

UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA



*Corso di Laurea Magistrale in
Ingegneria Informatica e dell'Automazione*

Corso di Data Science

Relazione per il progetto di Clustering, Classificazione e Forecasting

Studenti:

Angelone Mattia Domenico

Romanelli Marco

Giannelli Edoardo

Ali Waqar Badar

Anno Accademico 2022-2023

Python

Indice

1. Introduzione	3
1.1 Primo dataset	3
1.2 Secondo dataset.....	4
1.3 Preprocessing e analisi descrittiva (primo dataset)	5
2. Clustering.....	12
2.1 Clustering Max_air_temp / Max_rel_hum.....	13
2.1.1 K-Means Clustering.....	13
2.1.2 Silhouette	15
2.1.3 Hierarchical Agglomerative Clustering	16
2.2 Clustering Mean_air_temp / Mean_rel_hum.....	17
2.2.1 K-Means Clustering.....	17
2.2.2 Silhouette	19
2.2.3 Hierarchical Agglomerative Clustering	20
2.3 Clustering Sum_precip_amount e Mean_rel_hum	21
2.3.1 K-Means Clustering.....	21
2.3.2 Silhouette	23
2.3.3 Hierarchical Agglomerative Clustering	24
3. Serie Temporali.....	25
3.1 Preprocessing serie temporale	25
3.2 Analisi della stazionarietà	28
3.3 Stima parametro q	28
3.4 Stima parametro p	29
3.5 Stima parametri stagionalità (P, D, Q).....	30
3.6 Search Grid	32
3.7 Sviluppo di un modello della serie	34
3.8 Predizione & forecasting.....	35
3.9 Valutazione affidabilità del modello	36
4. Classificazione.....	38
4.1 Preprocessing Classificazione.....	38
4.2 Training e testing dei classificatori	41
4.3 Valutazione dei Classificatori.....	42
4.4 Conclusioni	45

1. Introduzione

In questo breve excursus introduttivo, si parlerà di due dataset differenti.

Nel primo, si affronterà l'analisi di serie temporali e di clustering, utilizzando un dataset contenente dati metereologici della Norvegia. Nel secondo, invece, si tratterà di classificazione, attraverso un dataset relativo ai funghi.

Il primo dataset è stato oggetto di uno studio che ha evidenziato come l'assenza di parametri importanti come le etichette di classificazione può limitare la capacità di classificazione dei dati. Infatti, il dataset contenente i dati meteorologici della Norvegia è stato utilizzato per l'analisi di serie temporali e di clustering, ma non è stato possibile effettuare una classificazione corretta dei dati a causa della mancanza di informazioni.

Nel secondo caso, invece, è stato possibile effettuare una classificazione accurata grazie all'utilizzo di un dataset relativo ai funghi. Questo dataset ha permesso di identificare e classificare correttamente i funghi in base alle loro caratteristiche, consentendo di ottenere risultati soddisfacenti. In generale, l'utilizzo di dataset ben strutturati e completi rappresenta un fattore determinante per ottenere risultati efficaci e precisi nelle analisi dei dati.

1.1 Primo dataset

Questo dataset contiene i vari dati metereologici di diverse stazioni meteo in Norvegia.

Il dataset è pubblico e consultabile al seguente link:

<https://www.kaggle.com/datasets/annbengardt/noway-meteorological-data> .

È composto da 237629 righe e 13 colonne. I dati presenti vanno dal 2010 al 2022 (escluso). Di seguito una tabella descrittiva dei vari campi del dataset:

Nome Parametro	Tipo	Descrizione
sourceId	string	ID relativo alla specifica stazione metereologica
latitude	float64	Latitudine coordinate stazione metereologica
longitude	float64	Longitudine coordinate stazione meterologica
max(air_temperature P1D)	float64	Temperatura massima (per un giorno)
max(relative_humidity P1D)	float64	Umidità massima (per un giorno)
max(wind_speed P1D)	float64	Vento massimo (per un giorno)
mean(air_temperature P1D)	float64	Temperatura media (per un giorno)
mean(relative_humidity P1D)	float64	Umidità media (per un giorno)
mean(wind_speed P1D)	float64	Vento medio (per un giorno)
sum(precipitation_amount P1D)	float64	Precipitazioni (per un giorno)
day	int64	Giorno
month	int64	Mese
year	int64	Anno

1.2 Secondo dataset

Il dataset in questione contiene informazioni relative a 23 specie di funghi appartenenti alla famiglia Agaricus e Lepiota. Per ogni specie sono fornite informazioni sulla commestibilità, indicando se il fungo è commestibile o velenoso, e su tale parametro è stata fatta la classificazione.

Il dataset è pubblico e consultabile al seguente Link: [Mushroom - UC Irvine Machine Learning Repository](#). È composto da 8124 righe e 22 attributi, i quali sono riassunti nella seguente tabella:

Attributo	Possible Values	Descrizione
cap-shape	bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s	Forma del cappello
cap-surface	fibrous=f, grooves=g, scaly=y, smooth=s	Superficie del cappello
cap-color	brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y	Colore del cappello
bruises?	bruises=t, no=f	Presenza di macchie sul cappello
odor	almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s	Odore
gill-attachment	attached=a, descending=d, free=f, notched=n	Attaccatura delle lamelle
gill-spacing	close=c, crowded=w, distant=d	Spaziatura delle lamelle
gill-size	broad=b, narrow=n	Dimensione delle lamelle
gill-color	black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y	Colore delle lamelle
stalk-shape	enlarging=e, tapering=t	Forma del gambo
stalk-root	bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?	Tipo di radici del gambo
stalk-surface-above-ring	fibrous=f, scaly=y, silky=k, smooth=s	Superficie del gambo sopra l'anello
stalk-surface-below-ring	fibrous=f, scaly=y, silky=k, smooth=s	Superficie del gambo sotto l'anello
stalk-color-above-ring	brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y	Colore del gambo sopra l'anello
stalk-color-below-ring	brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y	Colore del gambo sotto l'anello
veil-type	partial=p, universal=u	Tipo di velo
veil-color	brown=n, orange=o, white=w, yellow=y	Colore del velo
ring-number	none=n, one=o, two=t	Numero di anelli
ring-type	cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z	Tipo di anello
spore-print-color	black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y	Colore delle spore
population	abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y	Popolazione
habitat	grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d	Habitat
poisonous	edible=e, poisonous=p	Velenoso/Commestibile

1.3 Preprocessing e analisi descrittiva (primo dataset)

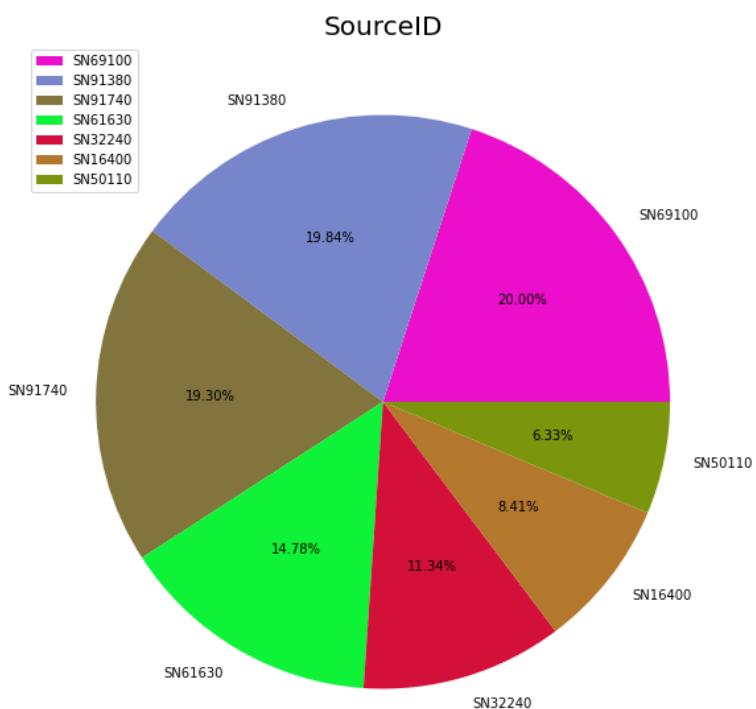
La prima fase si è basata sul fare ETL dei dati. Nello specifico attraverso Pandas una libreria di Python, i passaggi applicati una volta fatto il caricamento del dataset come DataFrame:

- Eliminazione dei parametri NaN (per mezzo della funzione ‘ .dropna() ’), questo ci ha permesso di eliminare tutti i campi vuoti, riducendo le righe da 237629 a 21799.
- Trasformazione delle colonne Day, Month e Year, in un'unica colonna contenente la data.

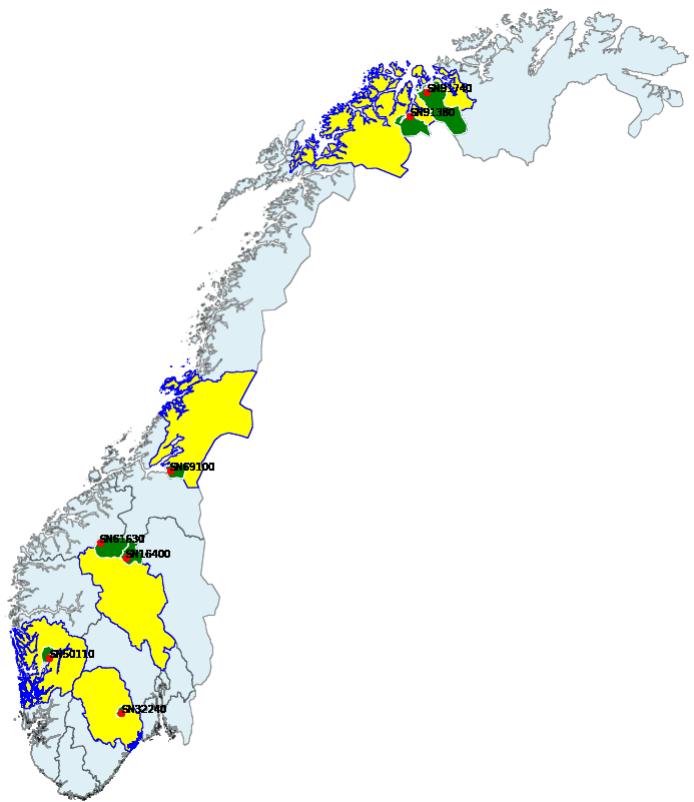
In un secondo momento è stata svolta l'analisi descrittiva dei vari parametri del dataset.

In funzione dell'attributo 'sourceID' è stata fatta un analisi quantitativa dei dati per ogni singola stazione meteorologica. Di seguito sono riportate le conversioni dell'id con le città associate e il numero di dati per ognuna di esse:

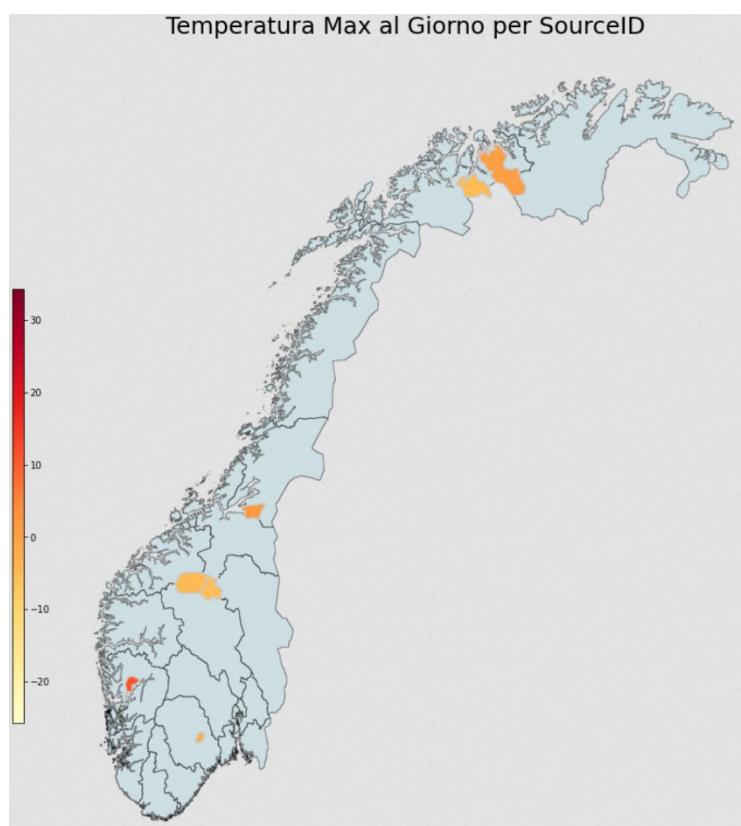
Source ID	City	Parameter
SN50110	Kvam	1380
SN69100	Stjørdal	4360
SN16400	Dovre	2472
SN61630	Lesja	4325
SN32240	Bø	1833
SN91740	Nordreisa	3222
SN91380	Storfjord	4207



Attraverso gli shapefile i quali sono immagini georeferenziate, è stato possibile stampare a schermo una cartina della Norvegia evidenziando le zone di interesse delle nostre stazioni metereologiche.

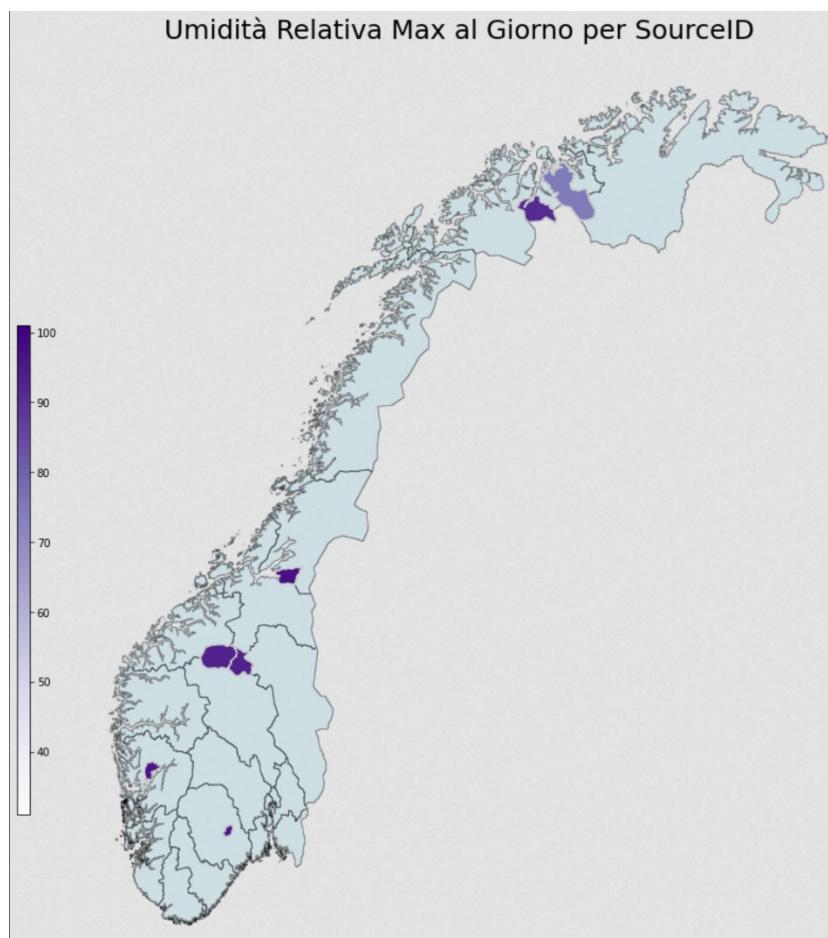


Nello specifico è stata fatta un'analisi sui parametri 'temperatura massima', 'umidità massima', 'vento massimo'. Per ogni parametro vengono riportati da un punto di vista grafico per ogni stazione meteorologica:

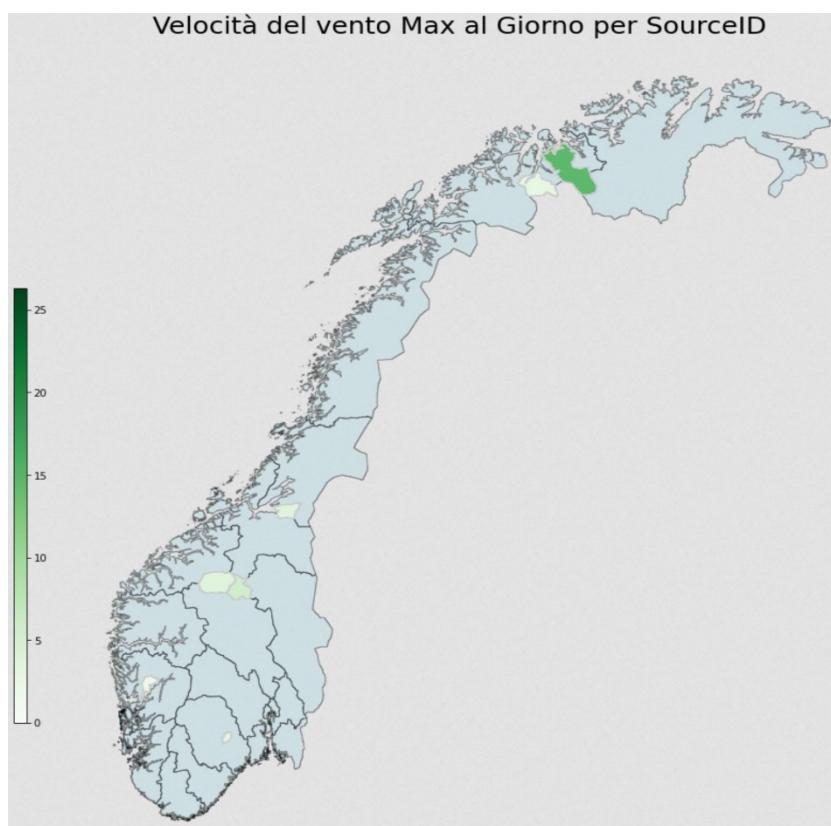


Sembrerebbe che nella città di Kvam si siano riscontrate le temperature massime.

Analogamente è stato fatto per l'umidità relativa di ogni stazione metereologica, dove si evince che solo la stazione metereologica di Storfjord ha registrato valori più bassi rispetto alle altre stazioni.

