



UNIVERSITA' DEGLI STUDI DI  
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base  
Corso di Laurea Magistrale in Ingegneria Informatica

## *Dashboard - "daily climate change in Dehli, India"*

Anno Accademico 2023/2024

Professore  
**Flora Amato**

Studenti  
**Gaetano Saviano matr. M63001502**  
**Francesco Della Valle matr. M63001500**

# Contents

<b>1</b>	<b>Dashboard interattiva</b>	<b>1</b>
1.1	Traccia . . . . .	1
1.2	Introduzione . . . . .	2
<b>2</b>	<b>Dataset</b>	<b>3</b>
2.1	Descrizione del Dataset . . . . .	3
2.2	Descrizione delle features . . . . .	3
2.3	Obiettivo . . . . .	4
<b>3</b>	<b>Dashboard Streamlit</b>	<b>5</b>
3.1	Librerie usate . . . . .	5
3.2	Primi passi . . . . .	6
3.3	Caricamento dataset . . . . .	7
3.4	Analisi in serie . . . . .	8
3.5	Relazioni tra feature . . . . .	9
3.6	Analisi dei dati con Plotly Express . . . . .	10
3.7	Predizioni con Sarima . . . . .	10

# Chapter 1

## Dashboard interattiva

### 1.1 Traccia

La traccia del secondo progetto dell'elaborato di Information Systems and Business Intelligence prevede di sviluppare una dashboard, col fine di:

- visualizzare informazioni chiave, tendenze e insight estratti dai dati.
- utilizzare grafici, tabelle, e mappe interattive.
- presentare i dati in modo che siano intuitivi e informativi per gli utenti finali.

In questa documentazione saranno riportati e commentati passo passo i codici utilizzati.

## 1.2 Introduzione

La comprensione approfondita dei dati richiede strumenti di visualizzazione intuitivi e potenti. In questo capitolo, esploreremo come creare una dashboard interattiva utilizzando **Streamlit**, una libreria Python che consente la costruzione rapida di applicazioni web per l'analisi dei dati.

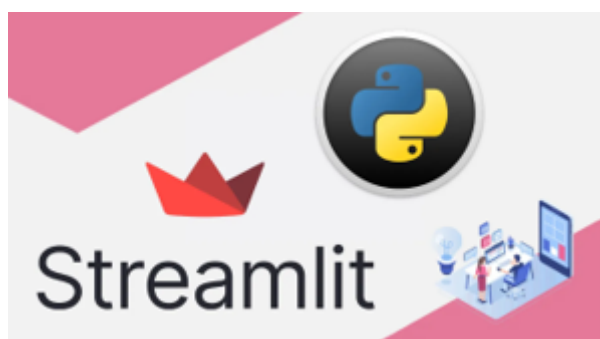


Figure 1.1: Logo Streamlit

L'obiettivo è trasformare un dataset grezzo in visualizzazioni che siano:

- significative.
- esplorabili.
- interattive.

Utilizzeremo le capacità di Streamlit per fornire agli utenti un'esperienza coinvolgente, consentendo loro di esaminare grafici, trend e pattern direttamente attraverso un'interfaccia web intuitiva.

# Chapter 2

## Dataset

Il dataset utilizzato per questa analisi del cambiamento climatico a Delhi è stato acquisito da Kaggle e può essere trovato al seguente indirizzo: `https://www.kaggle.com/datasets/sumanthvrao/daily-climate-time-series-data`

### 2.1 Descrizione del Dataset

Il dataset contiene dati sul cambiamento climatico nella città di Delhi, in India, dal 1° gennaio 2013 al 24 aprile 2017. I dati sono rappresentati da 5 indicatori per 1462 giorni.

### 2.2 Descrizione delle features

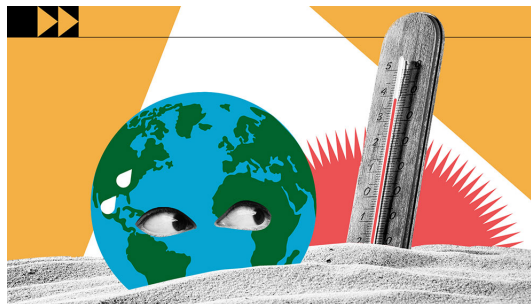
Il dataset presenta 5 feature per descrivere i 1462 campioni:

- **Data (date):** La data del campionamento dei dati.

