernel polinomiale (*poly*) per modellare relazioni non lineari e il kernel radial basis function (*RBF*) per i dati con relazioni più complesse.

Il parametro di regolarizzazione C controlla il trade-off tra massimizzare il margine del classificatore e minimizzare l'errore di classificazione. Per trovare il valore ottimale è stato utilizzato un range tra 1 e 100.

Segue l'output dell'esecuzione del tuning con tutte le combinazioni possibili:

```
Kernel: linear e C: 1 - Accuratezza: 0.88097
Kernel: linear e C: 10 - Accuratezza: 0.87481
Kernel: linear e C: 20 - Accuratezza: 0.87491
Kernel: linear e C: 30 - Accuratezza: 0.87491
Kernel: linear e C: 40 - Accuratezza: 0.87358
Kernel: linear e C: 50 - Accuratezza: 0.87462
Kernel: linear e C: 60 - Accuratezza: 0.87500
Kernel: linear e C: 70 - Accuratezza: 0.87367
Kernel: linear e C: 80 - Accuratezza: 0.87443
Kernel: linear e C: 90 - Accuratezza: 0.87415
Kernel: linear e C: 100 - Accuratezza: 0.87358


Kernel: poly e C: 1 - Accuratezza: 0.82936
Kernel: poly e C: 10 - Accuratezza: 0.86193
Kernel: poly e C: 20 - Accuratezza: 0.86828
Kernel: poly e C: 30 - Accuratezza: 0.87140
Kernel: poly e C: 40 - Accuratezza: 0.87292
Kernel: poly e C: 50 - Accuratezza: 0.87377
Kernel: poly e C: 60 - Accuratezza: 0.87405
Kernel: poly e C: 70 - Accuratezza: 0.87481
Kernel: poly e C: 80 - Accuratezza: 0.87585
Kernel: poly e C: 90 - Accuratezza: 0.87652
Kernel: poly e C: 100 - Accuratezza: 0.87680
```

```
Kernel: rbf e C: 1 - Accuratezza: 0.82292
Kernel: rbf e C: 10 - Accuratezza: 0.85578
Kernel: rbf e C: 20 - Accuratezza: 0.86373
Kernel: rbf e C: 30 - Accuratezza: 0.86847
Kernel: rbf e C: 40 - Accuratezza: 0.87131
Kernel: rbf e C: 50 - Accuratezza: 0.87282
Kernel: rbf e C: 60 - Accuratezza: 0.87415
Kernel: rbf e C: 70 - Accuratezza: 0.87491
Kernel: rbf e C: 80 - Accuratezza: 0.87547
Kernel: rbf e C: 90 - Accuratezza: 0.87689
Kernel: rbf e C: 100 - Accuratezza: 0.87708



Miglior Accuratezza: 0.88097 (Usando kernel "linear" e C "1")
```

Ad offrire le prestazioni migliori è stato l'SVM con kernel lineare e parametro di regolarizzazione pari a 1, suggerendo che una semplice separazione lineare è sufficiente per questo set di dati.

6.5 Custom Ensemble

L'ensemble si presenta con due semplici iperparametri: il tipo di votazione, hard o soft, e l'uso (o meno) dei pesi per le predizioni dei modelli. Tramite una gridsearch sono state provate tutte le possibili combinazioni degli iperparametri per giungere alla configurazione ottimale.