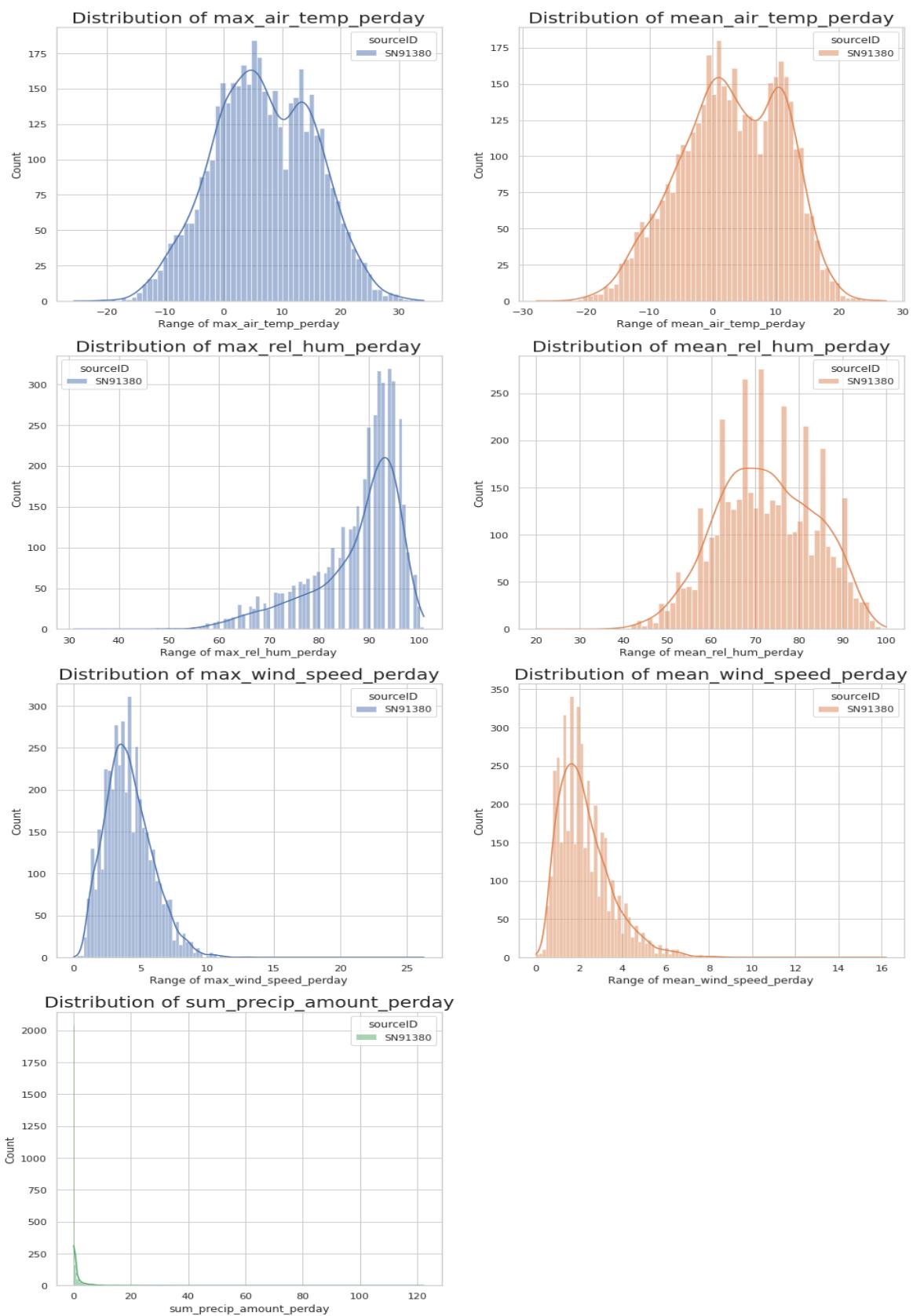


Medesimo grafico è stato fatto per il parametro relativo al vento massimo, in questo caso è possibile vedere che la città di Nordreisa ha registrato i valori massimi.



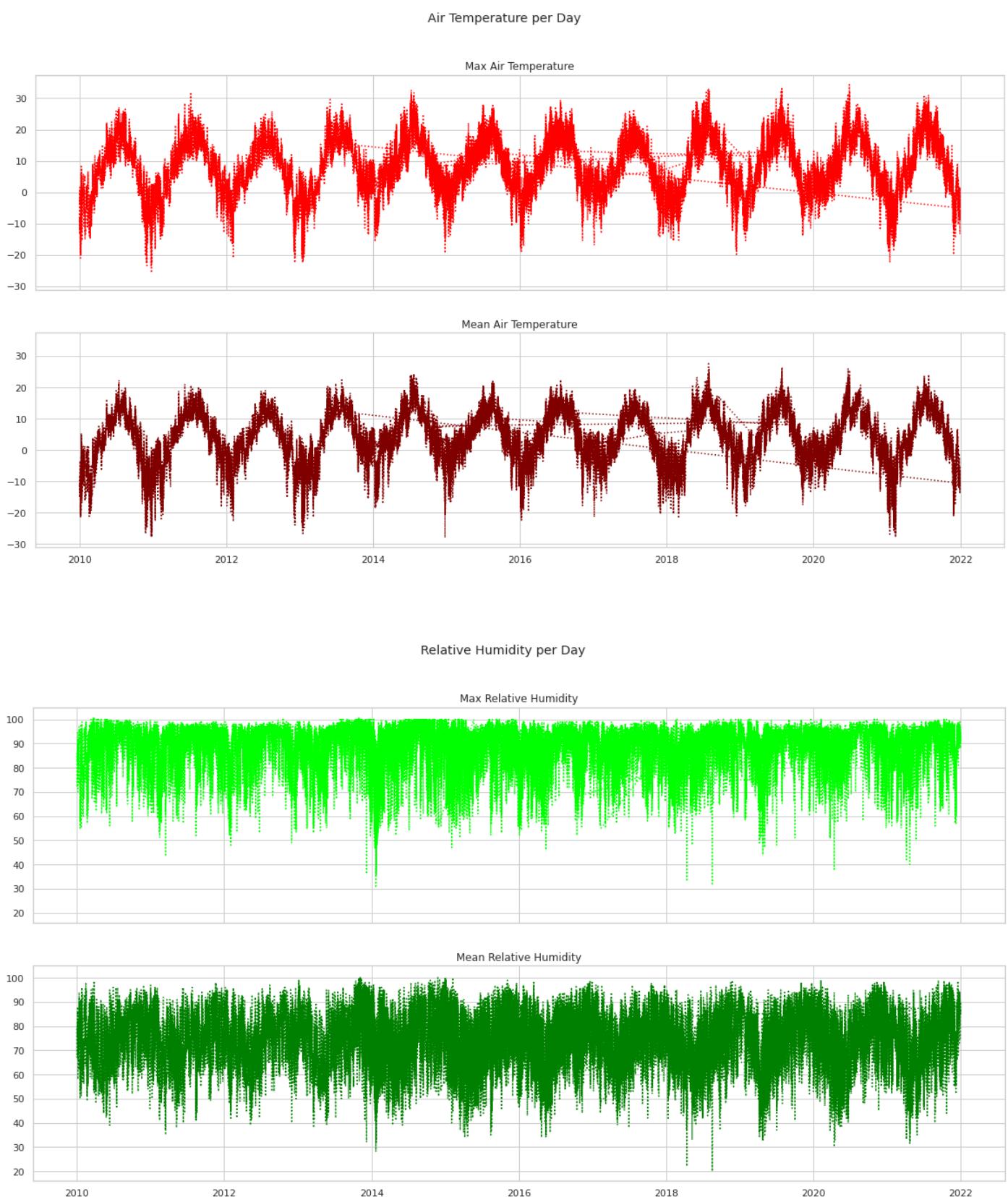
Dopo aver fatto un'analisi dei vari parametri mettendoli a confronto con ogni stazione metereologica, è stato fatto uno studio sulla distribuzione dei vari dati per ogni singola stazione metereologica.

Per semplicità saranno mostrati i grafici delle distribuzioni di una sola stazione, ovvero la stazione SN91380:

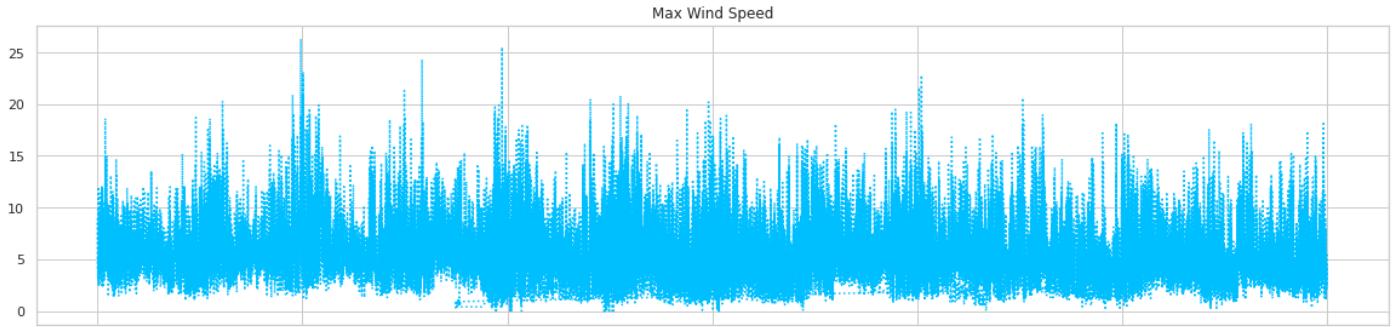


Si può notare come tutte le distribuzioni hanno un andamento a campana con la presenza di un picco o, nel caso della temperatura, di due picchi.

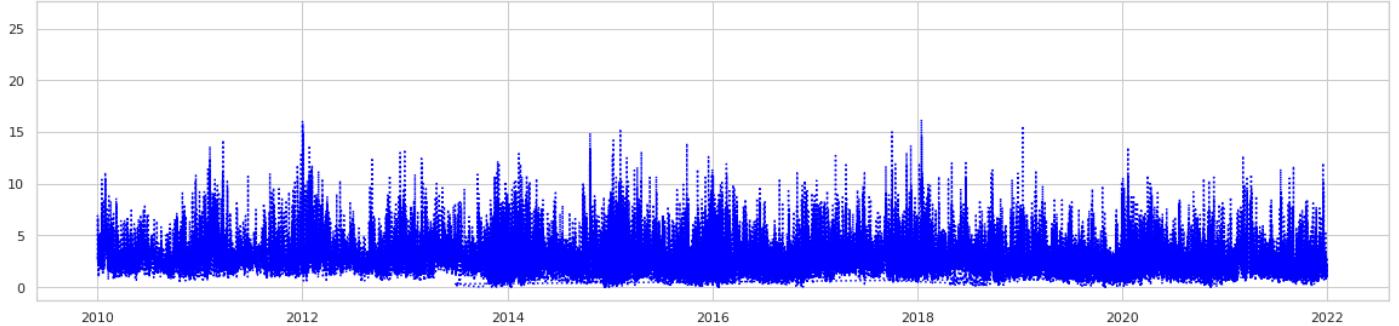
Sono stati anche analizzati gli andamenti temporali globali delle variabili metereologiche presenti nel dataset, le quali presentano una stagionalità che verrà analizzata in modo più approfondito quando di tratterà la parte delle serie temporali:



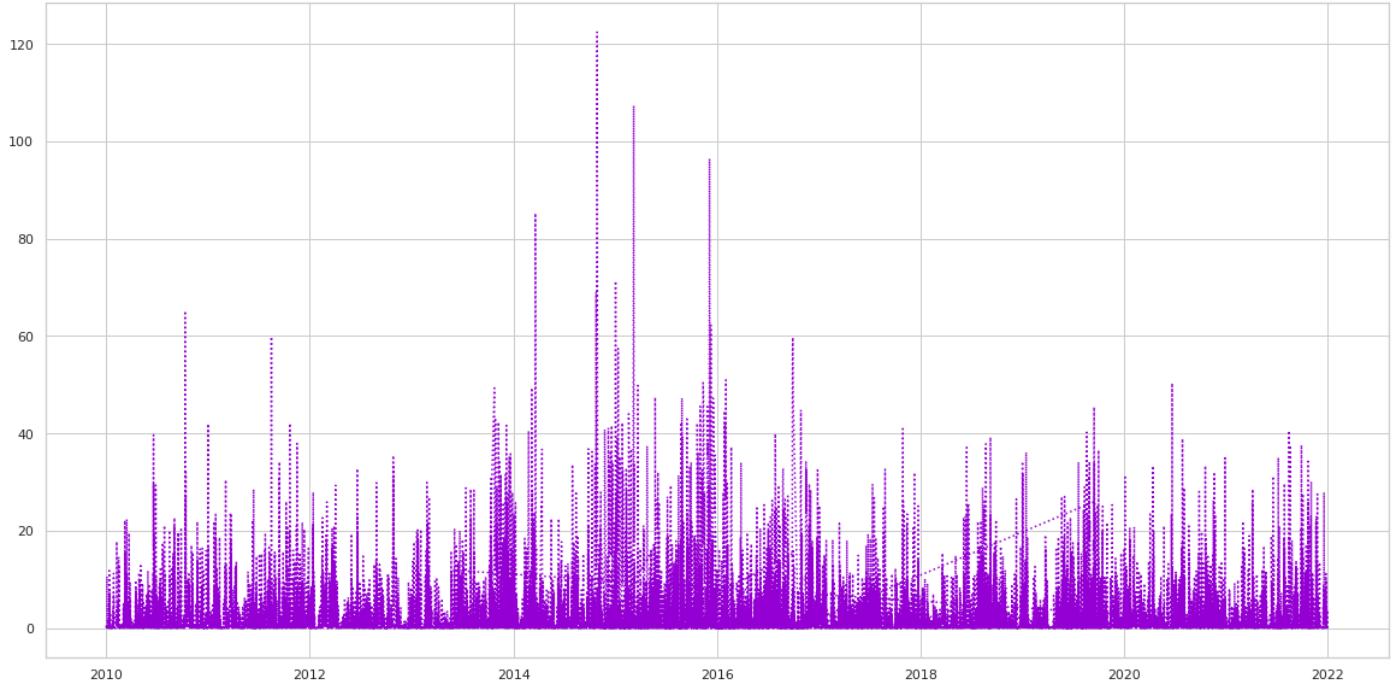
Wind Speed per Day



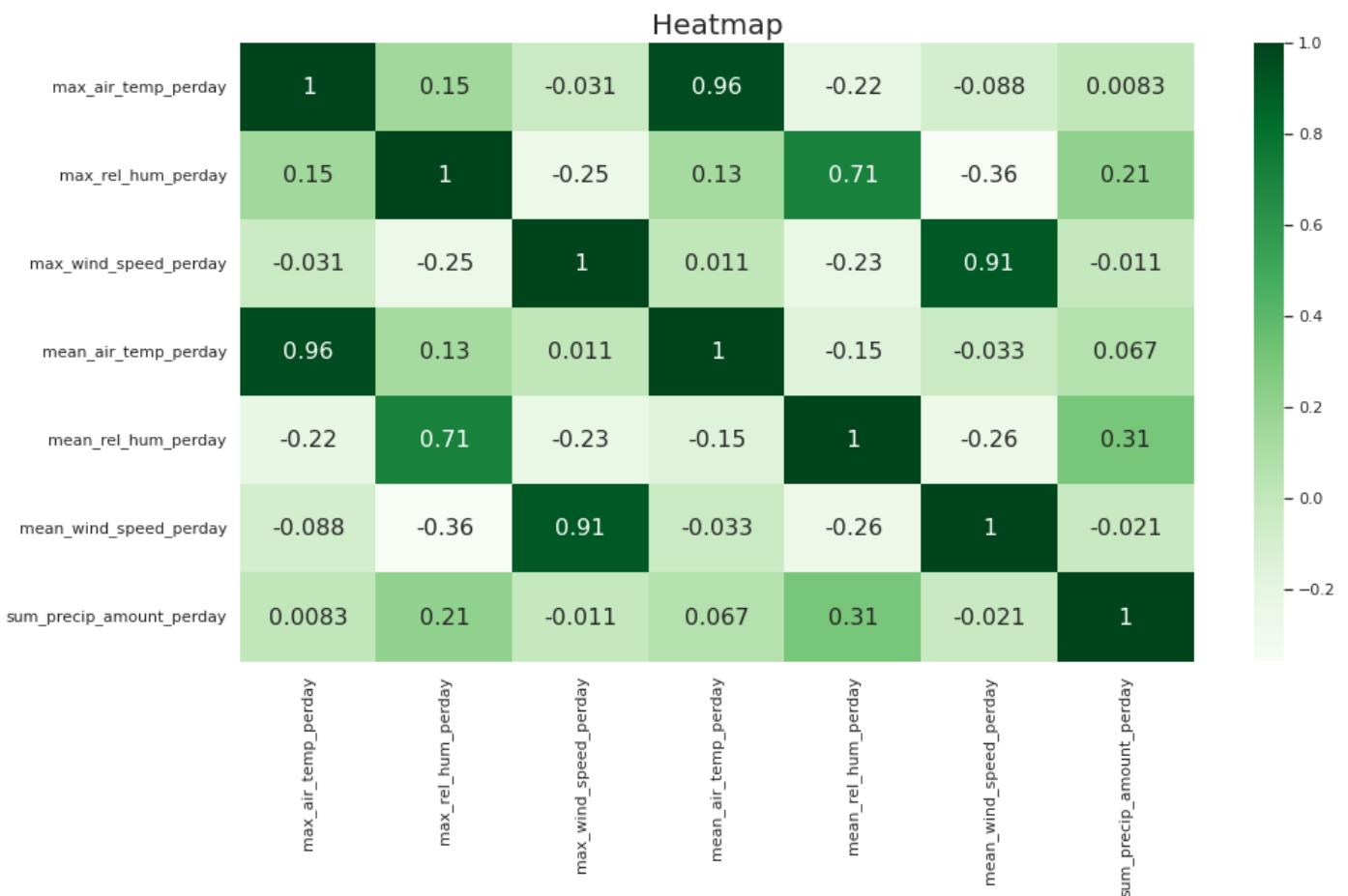
Mean Wind Speed



Sum of Precipitation Amount per Day



Come ultimo aspetto dell'analisi descrittiva del dataset è stata analizzata la correlazione tramite una matrice di correlazione:



Come si può notare, salta subito all'occhio una alta correlazione tra i vari termini max e mean delle variabili air_temp, rel_hum e wind_speed (un risultato del tutto normale considerando la natura delle variabili analizzate).

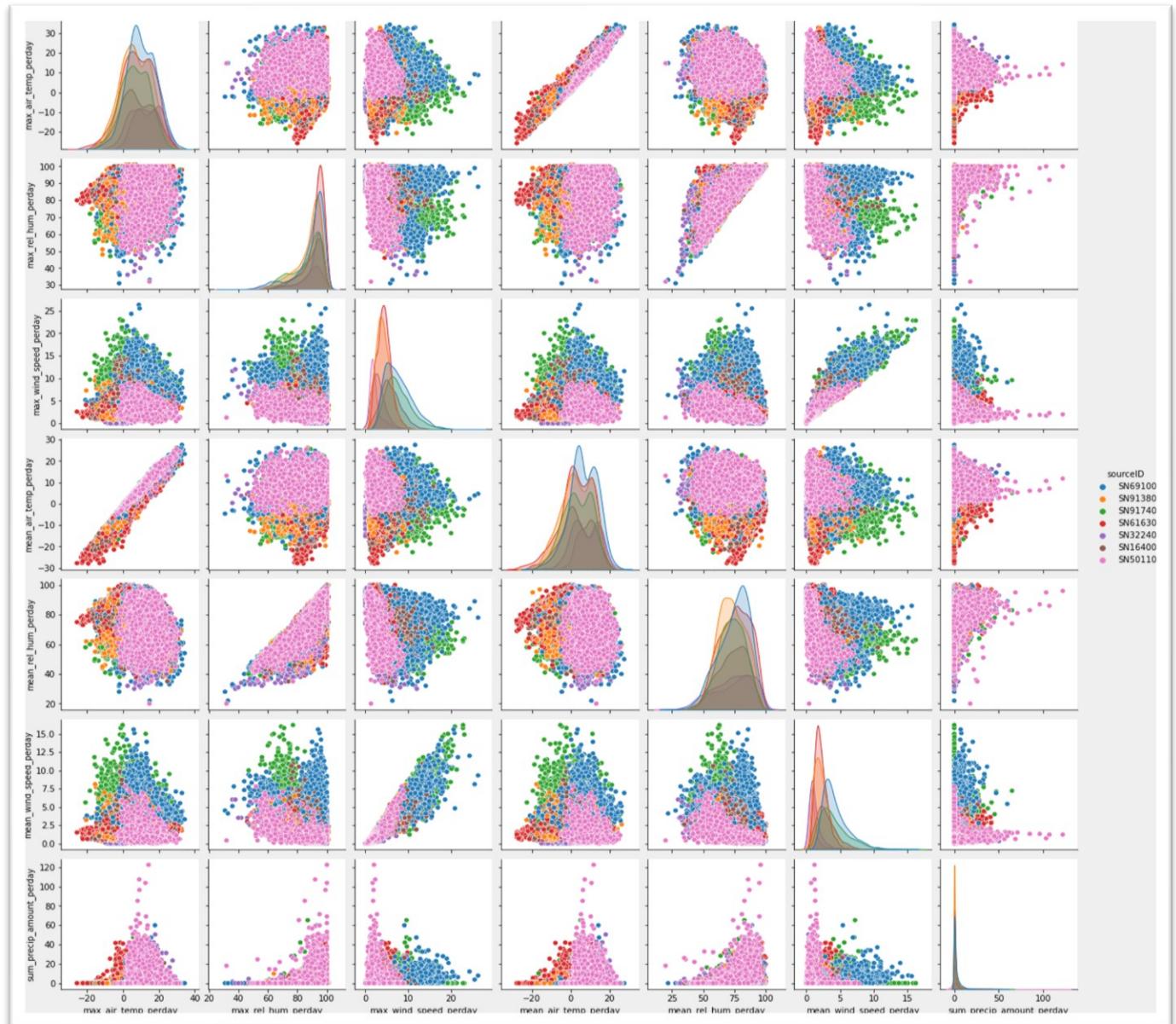
Questa analisi della correlazione verrà trattata ed analizzata meglio nel seguente task del progetto, ovvero quello del Clustering.

2. Clustering

Come primo task è stata eseguita un'operazione di clustering sul primo dataset al fine di identificare la presenza di elementi omogenei nei dati.

Infatti, l'obiettivo del clustering consiste nell'individuare all'interno del dataset gruppi di dati che presentano somiglianze tra di loro, ossia i cluster.

Per avere una prima visualizzazione dei dati si è utilizzata la funzione “pairplot” della libreria “seaborn”, che permette di graficare a coppie i valori degli attributi in un piano cartesiano.



Come visto in precedenza nella heatmap, anche in questo caso le coppie “max_air_temp/mean_air_temp”, “max_rel_hum / mean_rel_hum” e “max_wind_speed / mean_wind_speed” risultano altamente correlate.

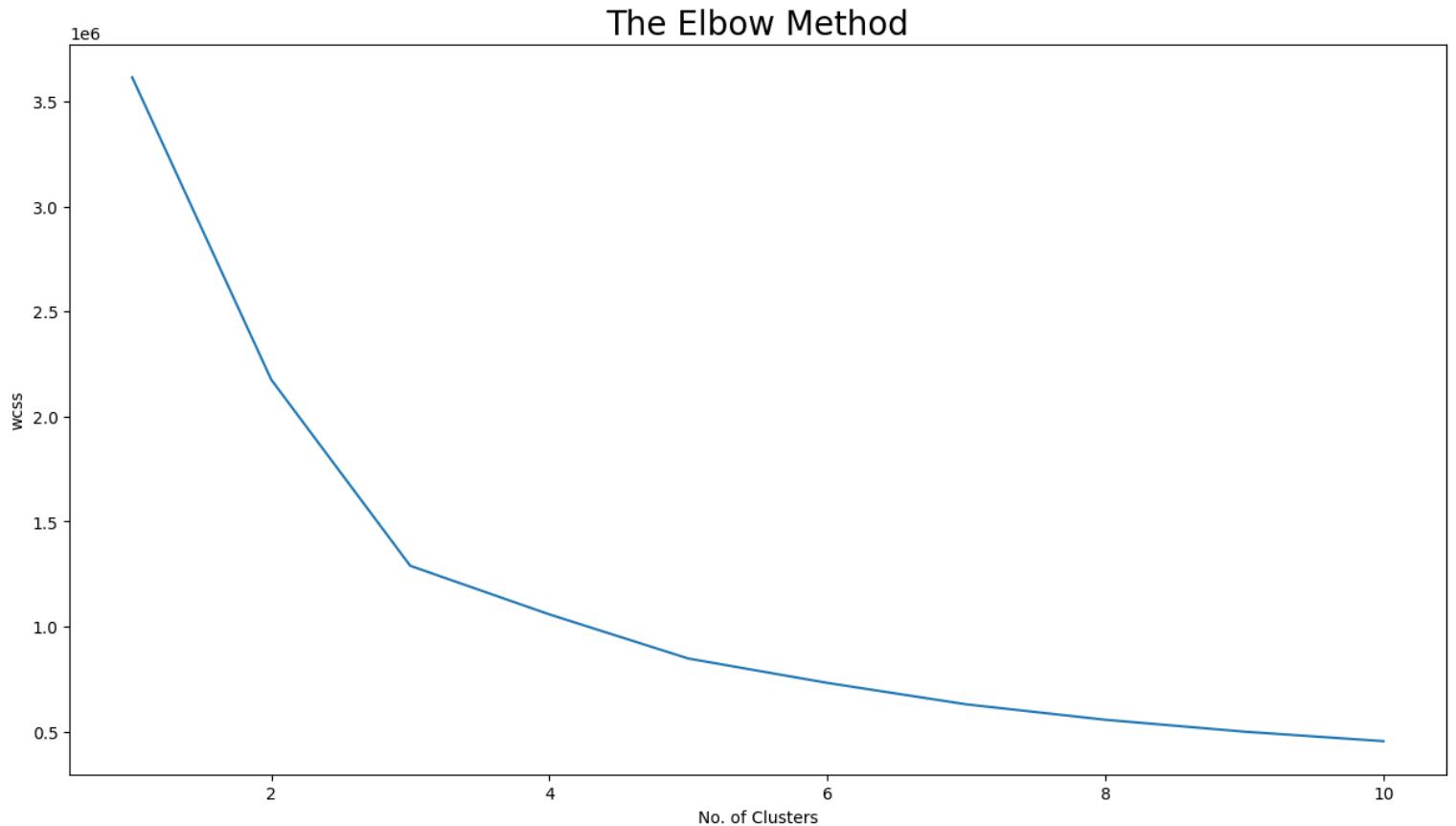
2.1 Clustering Max_air_temp / Max_rel_hum

Inizialmente, sono stati selezionati i dati relativi ai campi Max_air_temp e Max_rel_hum del dataset come prima operazione di clustering. La scelta di questi campi è stata motivata dal grafico generato tramite la funzione "pairplot", il quale ha mostrato una distribuzione uniforme dei valori nel piano cartesiano.

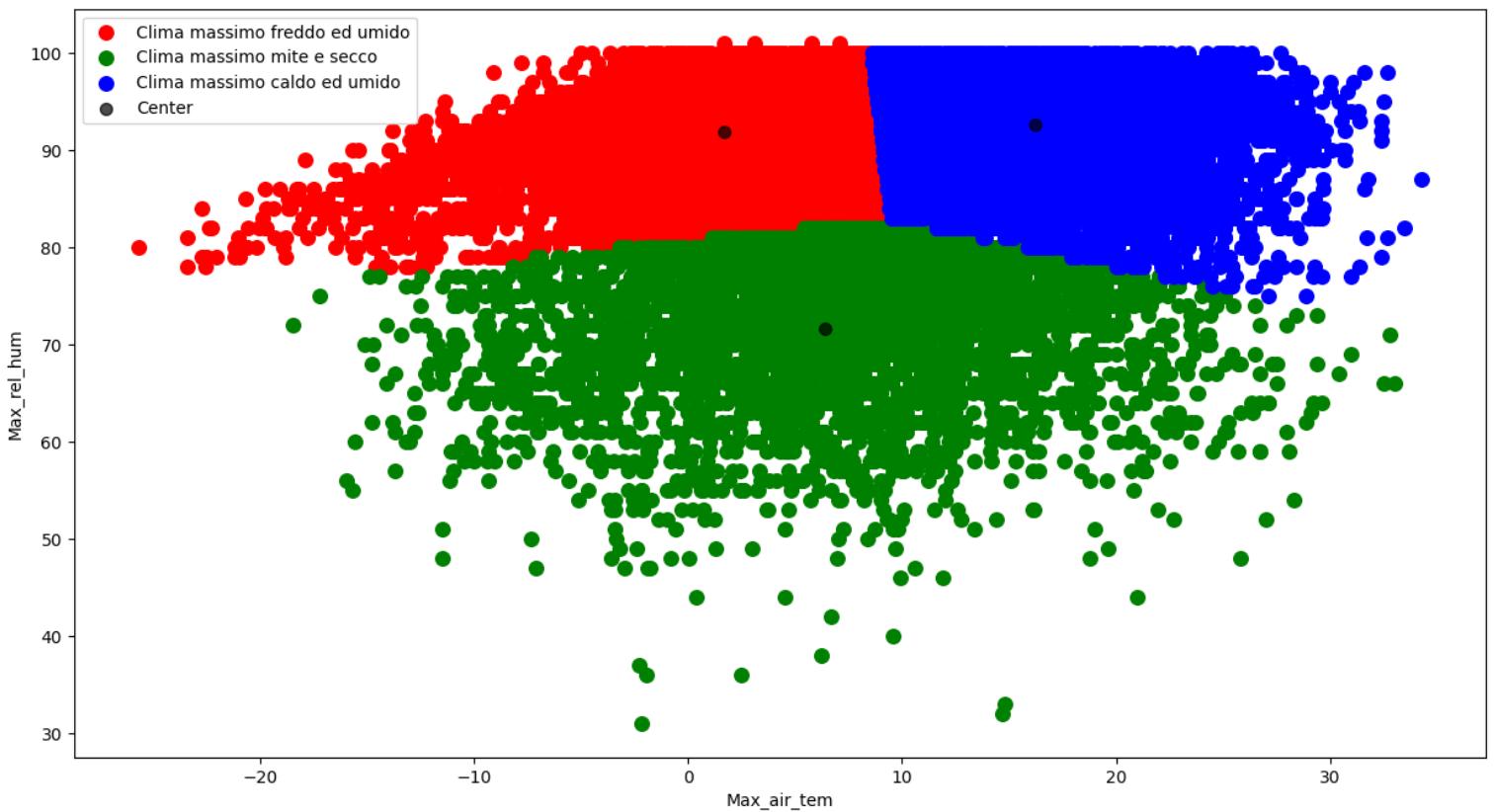
2.1.1 K-Means Clustering

Per il clustering, si è utilizzato l'algoritmo K-Means, il quale minimizza la varianza intra-gruppo totale e identifica ogni gruppo tramite un centroide o punto medio. L'algoritmo è stato eseguito in più fasi: inizialmente, sono stati determinati casualmente k punti di riferimento, ai quali sono stati associati gli elementi più vicini per formare i k cluster. Successivamente, è stato determinato il centroide di ogni cluster, il quale è diventato il nuovo punto di riferimento. Questa procedura è stata iterata finché le posizioni dei centroidi non convergono.

Per utilizzare l'algoritmo K-Means, è stato necessario specificare il numero di cluster da individuare, che è stato selezionato utilizzando l'Elbow Method. In particolare, la scelta del numero di cluster è stata effettuata osservando la curva dell'Elbow Method, riportata nella figura sotto, e selezionando il punto in cui la curva inizia a deviare. In questo caso, il numero di cluster scelto è stato $n=3$.



A questo punto è stato possibile effettuare il clustering mediante il K-Means, ottenendo il seguente risultato:



Sono stati individuati 3 gruppi. In particolare si ha:

- Il cluster rosso: corrispondente a record con valori alti di Max_rel_hum e valori bassi di Max_air_temp
- Il cluster verde: corrispondente a record con valori bassi di Max_rel_hum e valori uniformemente distribuiti di Max_air_temp
- Il cluster blu: corrispondente a record con valori alti di Max_rel_hum e di Max_air_temp

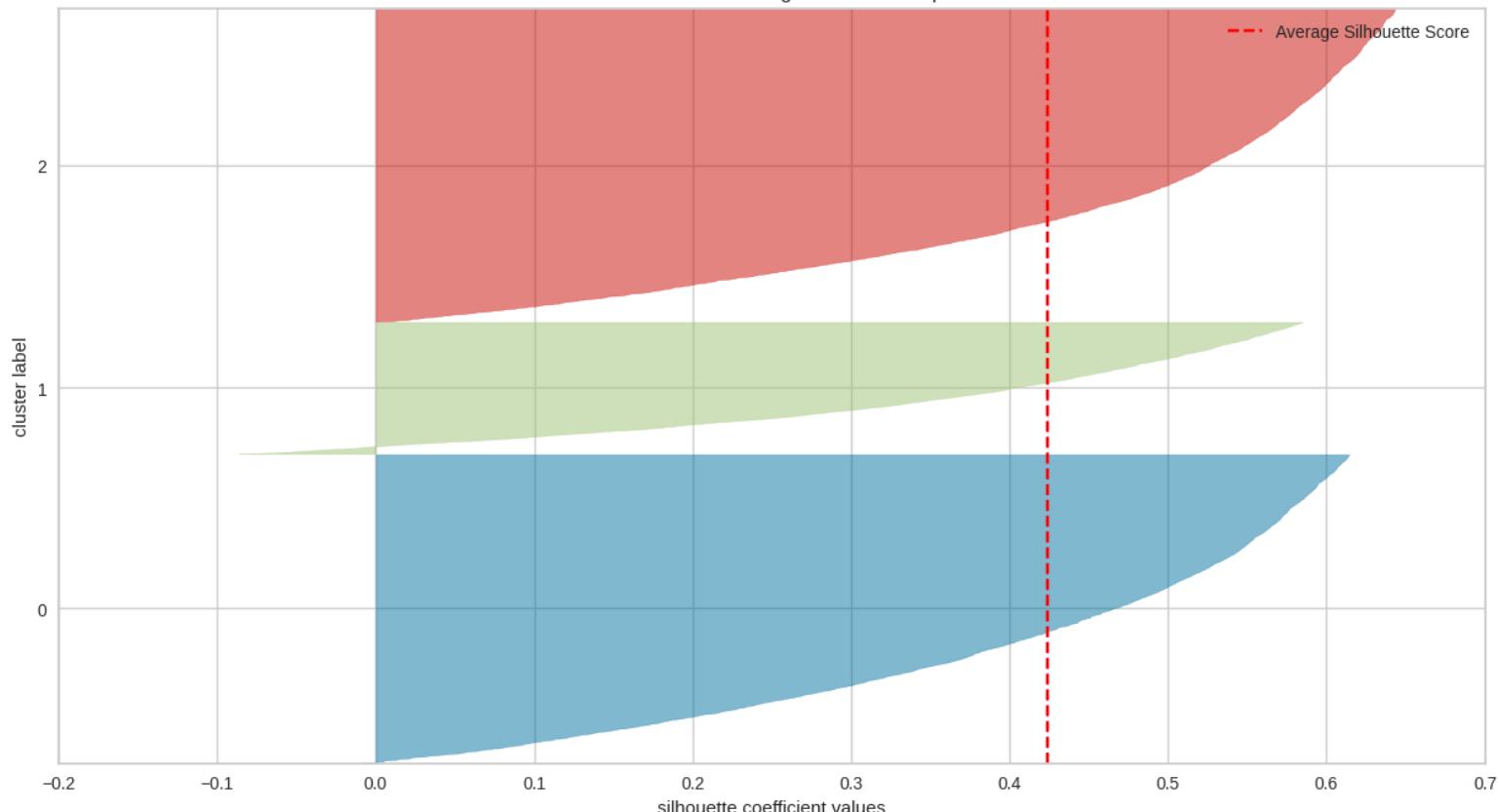
In conclusione, si può notare come nei giorni con valori bassi di massima umidità relativa la massima temperatura raggiunta dall'aria varia uniformemente passando da valori bassi (anche sotto lo zero) a valori alti (superando anche i 30 gradi). Mentre con valori alti di umidità relativa si possono individuare due cluster differenti dove nel primo abbiamo giorni con temperature dell'aria basse (con picchi che vanno al di sotto dei -20 gradi) e nel secondo ci sono giorni con temperature dell'aria alte (con picchi che superano i 30 gradi).

2.1.2 Silhouette

Si è optato per l'utilizzo della metrica Silhouette al fine di valutare i risultati in modo più accurato. Tale metrica quantifica la somiglianza di un oggetto rispetto agli oggetti appartenenti allo stesso cluster, rispetto a quelli contenuti in altri cluster, fornendo un valore compreso tra [-1, 1]. Un valore di Silhouette tendente a 1 indica che l'oggetto è ben collocato all'interno del cluster e differisce significativamente dalle caratteristiche degli oggetti presenti nei cluster vicini.

Si può vedere nel grafico seguente come molti elementi superino la Silhouette media maggiore di 0.40 e non sono praticamente presenti valori negativi. Si è quindi ritenuto il risultato di clustering accettabile.

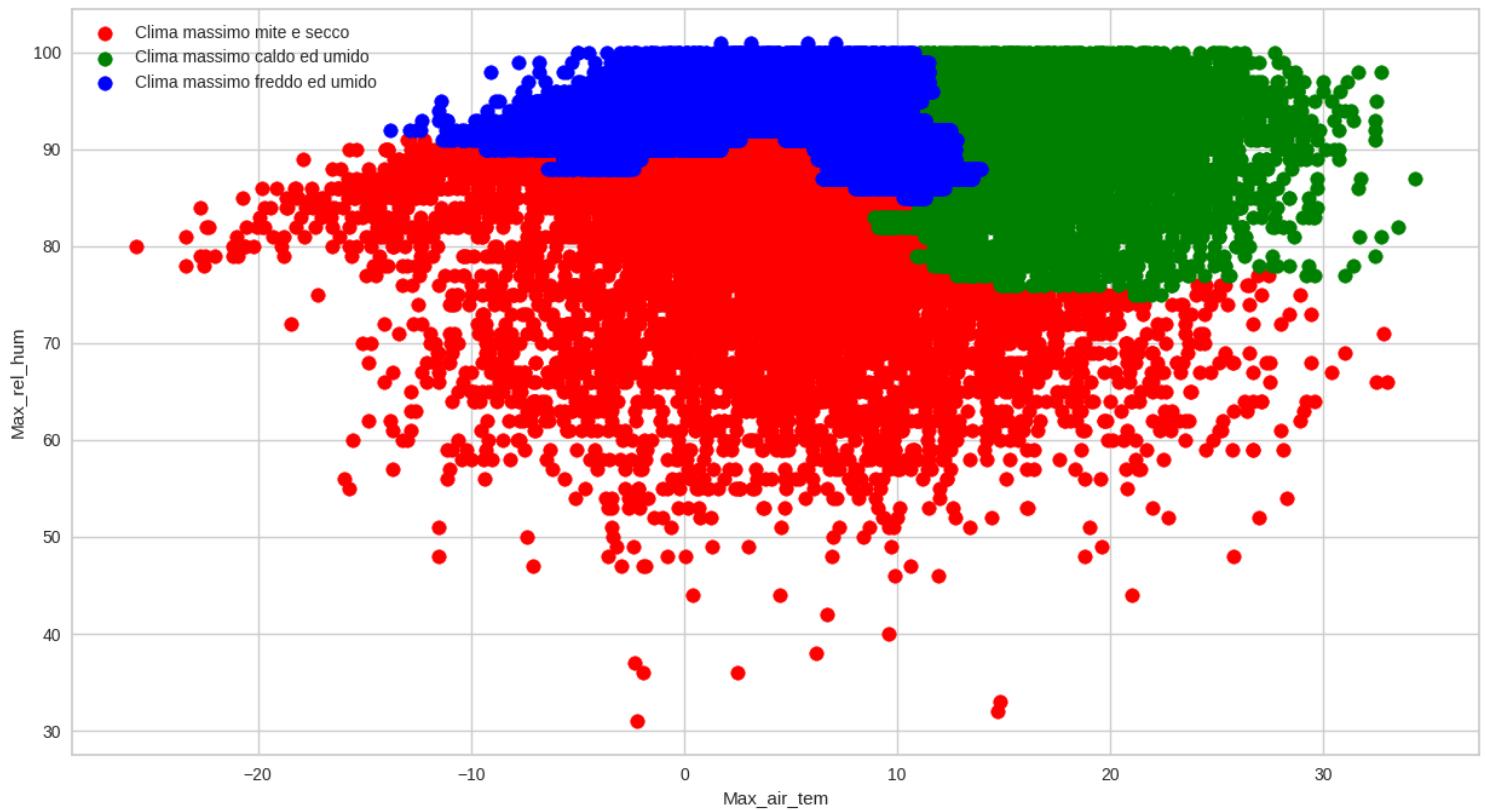
Silhouette Plot of KMeans Clustering for 21799 Samples in 3 Centers



2.1.3 Hierarchical Agglomerative Clustering

Per validare ulteriormente il risultato precedente, viene utilizzato un secondo tipo di algoritmo, ovvero l'agglomerative clustering, una variante del clustering gerarchico che mira a costruire una gerarchia di cluster. Ci sono due tipi di clustering gerarchico: agglomerativo e divisivo. L'agglomerativo utilizza una strategia "bottom-up" partendo da singoli punti in cluster separati e unendoli iterativamente a coppie fino ad ottenere cluster più grandi. Per scegliere il numero di cluster "k" è possibile utilizzare l'elbow method, come per altri algoritmi di clustering.