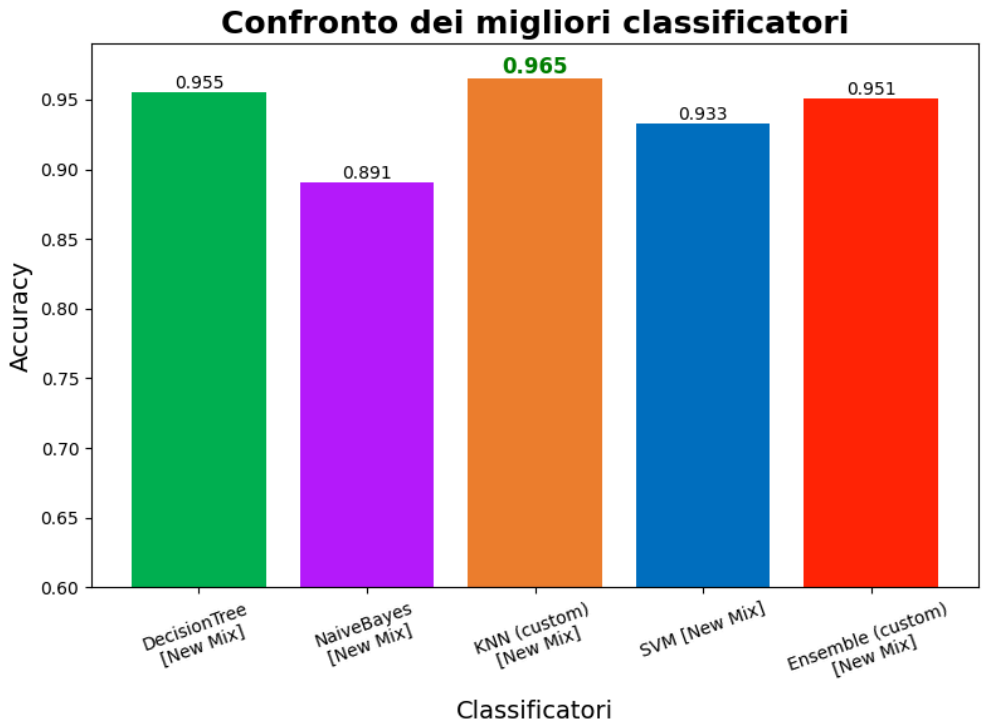
Il tuning degli iperparametri è stato applicato anche a tutti i modelli che compongono l'ensemble, in modo da sfruttarli al meglio.

```
Tipo di votazione: "hard", contributo pesato: True - Accuratezza: 0.91023
Tipo di votazione: "hard", contributo pesato: False - Accuratezza: 0.90417


Tipo di votazione: "soft", contributo pesato: True - Accuratezza: 0.91098
Tipo di votazione: "soft", contributo pesato: False - Accuratezza: 0.90985


Miglior Accuratezza: 0.91098 (Usando tipo di votazione "soft" e con contributo pesato = True)
```

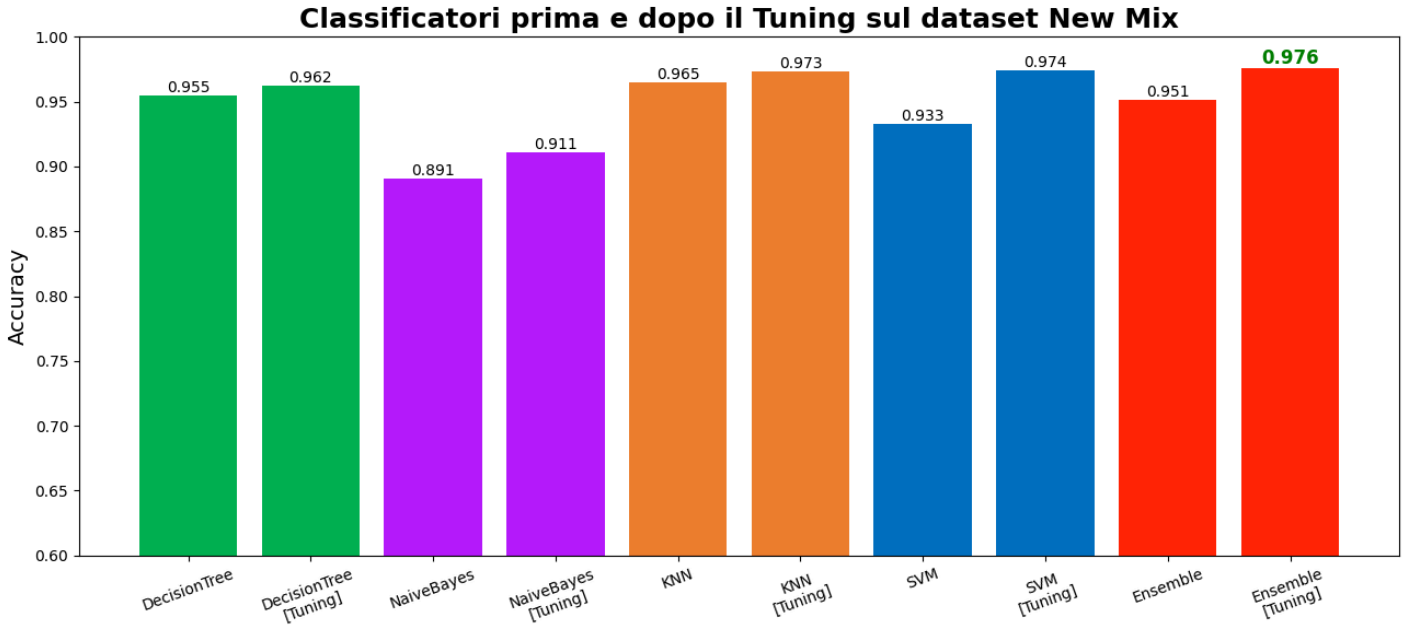
Come è possibile notare dall'output dell'esecuzione del tuning, la configurazione che permette di raggiungere le prestazioni più alte è quella con la votazione impostata a soft e il contributo pesato per ogni modello.

7. Confronti

Dopo aver analizzato le tecniche di pre-processing e le prestazioni di ogni singolo classificatore, è necessario soffermarsi per tirare le somme dei risultati ottenuti.