Si può notare come il risultato sia molto simile al risultato del K-Means Clustering e ciò porta a confermare la validità del risultato di clustering.

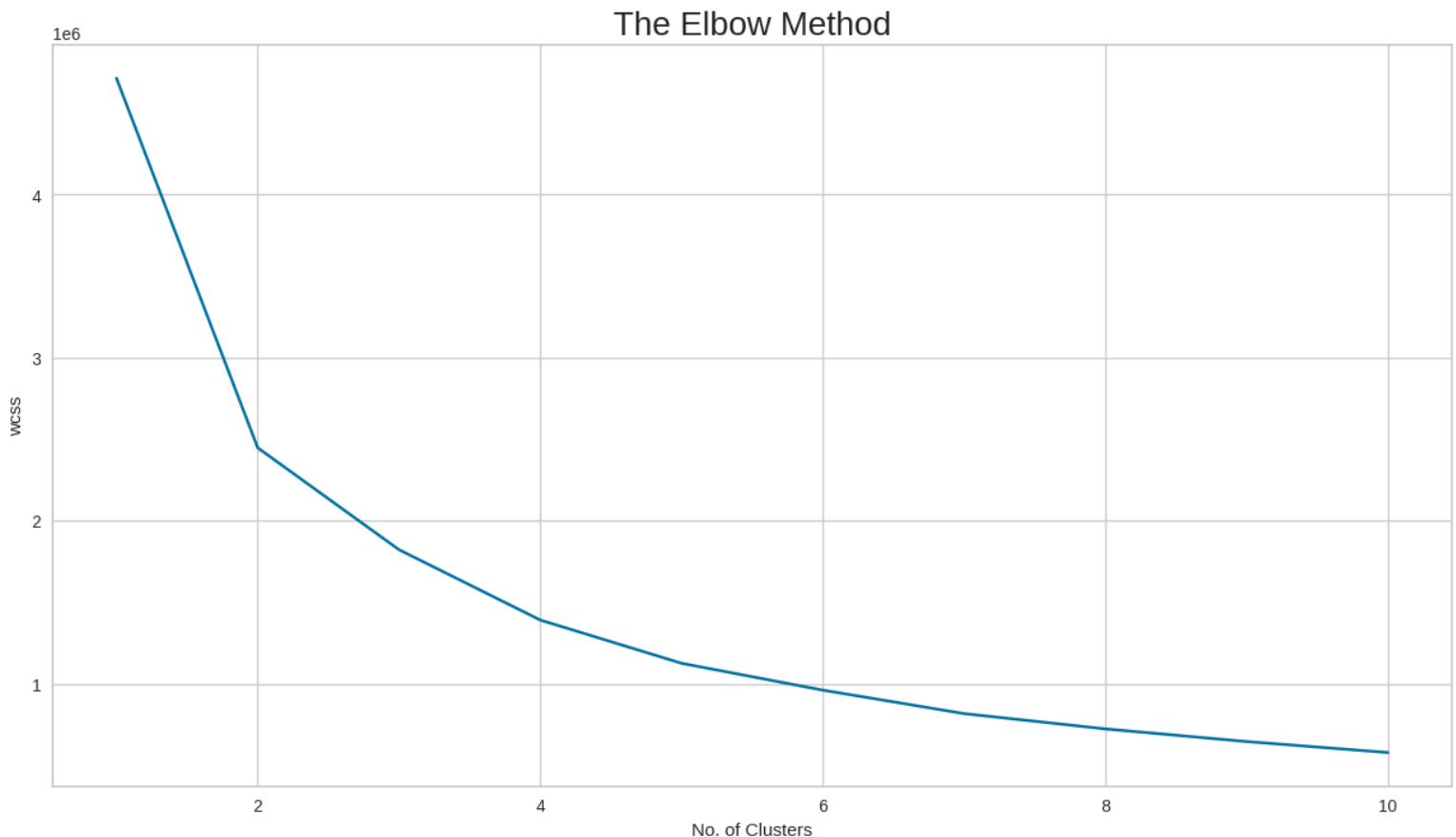


2.2 Clustering Mean_air_temp / Mean_rel_hum

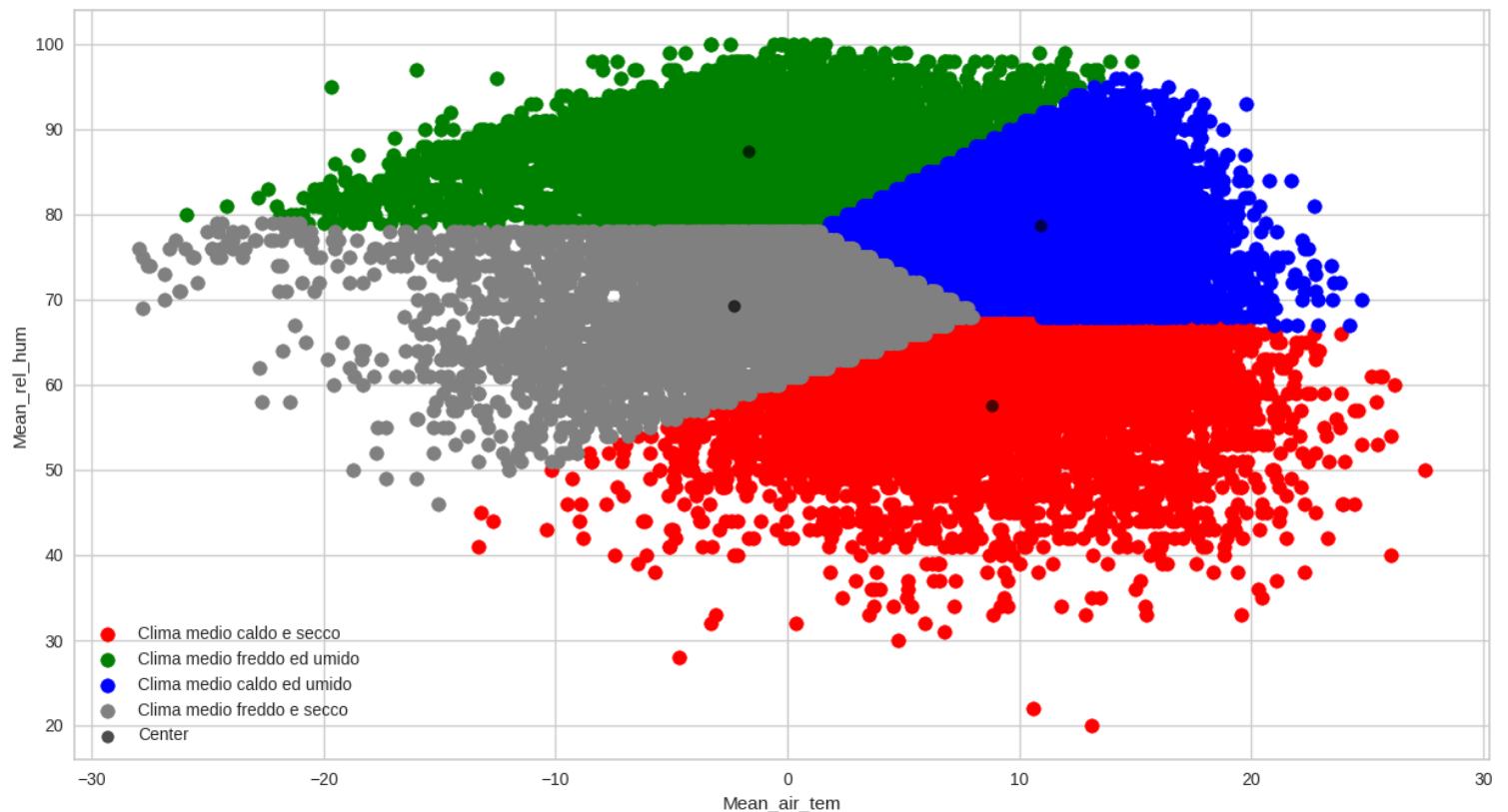
Visti i risultati ottenuti dal clustering tra i valori massimi di temperatura dell'aria e di umidità relativa, abbiamo voluto analizzare anche la correlazione tra i valori medi delle due variabili.

2.2.1 K-Means Clustering

Come precedentemente fatto, abbiamo utilizzato il K-Means Clustering per analizzare la correlazione tra mean_air_temp ed il mean_rel_hum. Di seguito possiamo vedere l'Elbow Method utilizzato per poter scegliere il numero di cluster da prendere in considerazione ai fini dell'algoritmo K-Means. A differenza del clustering precedente, in questo caso il numero di cluster scelto è stato n=5.



Successivamente è stato effettuato il clustering mediante il K-Means, ottenendo il seguente risultato:



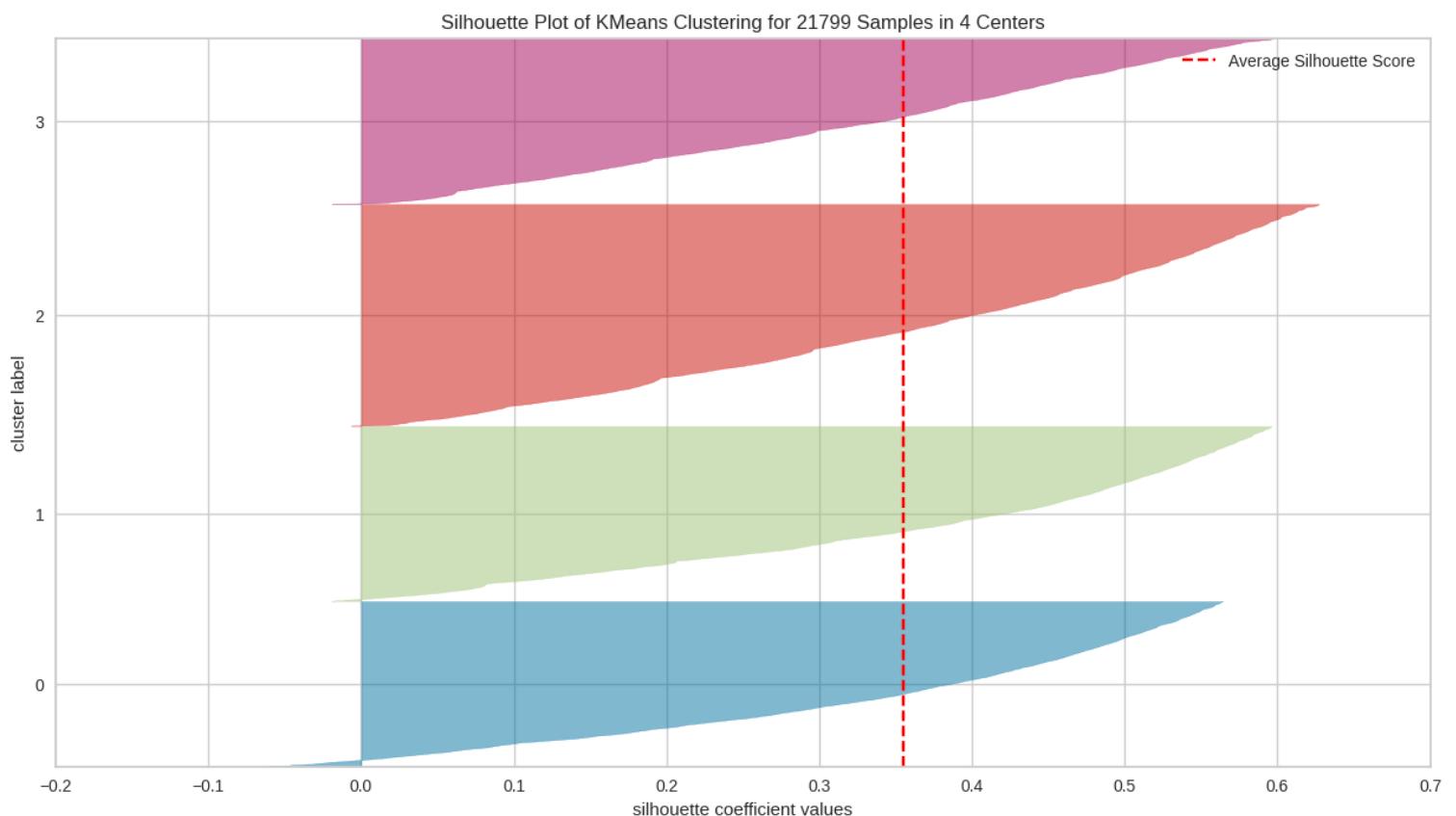
Dal risultato del clustering si può notare come non ci sia una forte correlazione tra i valori medi di temperatura dell'aria e umidità relativa ma possiamo comunque notare come a valori molto bassi di umidità corrispondano solo valori miti di temperatura (tra 10 e 15 gradi) mentre a valori massimi di umidità corrispondano solo temperature intorno agli zero gradi.

Inoltre per temperature molto fredde (sotto i -20 gradi) corrispondono mediamente valori di umidità che oscillano tra il 55% ed il 85%.

2.2.2 Silhouette

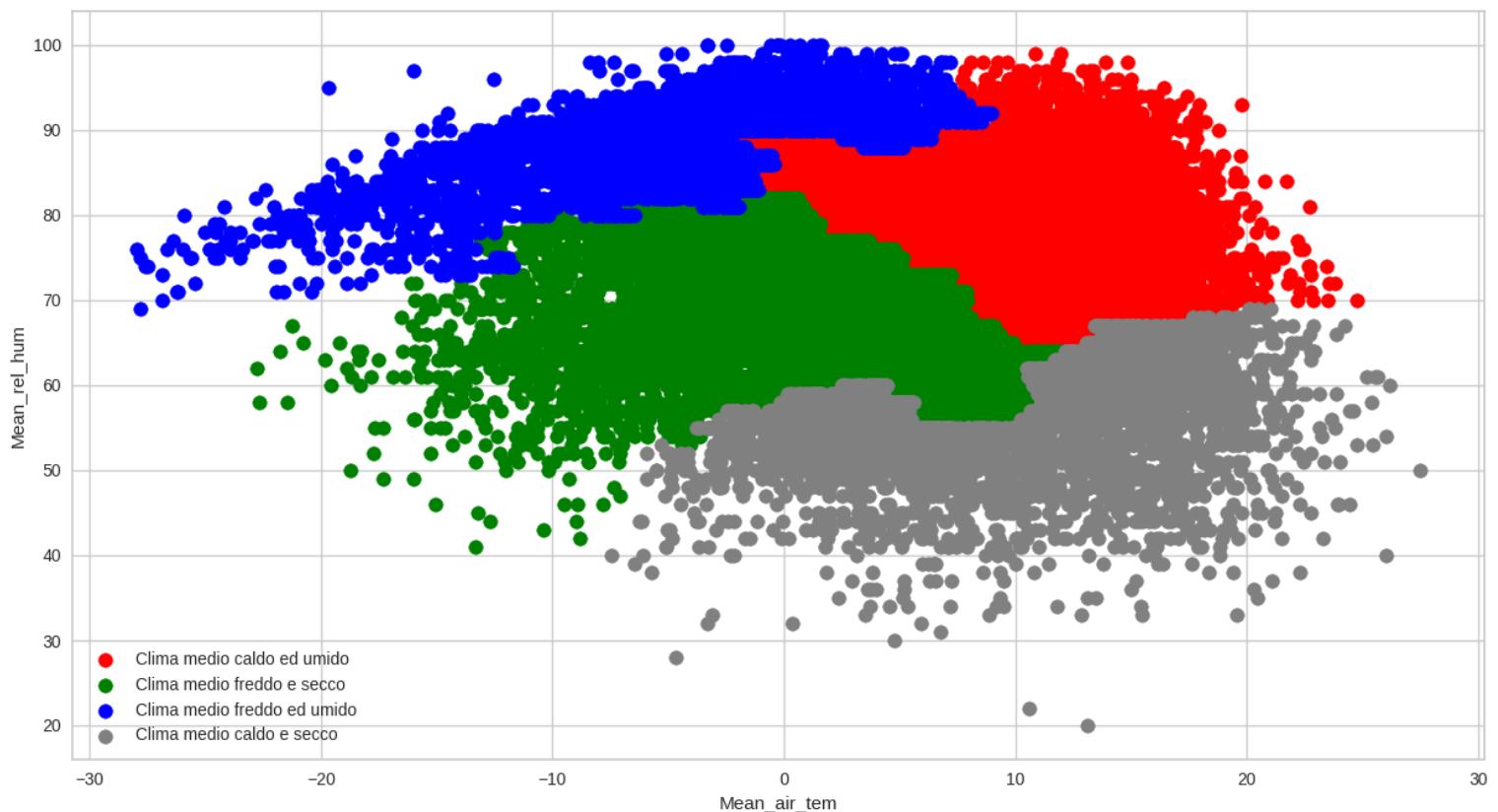
Di seguito è riportato il risultato della metrica Silhouette per determinare la validità del clustering effettuato.

Nonostante un valore medio della Silhouette non particolarmente alto (pari a 0.35) e inferiore al risultato del clustering dei valori massimi del max_air_temp e del max_rel_hum, possiamo comunque notare come gran parte dei valori superino il valor medio e come quasi non vi siano valore negativo. Pertanto possiamo accettare il risultato del clustering.



2.2.3 Hierarchical Agglomerative Clustering

Per validare ulteriormente il risultato, anche in questo caso abbiamo utilizzato l' hierarchical agglomerative clustering usando il risultato dell'elbow method utilizzato per il k-means clustering. Come per il clustering precedente, anche in questo caso abbiamo un risultato molto simile a quello del k-means clustering.

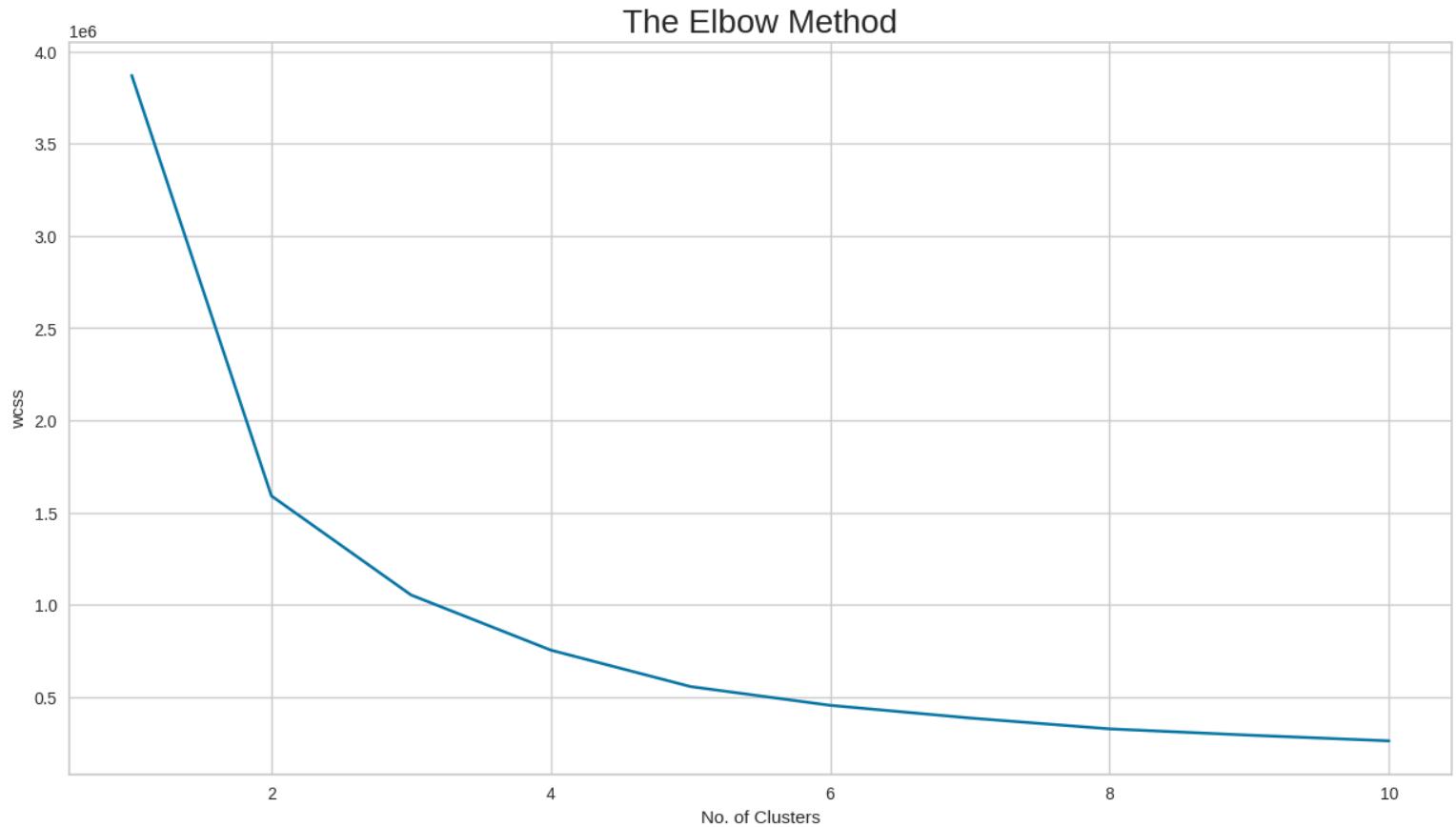


2.3 Clustering Sum_precip_amount e Mean_rel_hum

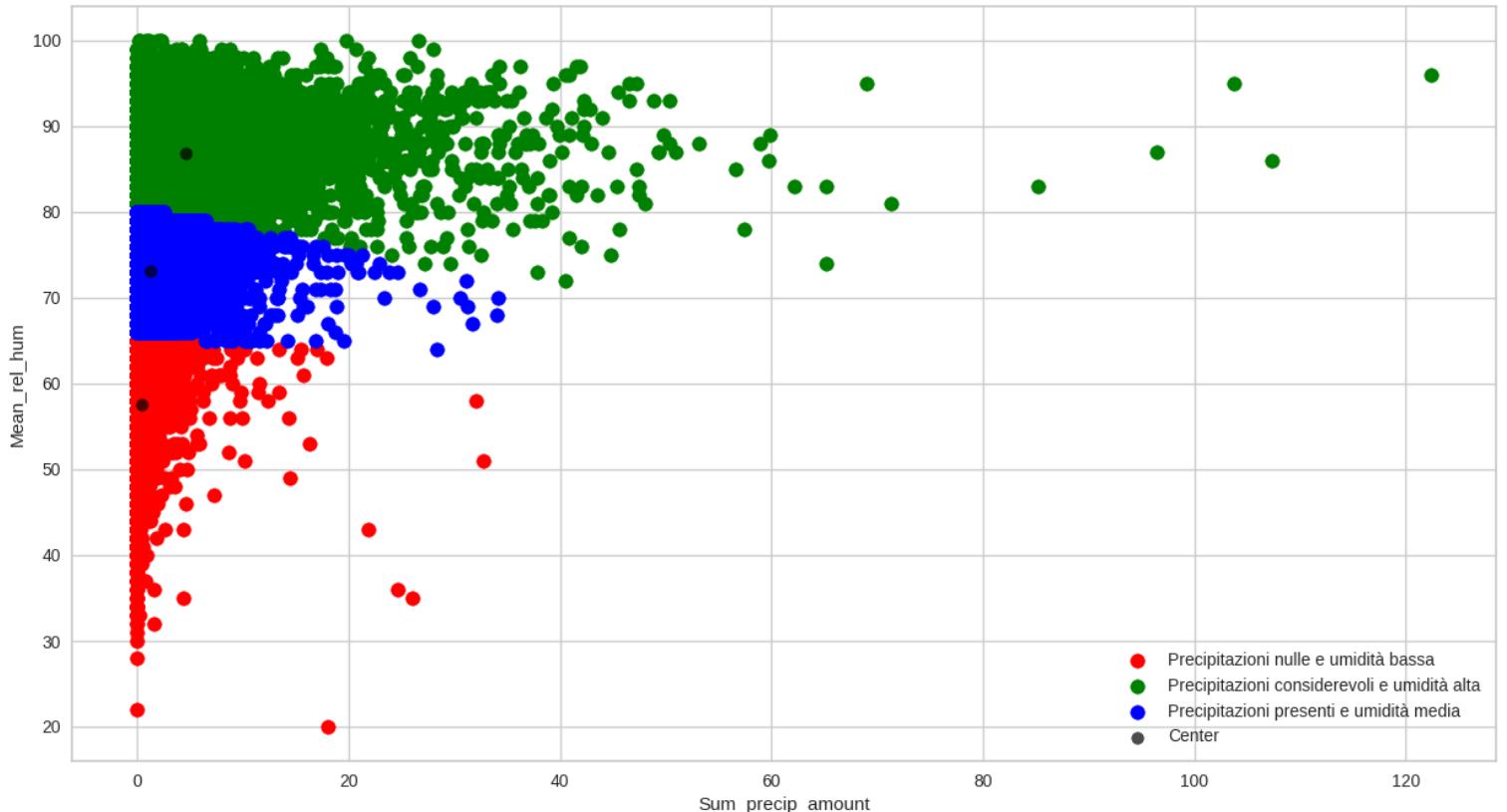
Per concludere il capitolo relativo al clustering abbiamo voluto analizzare i dati relativi alla quantità di precipitazioni per giorno e il valore medio dell'umidità relativa.

2.3.1 K-Means Clustering

Seguendo il procedimento usato nelle analisi precedenti abbiamo usato il K-Means come primo approccio per il clustering delle due variabili. Naturalmente è stato calcolato il numero di cluster da prendere in considerazione tramite l'elbow method con il quale è stato scelto un numero di cluster n=3.



Determinato il numero di cluster, è stato utilizzato l'algoritmo di clustering K-Means col seguente risultato:



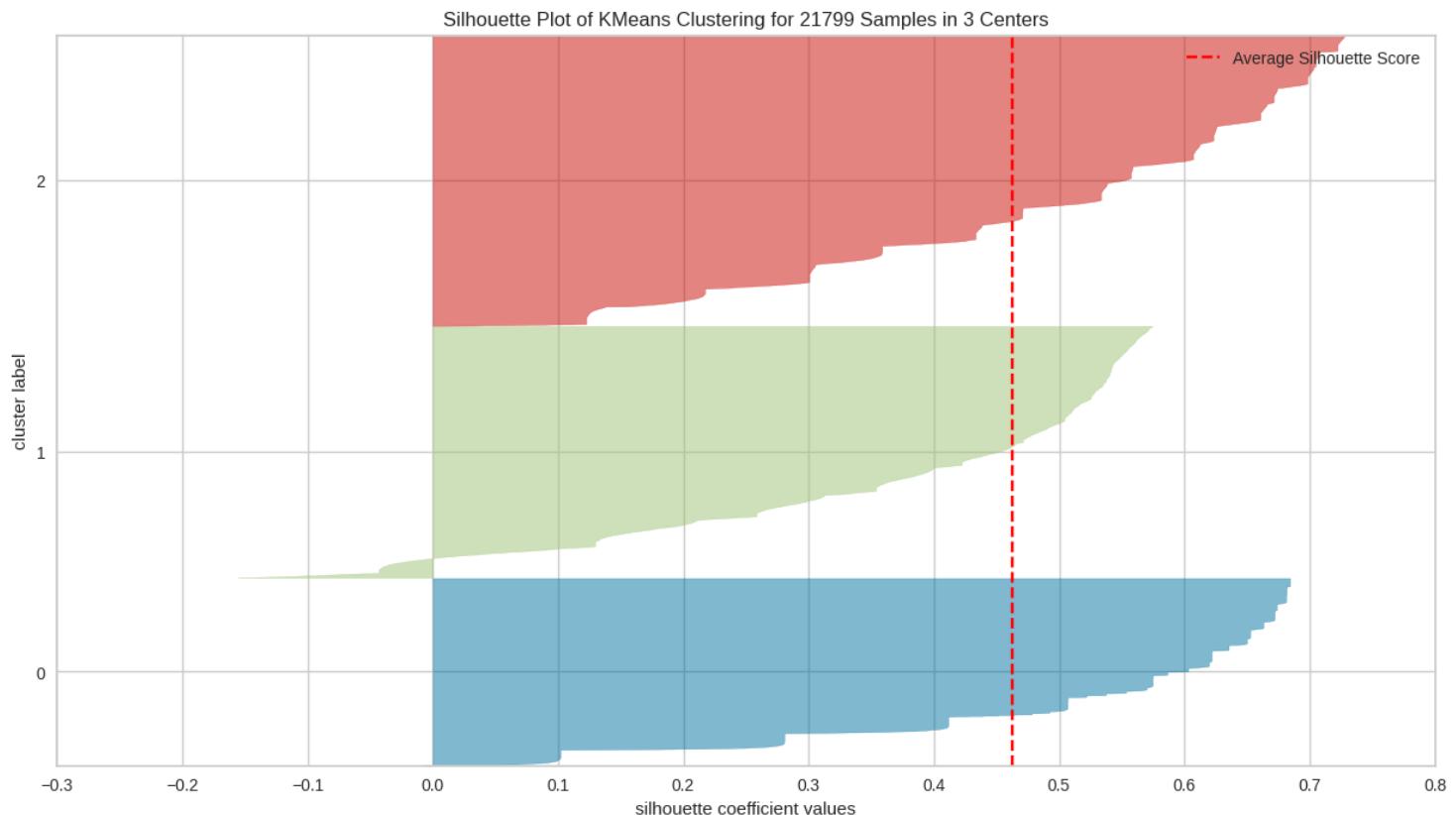
Da questa analisi si può immediatamente analizzare una correlazione tra l'aumento di precipitazione e l'aumento di umidità media per giorno: all'aumentare della quantità di precipitazioni durante il giorno corrisponde una sempre più alta percentuale di umidità media.

Inoltre si può altresì vedere come una percentuale alta di umidità può essere presente anche in presenza di basse precipitazioni o anche precipitazioni nulle.

2.3.2 Silhouette

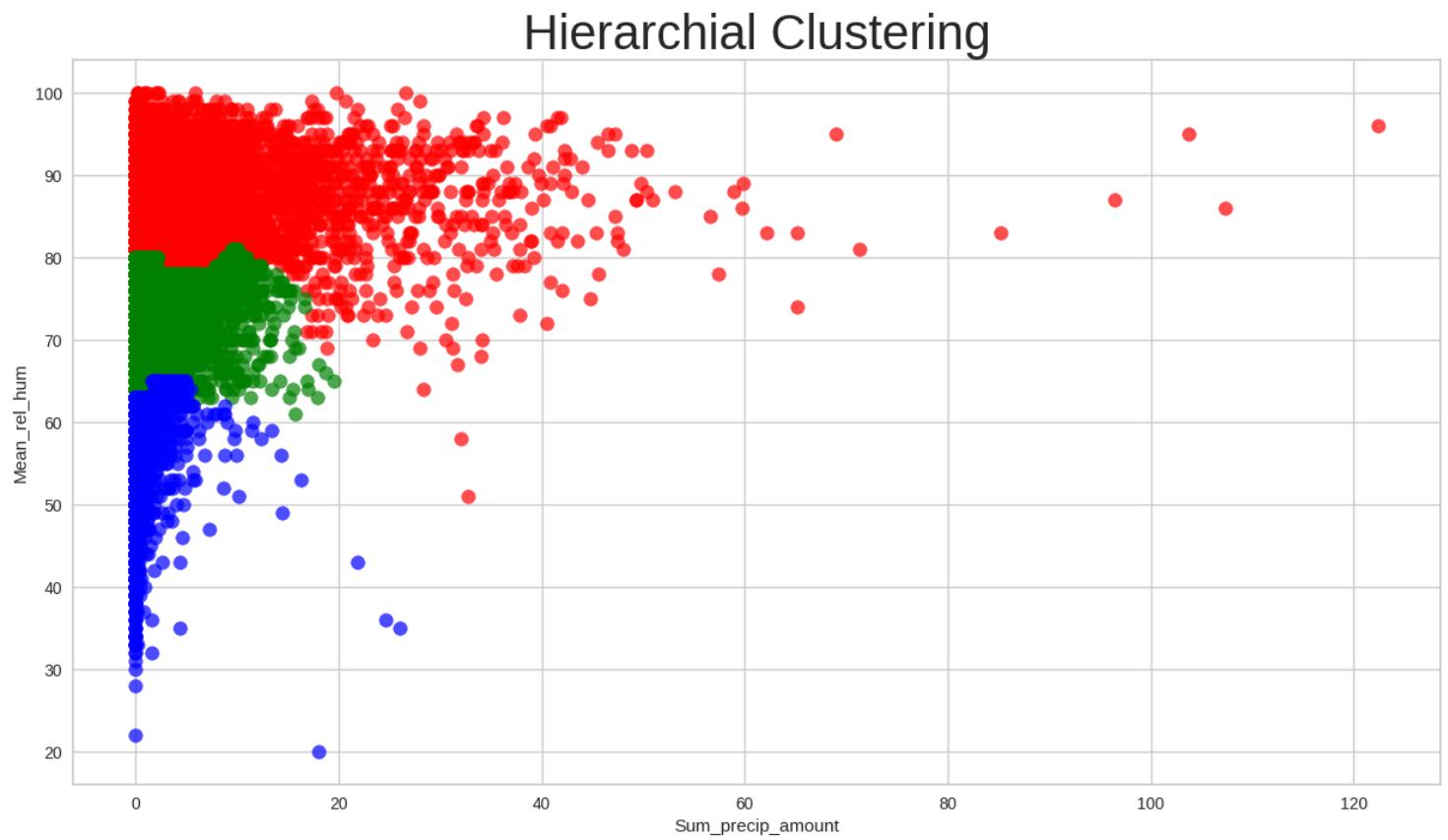
Come metrica per validare il risultato del clustering, anche in questo caso è stata usata la metrica Silhouette.

Come si può notare nel risultato qui sotto riportato, abbiamo un valore medio della Silhouette pari a 0.46 con una buona parte di valori oltre tale valore. Considerando inoltre una quasi assenza di valori negativi, possiamo considerare valido il clustering.



2.3.3 Hierarchical Agglomerative Clustering

Abbiamo confermato ulteriormente i nostri risultati attraverso l'utilizzo di hierarchical agglomerative clustering, utilizzando il criterio dell'elbow method precedentemente utilizzato per il k-means clustering. Il risultato ottenuto attraverso questa analisi è molto simile a quello ottenuto con il k-means clustering, rafforzando quindi la validità dei nostri risultati.



3. Serie Temporali

Nel secondo task è stata effettuata un'analisi delle serie temporali sempre sul primo dataset riguardante i dati metereologici di varie stazione meteo della Norvegia.

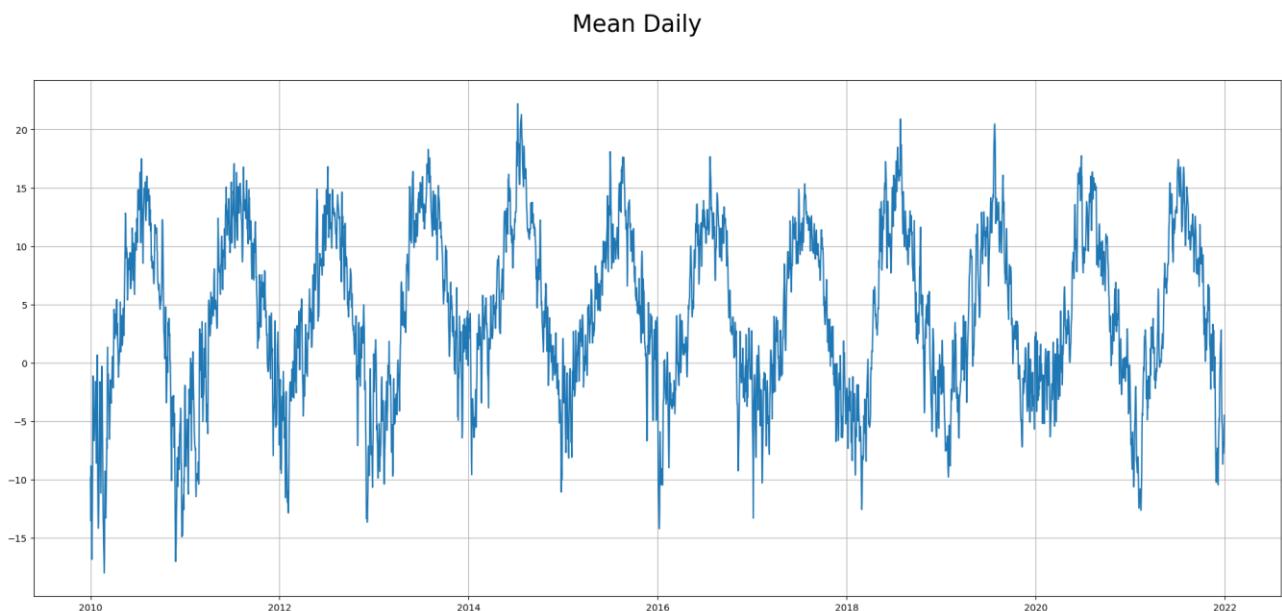
Si è deciso di considerare le temperature medie nel tempo per analizzare eventuali caratteristiche rilevanti della serie e fornire una predizione a breve termine dell'andamento.

3.1 Preprocessing serie temporale

Una volta eseguito il preprocessing iniziale di correzione e pulizia del dataset è stato considerato esclusivamente l'attributo “mean_air_temp_perday” relativo alla temperatura media giornaliera dell’ aria.

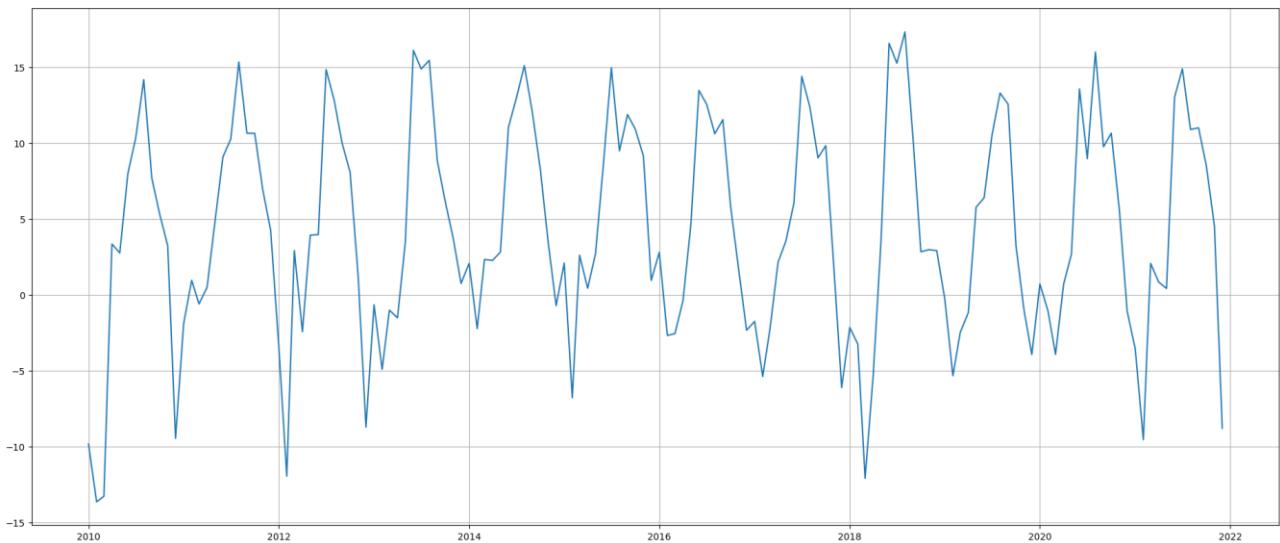
Poiché nel nostro dataset sono presenti i dati di 5 stazioni meteo sparse in tutta la Norvegia, sono presenti più temperature medie giornaliere per singolo giorno. Per ovviare a questa situazione, abbiamo calcolato la media delle varie temperature per singolo giorno così da avere la temperatura media giornaliera dell’intera Norvegia.