Il grafico, che raccoglie i migliori risultati per modello, mostra come tutti i classificatori abbiano ottenuto performance migliori sul dataset New Mix (No Outliers+Big+STD). Per provare a ottenere prestazioni ancora superiori, è stato effettuato il tuning degli iperparametri sul dataset New Mix.



Di seguito, per ogni modello, vengono riportati anche i migliori iperparametri ottenuti sul dataset New Mix.

- DecisionTree:

```
Miglior Accuratezza: 0.96326 (Usando criterio "gini" e profondita' "15")
```

- Naive Bayes:

```
Miglior Accuratezza: 0.91349 (Usando smoothing: 57)
```

Si è notato come il range utilizzato precedentemente non permettesse di raggiungere alte performance con smoothing bassi, così si è optato per un range tra 1 e 100, che raggiunge il picco delle sue performance con valore pari a 57.

- SVM:

```
Miglior Accuratezza: 0.97050 (Usando kernel "rbf" e C "20")
```

- Custom KNN:

```
Miglior Accuratezza: 0.97301 (Usando distanza di "Manhattan" e con k = 1)
```

- Custom Ensemble:

```
Miglior Accuratezza: 0.97652 (Usando tipo di votazione "soft" e con contributo pesato = True)
```

L'ottimizzazione degli iperparametri di ogni modello sul dataset New Mix porta il classificatore Ensemble a diventare il più performante, con un'accuracy di 97.6%.

Tabella riepilogativa di tutti i risultati ottenuti:

	DecisionTree	NaiveBayes	KNN	SVM	Ensemble
Base	0.912	0.867	0.898	0.887	0.911
Small	0.919	0.877	0.904	0.895	0.918
Big	0.931	0.870	0.958	0.888	0.937
Base + STD	0.912	0.809	0.897	0.895	0.904
Small + STD	0.918	0.811	0.899	0.902	0.906
Big + STD	0.931	0.815	0.959	0.895	0.938
Base + FS	0.897	0.867	0.898	0.867	0.900
Small + FS	0.919	0.879	0.902	0.883	0.913
Big + FS	0.922	0.873	0.960	0.868	0.929
Base + STD + FS	0.897	0.819	0.889	0.873	0.893
Small + STD + FS	0.910	0.833	0.900	0.886	0.908
Big + STD + FS	0.927	0.820	0.952	0.873	0.929
NewMix	0.955	0.891	0.965	0.933	0.951
NewMix (+Tuning)	0.962	0.911	0.973	0.974	0.976

8. Conclusioni

Il progetto ha esplorato l'applicazione di tecniche di machine learning per la classificazione delle condizioni meteorologiche utilizzando un dataset sintetico. Attraverso diverse fasi di pre-processing e tuning dei modelli, sono stati raggiunti risultati significativi che offrono spunti interessanti per future applicazioni.

8.1 Obiettivi raggiunti

Pre-Processing Efficace: L'adozione di tecniche di pre-processing come la standardizzazione e l'oversampling ha migliorato la qualità dei dati e le prestazioni dei modelli. La rimozione dei valori anomali e degli outlier ha ulteriormente raffinato il dataset, rendendo i dati più rappresentativi e riducendo la distorsione nei modelli.

Tuning dei Modelli: Il tuning dei modelli ha mostrato un significativo miglioramento delle prestazioni. In particolare, il classificatore Ensemble customizzato, con l'apposito tuning sul dataset pre-processato, ha raggiunto un'accuratezza del 97.6%, il miglior risultato ottenuto nel progetto.

8.2 Applicazioni future

Il lavoro svolto offre numerosi spunti per applicazioni future:

Scenari Reali: Sebbene il dataset utilizzato sia sintetico, le tecniche e le metodologie applicate possono essere facilmente adattate a scenari reali con dati meteorologici veri. Questo potrebbe aiutare nella previsione meteorologica, fornendo strumenti più accurati per settori come l'agricoltura, i trasporti e la gestione delle risorse naturali.

Espansione dei Modelli: L'integrazione di ulteriori modelli di machine learning e tecniche di ensemble potrebbe migliorare ulteriormente le prestazioni. La sperimentazione con tecniche di deep learning potrebbe offrire risultati ancora più promettenti.

In sintesi, il progetto ha dimostrato l'efficacia delle tecniche di machine learning nella classificazione delle condizioni meteorologiche, anche utilizzando un dataset sintetico. Le metodologie applicate hanno permesso di ottenere risultati robusti e affidabili, evidenziando l'importanza del pre-processing e del tuning dei modelli. Questi risultati forniscono una solida base per ulteriori ricerche e applicazioni in contesti reali, aprendo la strada a previsioni meteorologiche sempre più accurate e utili.