Di seguito è riportato l’andamento delle temperature giornaliere:



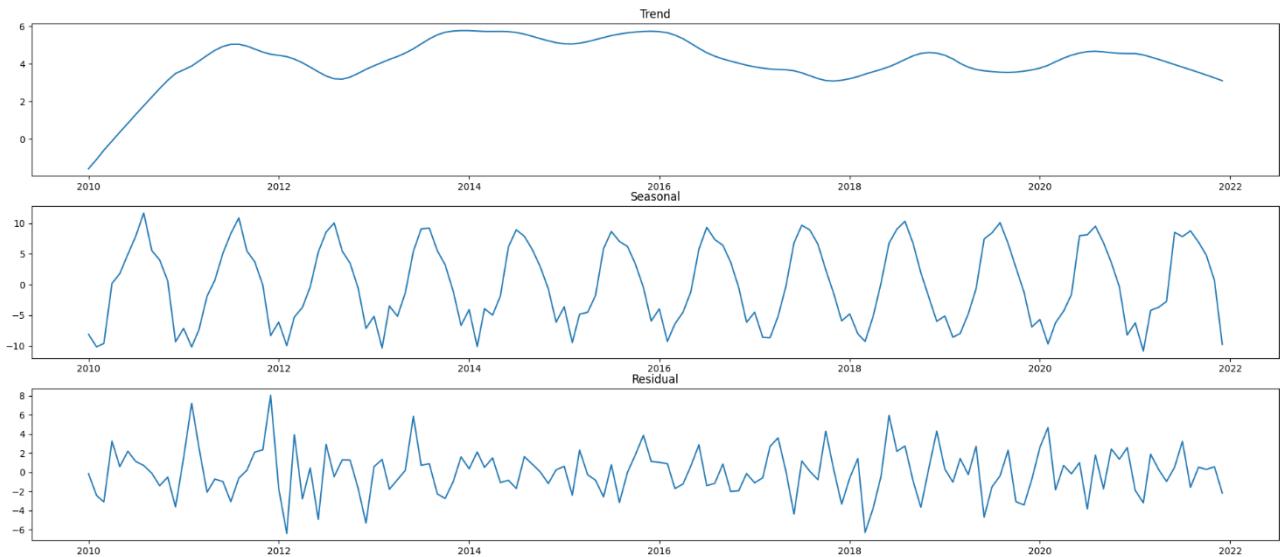
Poiché i dati giornalieri sono numerosi, si è riscontrata una complessità computazionale tale da dover prendere in considerazione non più le temperature medie giornaliere ma bensì quelle mensili calcolandole tramite la media delle varie temperature giornaliere dello stesso mese, con il seguente andamento:

Mean Monthly



Inoltre per una maggiore analisi della serie temporale, essa è stata anche decomposta in 3 componenti:

- Tendenza: La tendenza rappresenta la direzione generale dei dati nel tempo. Può essere crescente, decrescente o stabile. La tendenza può riflettere cambiamenti a lungo termine nelle serie temporali, come una crescita economica o una diminuzione delle vendite nel tempo. La presenza di una tendenza può indicare che la serie temporale non è stazionaria, poiché la media dei dati varia nel tempo.
- Stagionalità: La stagionalità si riferisce a fluttuazioni cicliche o periodiche nei dati che si ripetono regolarmente. Queste fluttuazioni possono essere legate a fattori stagionali come il tempo, i mesi dell'anno, le festività o eventi ricorrenti. La stagionalità può essere rappresentata da pattern che si ripetono nel corso di un anno o in periodi di tempo più brevi. La presenza di stagionalità può richiedere l'uso di modelli specifici come il modello SARIMA per catturare e modellare queste fluttuazioni periodiche.
- Rumore casuale: Il rumore casuale, anche noto come componente residuale, rappresenta le fluttuazioni irregolari e imprevedibili all'interno dei dati. Queste fluttuazioni possono essere causate da fattori casuali, errori di misurazione o altri fattori non spiegati dalla tendenza o dalla stagionalità. Il rumore casuale è considerato un termine di errore nel modello ed è generalmente considerato come un segnale non strutturato e casuale all'interno dei dati.



Da questa decomposizione della serie temporale possiamo subito notare la presenza di una stagionalità, così come ci si aspettava in quanto mediamente le temperature medie sono simili di anno in anno.

Data la presenza di stagionalità nella serie si è deciso di utilizzare come modello di predizione il SARIMA (Seasonal AutoRegressive Integrated Moving Average), una versione estesa del modello ARIMA (Autoregressive Integrated Moving Average) : questo modello è utile quando i dati presentano un pattern ricorrente in periodi di tempo regolari.

A tale modello sono associati dei parametri corrispondenti all'ordine delle varie componenti cioè quella auto regressiva con ordine p , quella di media mobile con ordine q e quella relativa alla differenziazione con ordine d . Questo sia per l'intera serie (p,q,d) che per la componente stagionale (P,Q,D,M), dove M è la lunghezza della stagionalità (nel nostro caso $M=12$).

3.2 Analisi della stazionarietà

Il primo componente da trovare è quello relativo all'ordine 'd' di differenziazione, ovvero il numero di differenziazioni necessarie per rendere stazionarie le serie temporali.

Da una prima analisi visiva sull'andamento del trend precedentemente visualizzato, si può già vedere come il grafico sia stazionario seguendo un andamento lineare.

Per studiare la stazionarietà in modo più preciso è stato utilizzato il metodo "Augmented Dickey Fuller test": un test statistico che valuta l'ipotesi nulla che la serie temporale sia non stazionaria.

Se il p-value del test è inferiore a una soglia di significatività predefinita (tipicamente 0,05), l'ipotesi nulla viene rigettata e si conclude che la serie temporale è stazionaria. Se il p-value è superiore alla soglia di significatività, l'ipotesi nulla non viene rigettata e si conclude che la serie temporale è non stazionaria.

Il risultato dell' Augmented Dickey Fuller test per la nostra serie temporale è il seguente:

ADF Statistic: -3.224902

p-value: 0.018588

Il risultato mostra un p-value minore di 0.05: questo risultato conferma la stazionarietà della serie temporale e non sarà necessario applicare una differenziazione.

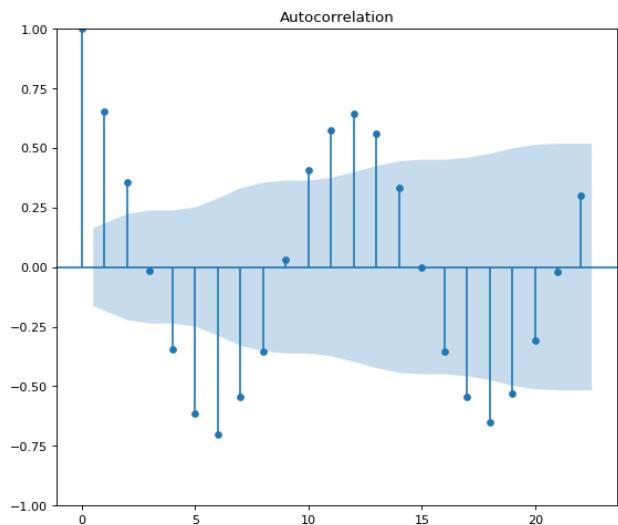
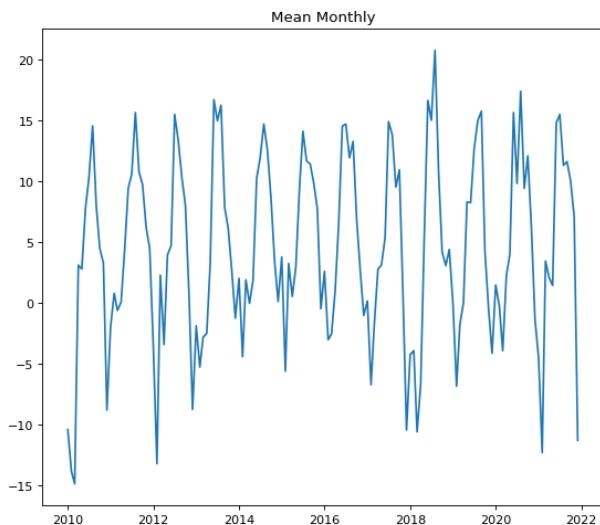
Alla luce di questi risultati si è scelto un valore dell'ordine di differenziazione d=0.

3.3 Stima parametro q

Il secondo componente da calcolare è l'ordine 'q' del termine di media mobile che rappresenta il numero di errori precedenti nella previsione che vengono utilizzati per predire il valore corrente.

Un valore q più alto indica che la previsione dipenderà da un numero maggiore di errori passati.

Per poter calcolare il valore del parametro q bisogna plottare l'autocorrelazione della serie temporale ed analizzare i lag al di fuori dell'area limite fino al primo lag che si trova dentro l'area.



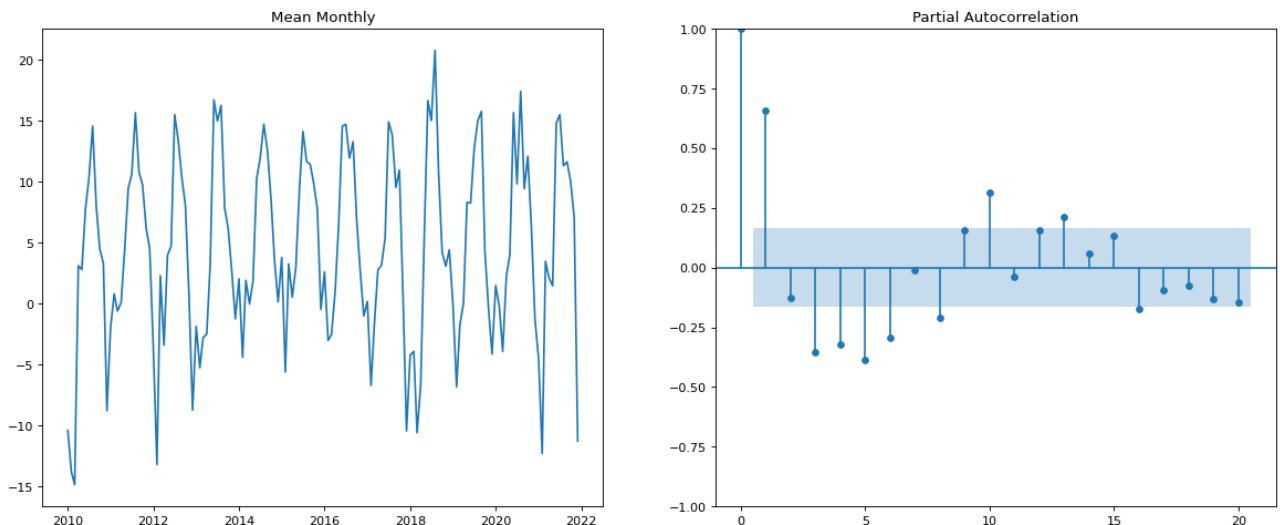
Dal risultato dell'autocorrelazione della serie temporale possiamo vedere come (escludendo sempre il lag(0) dall'analisi) i primi lag al di fuori dell'area limite sono il lag(1) ed il lag(2).

Da ciò possiamo concludere che il valore del parametro q sarà pari ad 1 o 2.

3.4 Stima parametro p

Ultimo componente da stimare è l'ordine 'p' del termine auto regressivo il quale indica il numero di valori precedenti nella serie temporale che vengono utilizzati per predire il valore corrente. Un valore p più alto indica che la previsione dipenderà da un numero maggiore di valori passati.

Per stimare il parametro p dobbiamo utilizzare la stessa procedura usata per stimare il parametro q ma non prendendo in considerazione l'autocorrelazione della serie temporale bensì l'autocorrelazione parziale.



Da questa analisi possiamo concludere che si debba prendere in considerazione solo il lag(1) per cui il valore del parametro p sarà pari a 1.

3.5 Stima parametri stagionalità (P, D, Q)

Ultimata la stima dei parametri (p,d,q) della serie temporale, per poter addestrare al meglio il modello SARIMA abbiamo bisogno di stimare anche i parametri (P,D,Q) della stagionalità della serie temporale.

- Parametro D

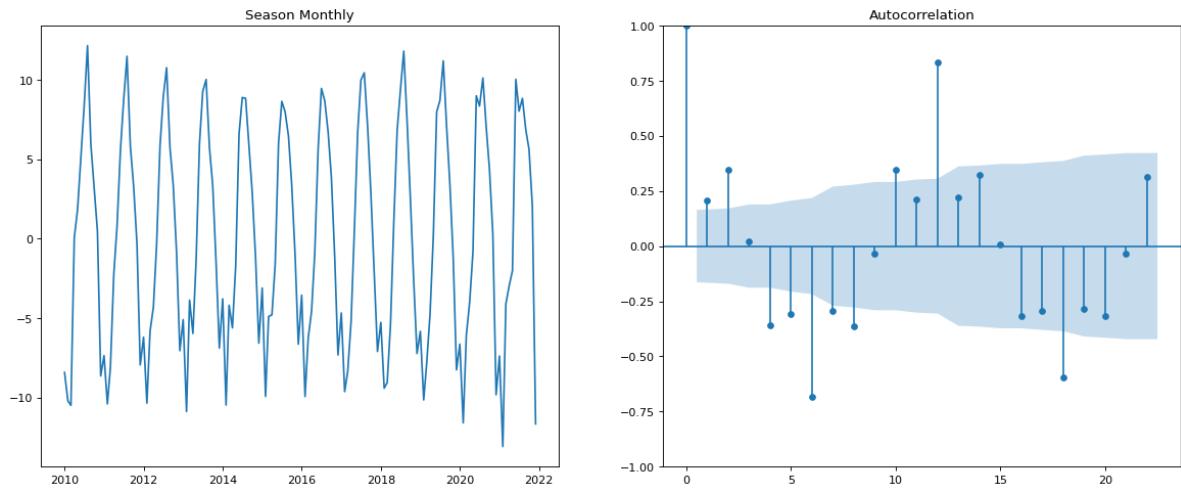
Il metodo “Augmented Dickey Fuller test” per il calcolo del parametro D della stagionalità ha dato il seguente risultato:

Statistic: -7.288619

p-value: 0.000000

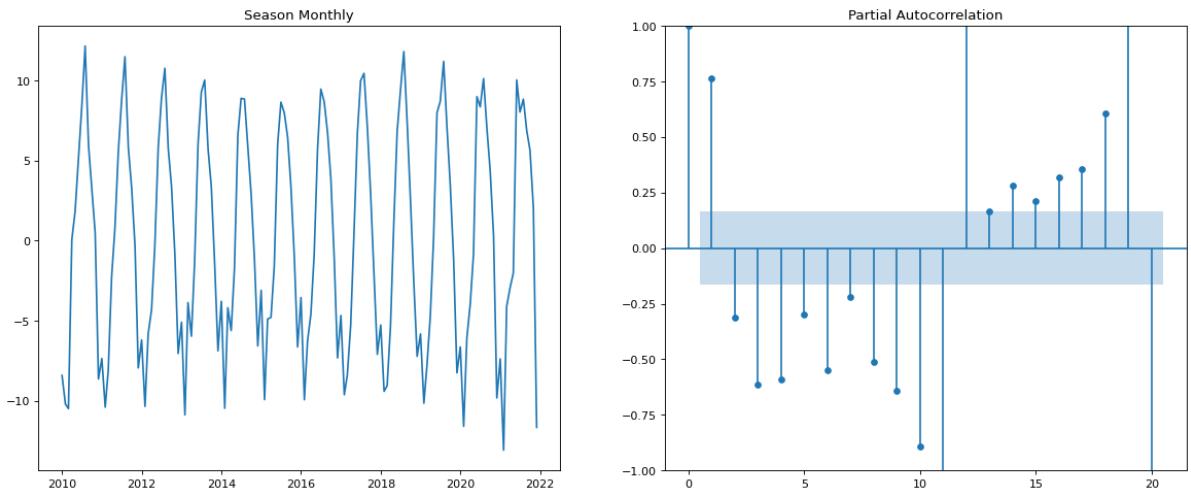
Possiamo concludere che anche la stagionalità della serie temporale è stazionaria, per cui abbiamo un valore D=0

- Parametro Q



L'analisi dell'autocorrelazione della stagionalità ha evidenziato come bisogni prendere in considerazione il lag(1) ed il lag(2), per cui il valore del parametro Q sarà pari ad 1 o 2.

- Parametro P



In questo caso abbiamo un'autocorrelazione parziale non del tutto affidabile in quanto praticamente tutti i lag sono al di fuori dell'area limite e ciò indica che non vi è alcuna autocorrelazione parziale significativa.

Ai fini del progetto, prenderemo in considerazione valori del parametro P in un range da 1 a 6.

3.6 Search Grid

Per avere una scelta ottimale nella combinazione delle componenti della serie temporale e le relative componenti della stagionalità, è stata adoperata una Search Grid: è una griglia di combinazioni di valori dei parametri del modello che vengono esplorate al fine di identificare la combinazione ottimale che fornisce i migliori risultati.

Vengono selezionate diverse combinazioni di valori per i parametri del modello e vengono valutate e confrontate utilizzando una metrica di valutazione appropriata, ossia l'errore quadratico medio (RMSE), tale metrica di valutazione è stata normalizzata, per poter avere una valutazione complessiva del modello allenato.

ARIMA Order	Seasonal Order	Metric Value
(1, 0, 1)	(1, 0, 0, 12)	0.22515665315990202
(1, 0, 1)	(2, 0, 0, 12)	0.15087350573839034
(1, 0, 1)	(3, 0, 0, 12)	0.14823718105326683
(1, 0, 1)	(4, 0, 0, 12)	0.14801320405218132
(1, 0, 1)	(5, 0, 0, 12)	0.13962558547242046
(1, 0, 1)	(6, 0, 0, 12)	0.142377272193972
(1, 0, 1)	(1, 0, 1, 12)	0.12909499249474418
(1, 0, 1)	(2, 0, 1, 12)	0.12884081090793538
(1, 0, 1)	(3, 0, 1, 12)	0.1203230024841622
(1, 0, 1)	(4, 0, 1, 12)	0.12445904690753617
(1, 0, 1)	(5, 0, 1, 12)	0.12313583298880257
(1, 0, 1)	(6, 0, 1, 12)	0.12575112890276045
(1, 0, 2)	(1, 0, 0, 12)	0.22265333283768567
(1, 0, 2)	(2, 0, 0, 12)	0.1527600801264048
(1, 0, 2)	(3, 0, 0, 12)	0.14767905272561704
(1, 0, 2)	(4, 0, 0, 12)	0.1514549904116326
(1, 0, 2)	(5, 0, 0, 12)	0.14282137076777195
(1, 0, 2)	(6, 0, 0, 12)	0.14340229785140028
(1, 0, 2)	(1, 0, 1, 12)	0.1325979704921517
(1, 0, 2)	(2, 0, 1, 12)	0.12933649333451932
(1, 0, 2)	(3, 0, 1, 12)	0.12855928090628816
(1, 0, 2)	(4, 0, 1, 12)	0.12749188474846475
(1, 0, 2)	(5, 0, 1, 12)	0.12581092832400133
(1, 0, 2)	(6, 0, 1, 12)	0.12331542111990966
(1, 0, 3)	(1, 0, 0, 12)	0.22649757815460172
(1, 0, 3)	(2, 0, 0, 12)	0.15120110767130857

ARIMA Order	Seasonal Order	Metric Value
(1, 0, 3)	(3, 0, 0, 12)	0.14640853151160836
(1, 0, 3)	(4, 0, 0, 12)	0.150178306037346
(1, 0, 3)	(5, 0, 0, 12)	0.1397259654238116
(1, 0, 3)	(6, 0, 0, 12)	0.1423410952635715
(1, 0, 3)	(1, 0, 1, 12)	0.1325147882543342
(1, 0, 3)	(2, 0, 1, 12)	0.1290343319268425
(1, 0, 3)	(3, 0, 1, 12)	0.1285117778767398
(1, 0, 3)	(4, 0, 1, 12)	0.12774851482673893
(1, 0, 3)	(5, 0, 1, 12)	0.1259383260152187
(1, 0, 3)	(6, 0, 1, 12)	0.1232378539337854

Dalla valutazione del RMSE delle varie combinazioni di (p,d,q) e (P,D,Q,M) , si può concludere che la miglior combinazione di parametri è $(1,0,1)/(3, 0, 1, 12)$ con un Normalized-RMSE pari a 0.1203230024841622.

3.7 Sviluppo di un modello della serie

A questo punto, avendo scelto i parametri, è stato possibile applicare il modello SARIMA per tentare di riprodurre l'andamento della serie.

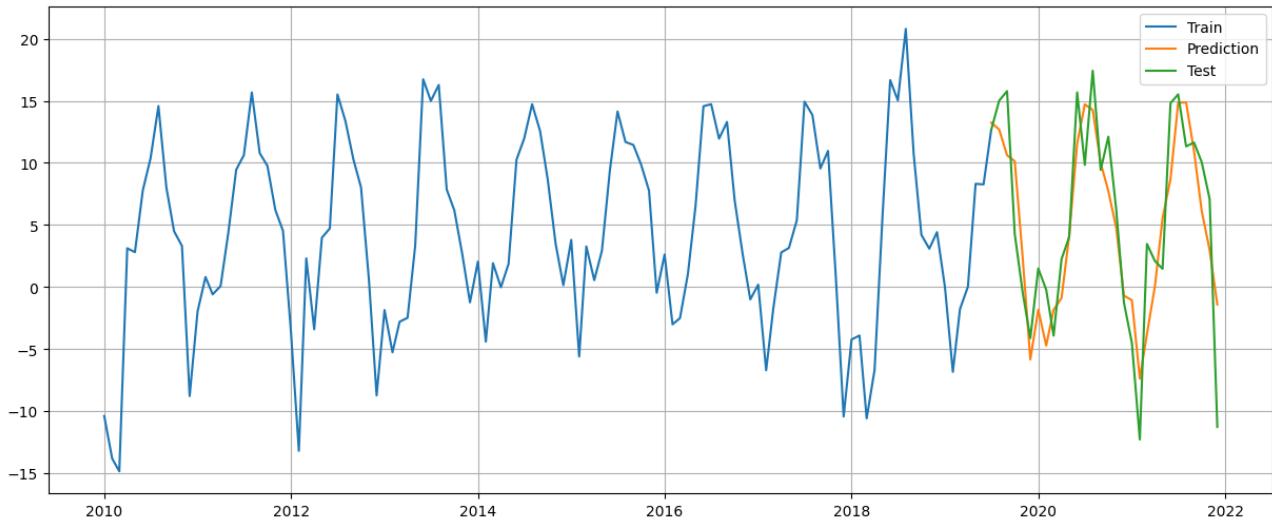
Il comando generale utilizzato è SARIMAX, ma non inserendo variabili esogene, si riduce a un modello SARIMA:

SARIMAX Results						
Dep. Variable:	Temperatury	No. Observations:	144			
Model:	SARIMAX(1, 0, 1)x(3, 0, 1, 12)	Log Likelihood	-408.970			
Date:	Thu, 01 Jun 2023	AIC	831.940			
Time:	09:24:29	BIC	852.728			
Sample:	01-01-2010 - 12-01-2021	HQIC	840.387			
Covariance Type:	opg					
	coef	std err	z	P> z	[0.025	0.975]
ar.L1	0.5397	0.122	4.409	0.000	0.300	0.780
ma.L1	-0.2950	0.078	-3.794	0.000	-0.447	-0.143
ar.S.L12	0.7637	0.086	8.854	0.000	0.595	0.933
ar.S.L24	0.3096	0.088	3.516	0.000	0.137	0.482
ar.S.L36	-0.0735	0.065	-1.129	0.259	-0.201	0.054
ma.S.L12	-0.9626	0.287	-3.350	0.001	-1.526	-0.399
sigma2	12.7914	3.491	3.664	0.000	5.950	19.633
Ljung-Box (L1) (Q):	0.50	Jarque-Bera (JB):		2.38		
Prob(Q):	0.48	Prob(JB):		0.30		
Heteroskedasticity (H):	0.99	Skew:		-0.31		
Prob(H) (two-sided):	0.97	Kurtosis:		3.07		

Analizzando le statistiche sul modello si nota come la statistica $P>|z|$ per ogni ordine delle componenti di autoregressione e media mobile risulti sempre al di sotto della soglia di significatività di 0.05, per cui l'ordine delle componenti del modello risultano avere tutte un contributo molto significativo e quindi necessarie.

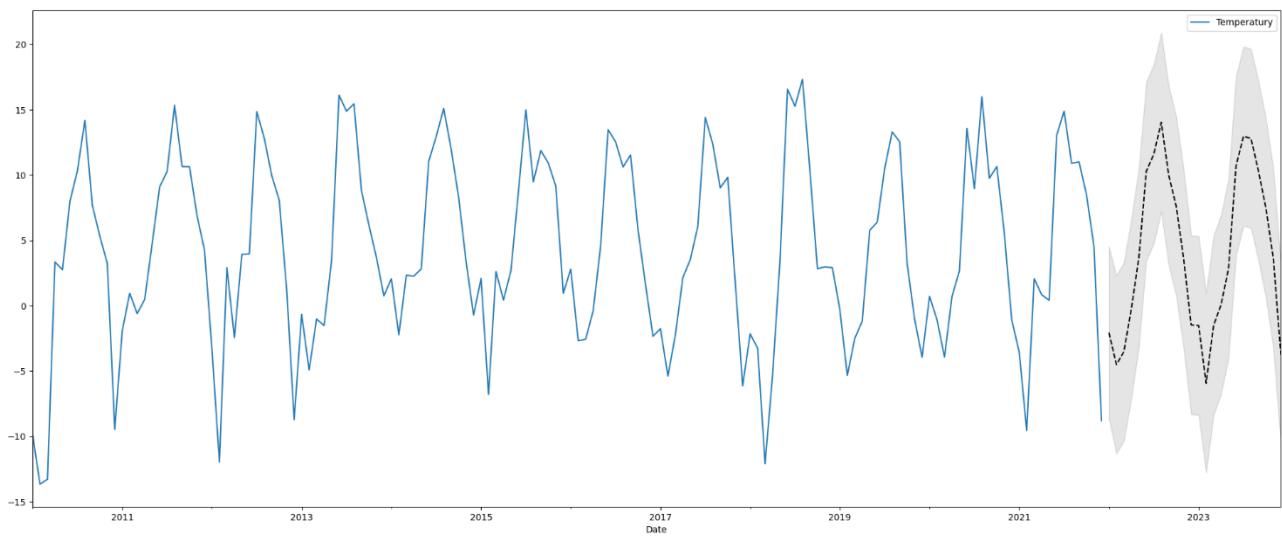
3.8 Predizione & forecasting

Il modello SARIMA utilizzato è definito quindi dai parametri $(p,d,q)=(1,0,1)$ e $(P,D,Q,M)=(3,0,1,12)$. Si è suddivisa la serie in training e testing set tramite una split con percentuali rispettivamente di 80% e 20% e si è effettuata la predizione dell'andamento temporale del testing set ottenendo il seguente risultato:



Si può notare una predizione abbastanza precisa in quanto il grafico creato dal modello segue in modo alquanto fedele l'andamento reale delle temperature.

A questo punto è stato fatto un forecasting dell'andamento temporale delle temperature partendo dall'anno 2022 con il seguente risultato:



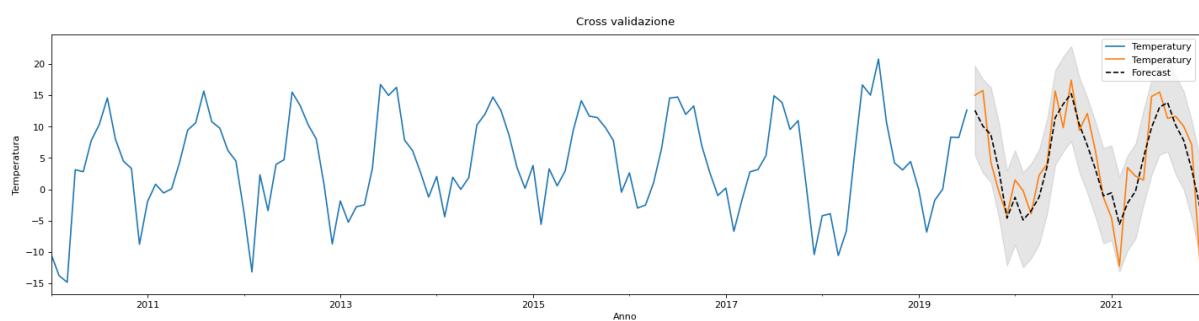
Dal grafico si può vedere come la previsione sia abbastanza accurata, dal momento che rispecchia l'oscillazione dovuta alla stagionalità.

3.9 Valutazione affidabilità del modello

Per valutare il modello SARIMA addestrato sono state calcolate alcune metriche per avere una valutazione quantitativa delle performance:

- **Cross-Validazione**

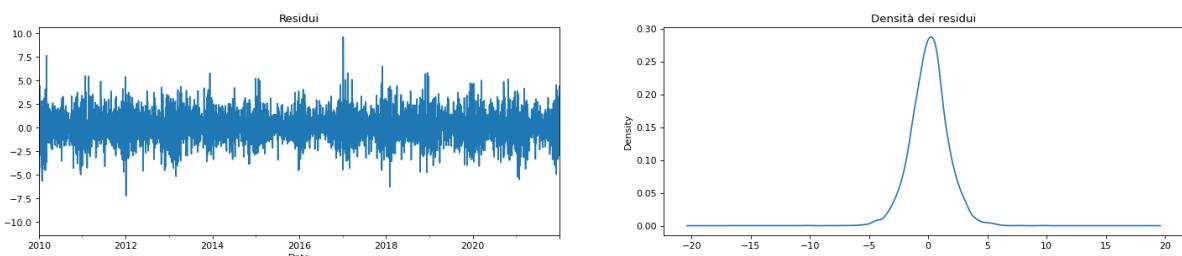
Nella Cross-Validation, si fanno alcuni passi indietro nel tempo e si prevede il valore predetto facendo un confronto con il valore reale. Anche in questa sezione verrà ripetuto l'addestramento del modello utilizzando l'80% dei dati. Il restante 20% viene utilizzato per la fase di test. La previsione deve aderire il più possibile al valore reale.



Si può vedere come la previsione rappresenti abbastanza fedelmente l'effettivo andamento. Le uniche differenze sono relative ai due picchi della metà del 2019 e del 2021.

- **Residui**

Sono stati calcolati i residui per assicurarci che non ci siano pattern particolari, e che abbiano media e varianza costante. In figura, dall'andamento temporale è possibile notare come la media dei residui sia nulla in quanto il grafico oscilla intorno allo zero, mentre dal grafico KDE a destra, riportante la distribuzione dei residui, si nota come i valori siano distribuiti nel range [-5,5], quindi con una varianza costante e relativamente bassa.



- **Metriche di precisione**

Si è infine valutato il modello utilizzando più metriche di precisione, con i seguenti risultati:

MAPE: 1.6037160860591977

ME: -0.6518011508694209

MAE: 3.3070691727762895

MPE: 0.23220672426443284

NRMSE: 0.1203230024841622

CORR: 0.8836854779614668

MINMAX: -0.749842960716361

ACF1: -0.14679097615965667

Si nota subito come dal calcolo delle metriche di precisione si possa concludere che il modello ricavato sia abbastanza accurato. Ciò si evince dal valore del MAPE pari all' 1,60%, col quale si ottiene che il modello è al 98,40% accurato nella predizione, e dal basso valore del Normalized-RMSE pari al 0.12.

4. Classificazione

L'ultimo task è riservato alla classificazione eseguita sul secondo dataset, ovvero quello riguardante i funghi.

L'obiettivo di questo lavoro è addestrare un classificatore che sia capace di classificare un fungo in velenoso o edibile in base alle sue caratteristiche fisiche e al luogo dove si possono trovare.

4.1 Preprocessing Classificazione

Come analizzato precedentemente, abbiamo a disposizione un dataset comprendente 8124 tipi diversi di funghi, ognuno dei quali è stato etichettato in base alla sua velenosità o non velenosità e descritto tramite 22 attributi categorici.

Come primo step è stato effettuato un pre processamento dei dati a disposizione:

1. sono stati convertiti tutti gli attributi da categorici a numerici per facilitare il lavoro di classificazione:

```
df['poisonous'] = df['poisonous'].apply(lambda x: 1 if x == 'POISONOUS' else 0)

mapping = {'CONVEX': 0, 'FLAT': 1, 'BELL': 2, 'SUNKEN': 3, 'KNOBBED': 4, 'CONICAL': 5}
df['cap-shape'] = df['cap-shape'].map(mapping)

mapping = {'SMOOTH': 0, 'FIBROUS': 1, 'SCALY': 2, 'GROOVES': 3}
df['cap-surface'] = df['cap-surface'].map(mapping)

mapping = {'WHITE': 0, 'YELLOW': 1, 'BROWN': 2, 'GRAY': 3, 'RED': 4, 'PINK': 5, 'PURPLE': 6, 'GREEN': 7, 'BUFF': 8, 'CINNAMON': 9}
df['cap-color'] = df['cap-color'].map(mapping)

mapping = {'BRUISES': 1, 'NO': 0}
df['bruises?'] = df['bruises?'].map(mapping)

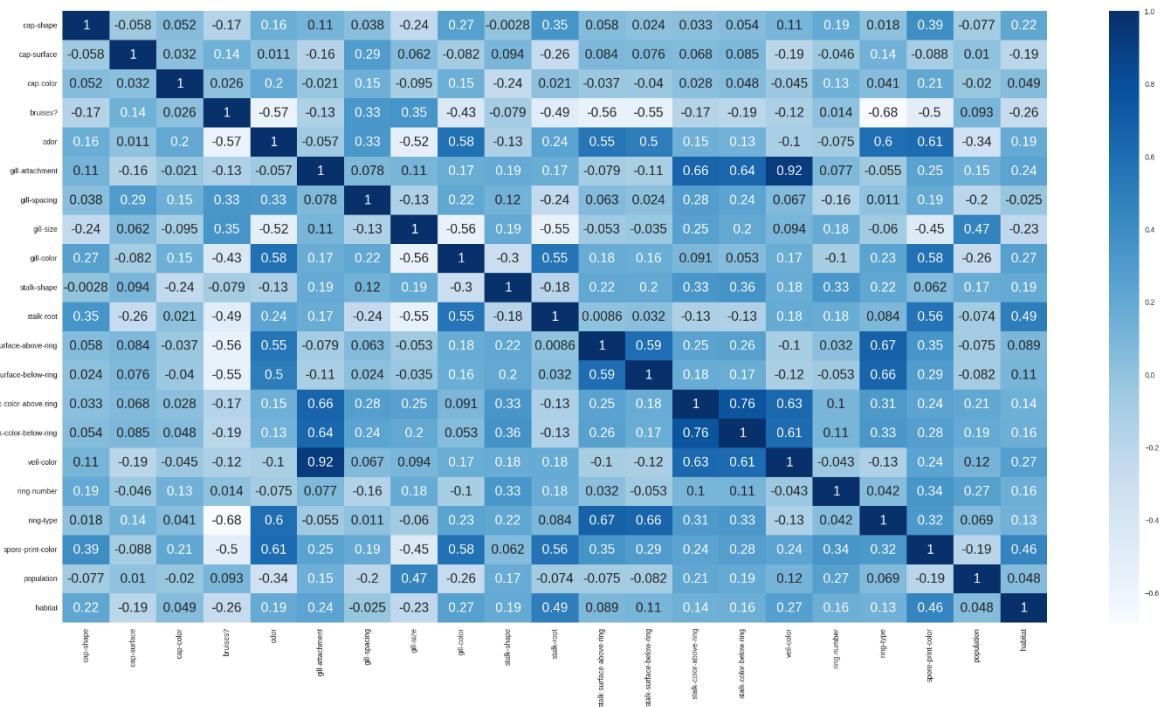
mapping = {'ALMOND': 0, 'ANISE': 1, 'NONE': 2, 'PUNGENT': 3, 'CREOSOTE': 4, 'FOUL': 5, 'FISHY': 6, 'SPICY': 7, 'MUSTY': 8}
df['odor'] = df['odor'].map(mapping)
```

Con questa modifica ogni attributo del fungo è descritto da valori numerici che possono variare da 0 a 11:

```
poisonous [0 1]
cap-shape [0 1 2 3 4 5]
cap-surface [0 1 2 3]
cap-color [0 1 2 3 4 5 6 7 8 9]
bruises? [1 0]
odor [0 1 2 3 4 5 6 7 8]
gill-attachment [0 1]
gill-spacing [0 1]
gill-size [0 1]
gill-color [0 1 2 3 4 5 6 7 8 9 10 11]
stalk-shape [0 1]
stalk-root [0 1 2 3 4]
stalk-surface-above-ring [0 1 2 3]
stalk-surface-below-ring [0 1 2 3]
stalk-color-above-ring [0 1 2 3 4 5 6 7 8]
stalk-color-below-ring [0 1 2 3 4 5 6 7 8]
veil-color [0 1 2 3]
ring-number [0 1 2]
ring-type [0 1 2 3 4]
spore-print-color [0 1 2 3 4 5 6 7 8]
population [0 1 2 3 4 5]
habitat [0 1 2 3 4 5 6]
```

Tramite questa variazione di parametri si è anche reso superfluo eseguire un processo di *One Hot Encoding*, un metodo per codificare variabili categoriche in modo da renderle adatte all'elaborazione da parte degli algoritmi di apprendimento automatico.

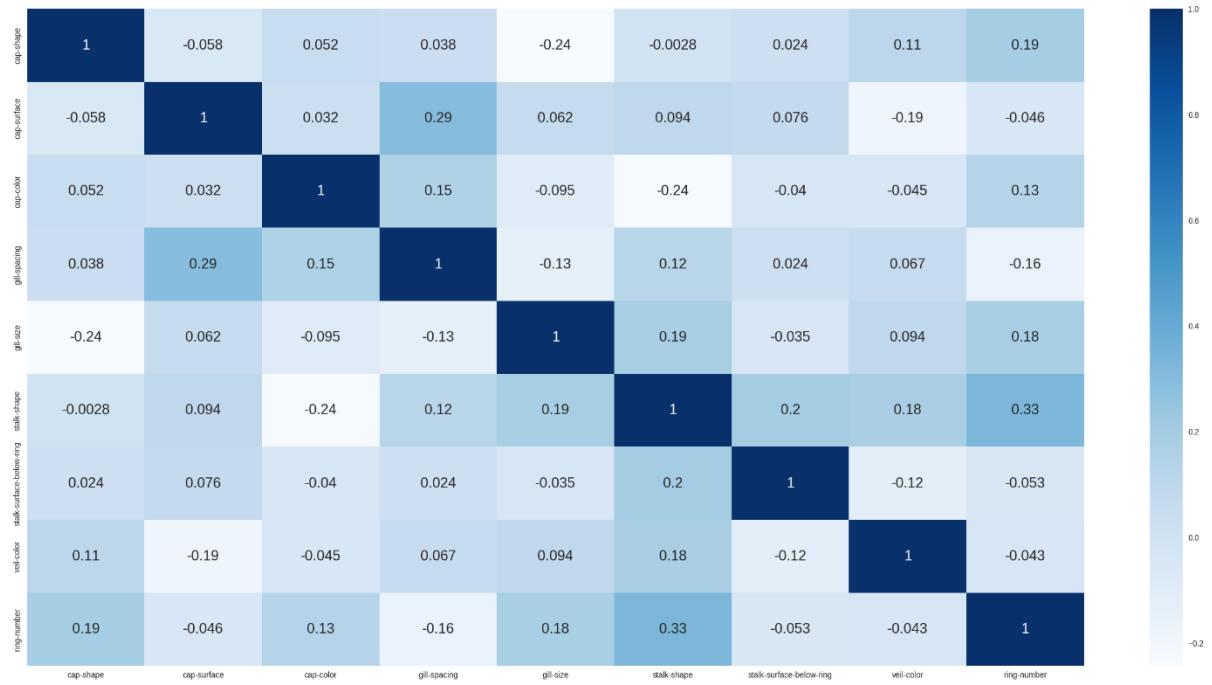
- è stata creata una matrice di correlazione tra i vari attributi per analizzare la connessione tra di essi:



Da questa matrice si possono notare valori elevati di correlazione che arrivano fino a 0.92.

Per azzerare qualsiasi possibilità di overfitting dal processo di classificazione, sono stati eliminati tutti gli attributi che hanno registrato un valore di correlazione maggiore dello 0.5. Questa operazione ha portato all'eliminazione di 12 attributi, mantenendone in considerazione solo 9 (con un valore di correlazione massimo di 0.33):

- Cap-shape
- Cap-surface
- Cap-color
- Gill-spacing
- Gill-size
- Stalk-shape
- Stalk-surface-below-ring
- Veil-color
- Ring-number



3. il dataset pre processato è stato diviso in 2 dataset distinti con relavite label:

```
# Split the data to check which algorithms learn better (later on we can check )
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, stratify=y, random_state=rs)

# look at the shape of the data (many problems can arise from wrong shape)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)

Index(['cap-shape', 'cap-surface', 'cap-color', 'gill-spacing', 'gill-size',
       'stalk-shape', 'stalk-surface-below-ring', 'veil-color', 'ring-number'],
      dtype='object')
(6732, 9)
(1684, 9)
(6732,)
(1684,)
```

- **x_train**: dataset di training per addestrare i vari classificatori pari all' 80% del dataset
- **y_train**: vettore dell'attributo 'Poisonous' riferito al dataset di training
- **x_test** : dataset di testing per testare i classificatori pari al 20% del dataset
- **y_test** : vettore dell'attributo 'Poisonous' riferito al dataset di testing

4.2 Training e testing dei classificatori

Finita la fase di pre processamento del dataset, si è passato all’addestramento e testing di vari classificatori con i rispettivi parametri:

```
# List of classifiers:  
classifiers = [  
    LogisticRegression(random_state = rs),  
    DecisionTreeClassifier(max_depth=8, random_state=rs),  
    LinearSVC(random_state=rs),  
    RandomForestClassifier(n_estimators = 10, random_state=rs),  
    MLPClassifier(hidden_layer_sizes=(100,), max_iter=300, random_state=rs),  
    KNeighborsClassifier(),  
    GaussianNB(),  
    LinearDiscriminantAnalysis()  
]
```

Ogni algoritmo di classificazione è stato addestrato tramite il training dataset `x_train` e il rispettivo vettore delle label `y_train` e successivamente è stato testato attraverso il testing dataset `x_test` ed il vettore delle label `y_test`:

```
# Training the algorithms and results  
for clf in classifiers:  
    name = clf.__class__.__name__  
    clf_name.append(name)  
  
    #fitting and predictions  
    model = clf.fit(x_train, y_train)  
    y_pred = model.predict(x_test)  
    model_results[name] = y_pred  
  
    #accuracy and log loss  
    cv_results.append(cross_val_score(clf, x_train, y_train, scoring = "accuracy",cv = kfold))  
    acc = round(accuracy_score(y_test, y_pred), 2) #need to maximize  
    # train_pred = clf.predict_proba(x_test)|  
    print(f'Accuracy: {acc} \t ---> {name} ')
```

Come parametro di valutazione del testing dei vari classificatori è stata calcolata la rispettiva accuracy con i seguenti risultati:

```
Accuracy: 0.94    ---> LogisticRegression  
Accuracy: 0.99    ---> DecisionTreeClassifier  
Accuracy: 0.94    ---> LinearSVC  
Accuracy: 0.99    ---> RandomForestClassifier  
Accuracy: 1.0     ---> MLPClassifier  
Accuracy: 0.99    ---> KNeighborsClassifier  
Accuracy: 0.65    ---> GaussianNB  
Accuracy: 0.94    ---> LinearDiscriminantAnalysis
```

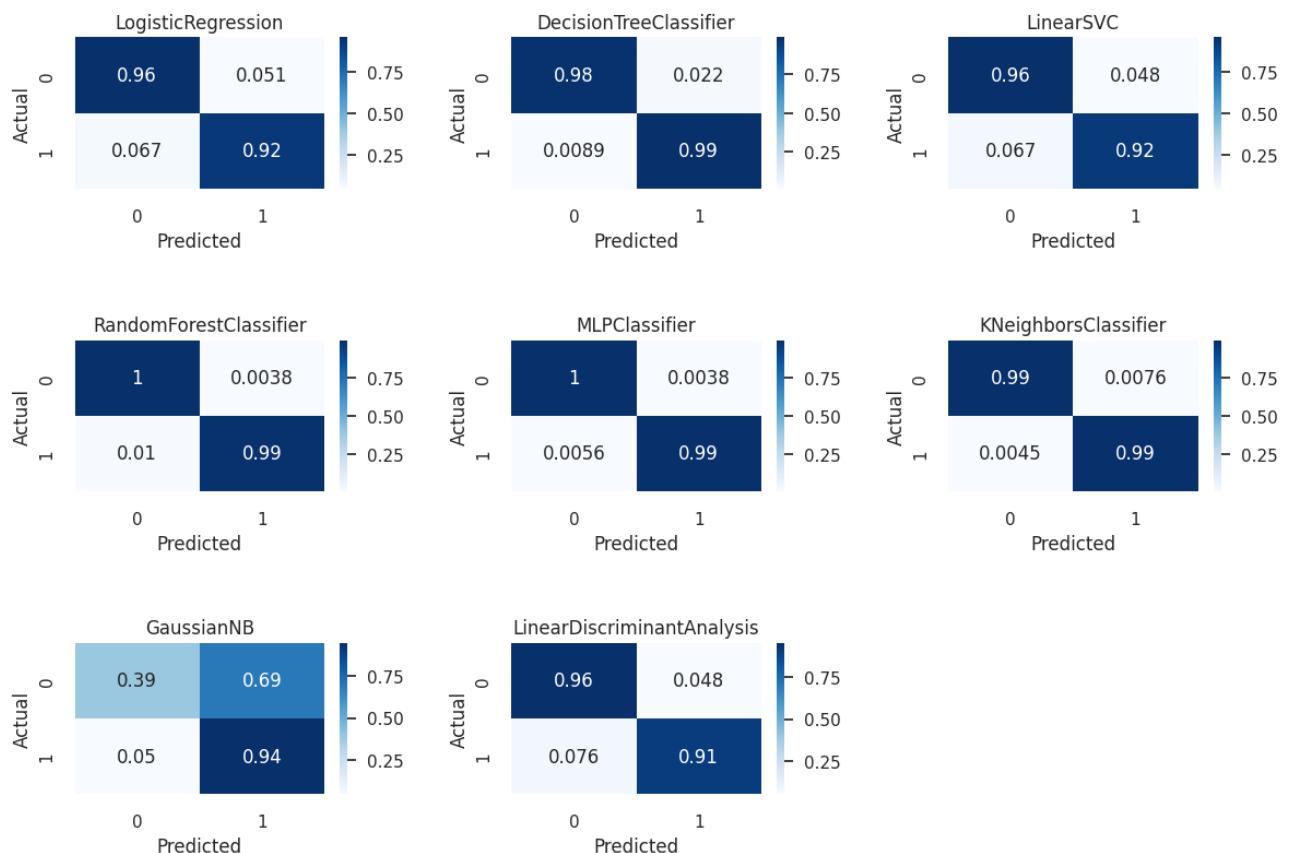
4.3 Valutazione dei Classificatori

Per valutare correttamente i vari algoritmi utilizzati abbiamo innanzitutto analizzato le matrici di confusione: esse sono strumenti utilizzati per visualizzare le prestazioni di un modello di classificazione in modo più dettagliato. Sono rappresentate da una tabella che mostra il numero di previsioni fatte dal modello per ciascuna classe target e la corrispondente etichetta reale.

Le matrici di confusione sono basate sui quattro possibili risultati di una classificazione binaria:

1. True Positive (TP): Rappresenta i casi in cui il modello ha correttamente previsto una classe positiva.
2. True Negative (TN): Rappresenta i casi in cui il modello ha correttamente previsto una classe negativa.
3. False Positive (FP): Rappresenta i casi in cui il modello ha erroneamente previsto una classe positiva quando in realtà era negativa (errore di tipo I).
4. False Negative (FN): Rappresenta i casi in cui il modello ha erroneamente previsto una classe negativa quando in realtà era positiva (errore di tipo II).

La matrice di confusione organizza questi risultati in una tabella, dove le righe rappresentano la classe reale e le colonne rappresentano la classe predetta dal modello.

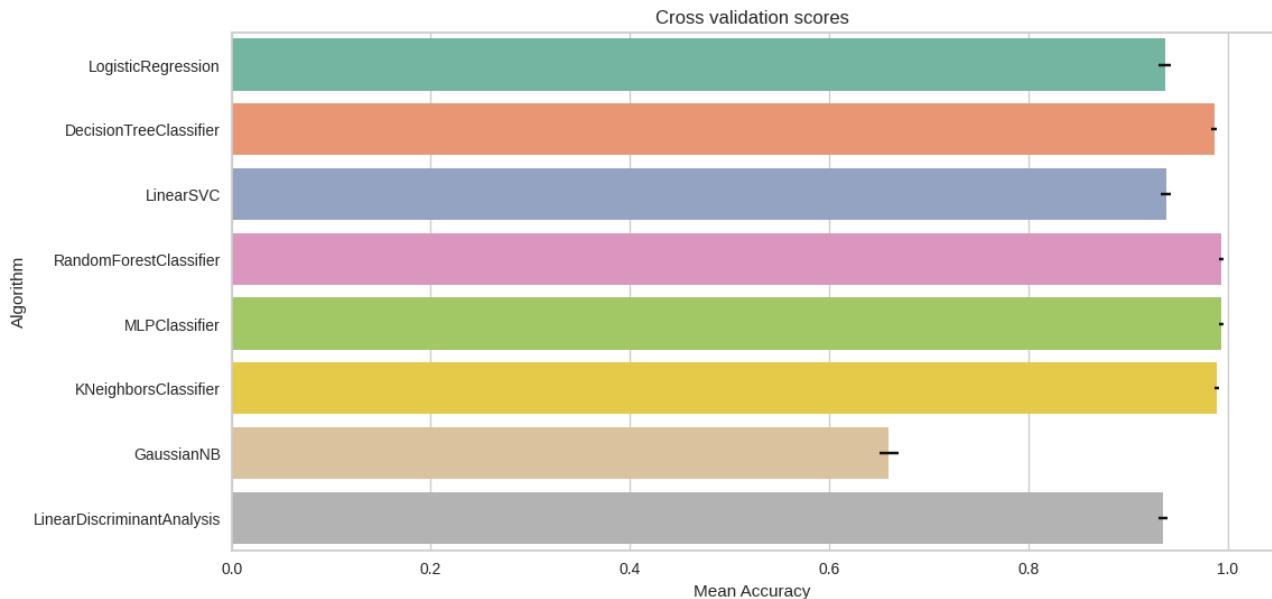


Le matrici di confusione inoltre forniscono informazioni dettagliate sulle performance del modello, consentendo di calcolare ulteriori metriche:

- Accuracy: L'accuracy misura la percentuale di previsioni corrette rispetto al numero totale di campioni.
È calcolata come il rapporto tra il numero di previsioni corrette (veri positivi e veri negativi) e il numero totale di campioni.
- Precision: La precision misura la percentuale di previsioni positive corrette rispetto al numero totale di previsioni positive fatte dal modello.
È calcolata come il rapporto tra il numero di veri positivi e la somma dei veri positivi e dei falsi positivi.
- Recall: La recall (o sensibilità o true positive rate) misura la percentuale di campioni positivi correttamente identificati dal modello rispetto al numero totale di campioni positivi nella popolazione.
È calcolata come il rapporto tra il numero di veri positivi e la somma dei veri positivi e dei falsi negativi.
- F1-score: L'F1-score è una media armonica tra la precisione e la recall. È utile quando si desidera trovare un equilibrio tra precisione e recall.
È calcolato come $2 * ((\text{precision} * \text{recall}) / (\text{precision} + \text{recall}))$.

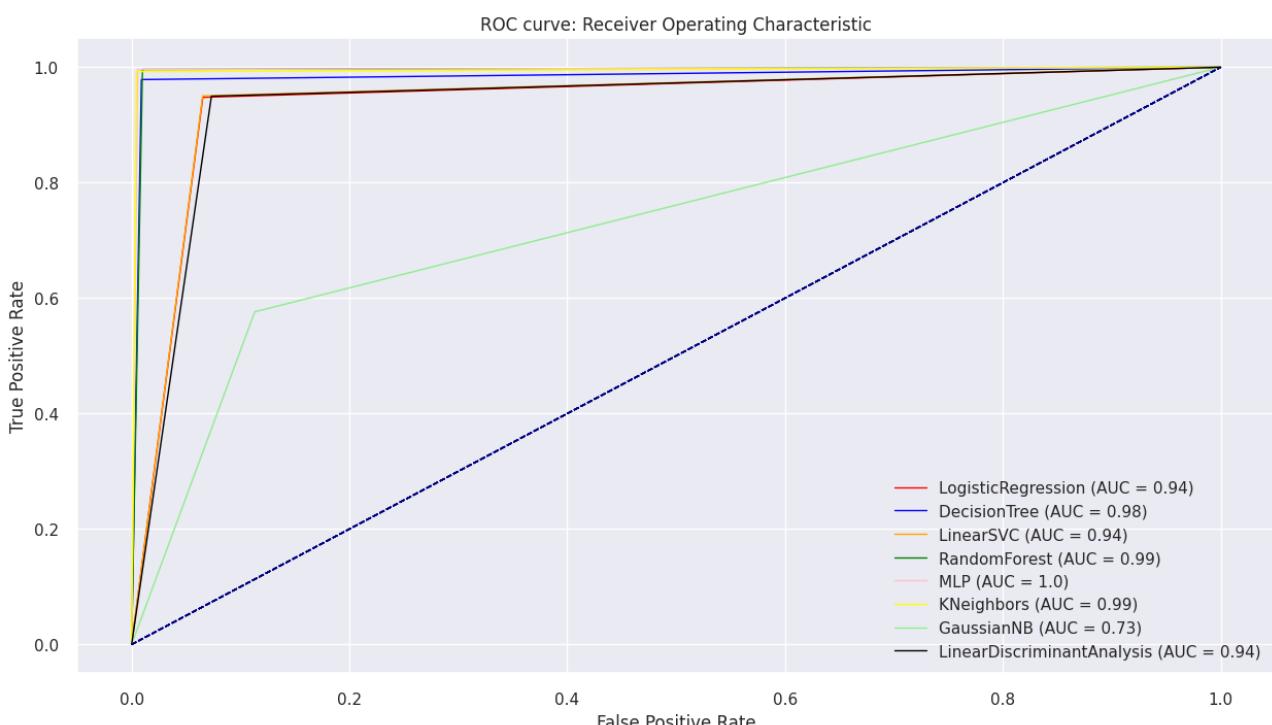
		Precision	Recall	F1-score	Accuracy
<i>LogisticRegression</i>	<i>Edible</i>	0.93	0.96	0.94	0.94
	<i>Poisonous</i>	0.95	0.92	0.94	
<i>DecisionTreeClassifier</i>	<i>Edible</i>	0.99	0.98	0.99	0.99
	<i>Poisonous</i>	0.98	0.99	0.98	
<i>LinearSVC</i>	<i>Edible</i>	0.93	0.96	0.95	0.94
	<i>Poisonous</i>	0.95	0.92	0.94	
<i>RandomForestClassifier</i>	<i>Edible</i>	0.99	1.00	0.99	0.99
	<i>Poisonous</i>	1.00	0.99	0.99	
<i>MLPClassifier</i>	<i>Edible</i>	0.99	1.00	1.00	1.0
	<i>Poisonous</i>	1.00	0.99	0.99	
<i>KNeighborsClassifier</i>	<i>Edible</i>	1.00	0.99	0.99	0.99
	<i>Poisonous</i>	0.99	0.99	0.99	
<i>GaussianNB</i>	<i>Edible</i>	0.89	0.39	0.54	0.65
	<i>Poisonous</i>	0.58	0.94	0.72	
<i>LinearDiscriminantAnalysis</i>	<i>Edible</i>	0.93	0.96	0.94	0.94
	<i>Poisonous</i>	0.95	0.91	0.93	

Un'altra metrica usata per valutare gli algoritmi di classificazione è la mean cross-validation accuracy: è calcolata come la media delle accuracies ottenute in ogni iterazione di cross validazione.



Ultima metrica utilizzata è la Receiver Operating Characteristic (curva ROC). Questa curva esprime il numero dei veri positivi in funzione dei falsi positivi, al variare di un parametro del classificatore. Un buon classificatore dovrebbe avere una curva ROC il più distante possibile da quella associata al classificatore casuale, cioè la retta che passa per l'origine e taglia il grafico a 45°.

La valore associato alla curva ROC è l'Area Under the Receiver Operating Characteristic curve (AUC-ROC) che rappresenta l'area sottesa alla curva ROC e fornisce una misura complessiva delle prestazioni del modello.



4.4 Conclusioni

In conclusione, analizzando le metriche di precisione, recall, f1-score e accuracy, possiamo determinare il classificatore migliore da scegliere.

Ecco una breve analisi dei risultati:

- LogisticRegression: Mostra buone prestazioni con una precisione media, una recall media e un f1-score medio del 94%. L'accuracy è anche del 94%. È un classificatore solido e bilanciato.
- DecisionTreeClassifier: Mostra prestazioni eccellenti con una precisione media, una recall media e un f1-score medio del 99%. L'accuracy è anche del 99%. Questo classificatore sembra essere molto efficace e preciso.
- LinearSVC: Ha risultati simili al LogisticRegression con una precisione media, una recall media e un f1-score medio del 94%. L'accuracy è anche del 94%. Sembra essere un classificatore affidabile.
- RandomForestClassifier: Ottiene risultati eccezionali con una precisione media, una recall media e un f1-score medio del 99%. L'accuracy è anche del 99%. È un classificatore molto potente e preciso, anche se potrebbe essere un po' più complesso rispetto ad altri classificatori.
- MLPClassifier: Mostra prestazioni perfette con una precisione media, una recall media e un f1-score medio del 100%. L'accuracy è anche del 100%. Questo classificatore sembra essere altamente accurato e potente, anche se potrebbe richiedere più risorse computazionali.
- KNeighborsClassifier: Ottiene risultati molto buoni con una precisione media, una recall media e un f1-score medio del 99%. L'accuracy è anche del 99%. È un classificatore affidabile e preciso.
- GaussianNB: Mostra prestazioni inferiori rispetto agli altri classificatori con una precisione media, una recall media e un f1-score medio del 65%. L'accuracy è del 65%. Potrebbe essere meno affidabile rispetto agli altri classificatori in questo caso specifico.
- LinearDiscriminantAnalysis: Ottiene risultati molto simili a LogisticRegression e LinearSVC con una precisione media, una recall media e un f1-score medio del 94%. L'accuracy è anche del 94%. Sembra essere un classificatore bilanciato e affidabile.

Dai dati sopra riportati, il classificatore migliore sembra essere il MLPClassifier, che raggiunge una precisione, recall, f1-score e accuracy del 100%. Tuttavia, va notato che potrebbe richiedere più risorse computazionali rispetto agli altri classificatori. Altrimenti, se si preferisce un classificatore con risultati eccellenti ma meno complesso, il RandomForestClassifier e il DecisionTreeClassifier sono scelte valide con una precisione, recall, f1-score e accuracy del 99%.

In definitiva, la scelta del classificatore migliore dipenderà anche da altri fattori come il contesto specifico, la dimensione del dataset, la scalabilità e le risorse computazionali disponibili