- **Temperatura Media (meantemp):** La temperatura media calcolata da intervalli multipli di 3 ore nel corso della giornata.
- **Umidità (humidity):** Il valore di umidità per la giornata, espresso in grammi di vapore acqueo per metro cubo di volume d'aria.
- **Velocità del Vento (wind\_speed):** La velocità del vento misurata in chilometri orari.
- **Pressione Atmosferica (pressure):** La lettura della pressione atmosferica, misurata in atmosfere.

## 2.3 Obiettivo

L'obiettivo principale è comprendere le variazioni climatiche e effettuare previsioni utilizzando competenze di machine learning e analisi dei dati. Questo dataset è stato raccolto per supportare la ricerca nell'ambito delle scienze ambientali, consentendo analisi approfondite e lo sviluppo di modelli predittivi per comprendere meglio il cambiamento climatico nella città di Delhi nel periodo specificato.



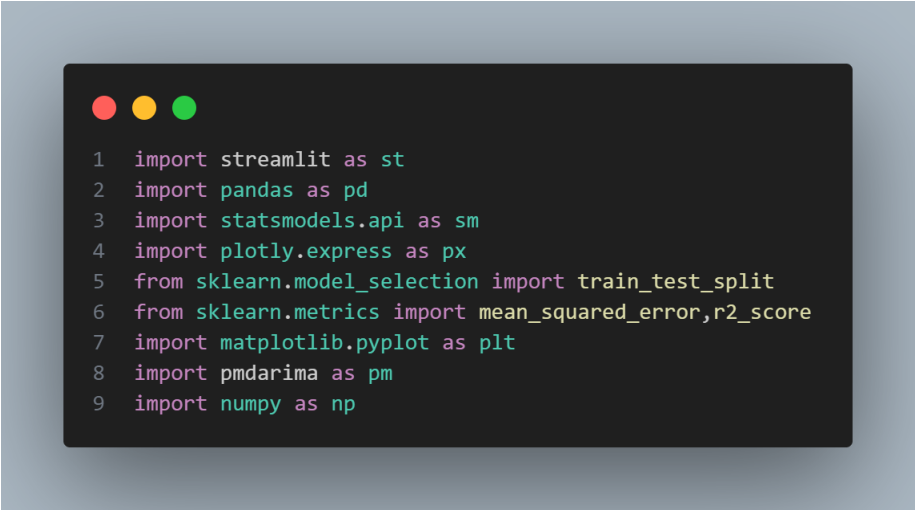
## Chapter 3

# Dashboard Streamlit

### 3.1 Librerie usate

Tra i principali vantaggi dell'utilizzo del linguaggio Python vi è l'enorme quantità di librerie messe a disposizione dello sviluppatore, tra le quali anche quelle per l'analisi dati e la creazione di app web, che rendono la programmazione più efficiente e potente. Tra queste citiamo **Streamlit**, che è la chiave per creare app web interattive con Python, rendendo la visualizzazione dei dati dinamica e coinvolgente, **Pandas** è un potente strumento per la manipolazione e l'analisi dei dati, fornendo strutture dati flessibili e intuitive, **Statsmodels** offre un vasto set di strumenti per l'analisi statistica, consentendo la creazione e l'esplorazione di modelli per comprendere meglio i dati, **Plotly Express** è un modo rapido e intuitivo per creare visualizzazioni di dati interattive e accattivanti, **Matplotlib** è una libreria fondamentale per

la creazione di grafici e visualizzazioni personalizzate in Python. Di seguito riportato il codice:

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains a list of Python import statements, each preceded by a line number from 1 to 9. The imports are for streamlit, pandas, statsmodels, plotly, sklearn, matplotlib, pmdarima, and numpy.

```
1 import streamlit as st
2 import pandas as pd
3 import statsmodels.api as sm
4 import plotly.express as px
5 from sklearn.model_selection import train_test_split
6 from sklearn.metrics import mean_squared_error, r2_score
7 import matplotlib.pyplot as plt
8 import pmdarima as pm
9 import numpy as np
```

## 3.2 Primi passi

Prima di procedere con l'implementazione del codice vero e proprio, è necessario eseguire una serie di step preliminari per consentire il corretto deploy della dashboard. E' necessario, infatti, dapprima la creazione di un ambiente virtuale al nostro code editor, tramite:

```
python -m venv venv
```



Dopo aver attivato l'ambiente virtuale, si procede all'installazione dei framework necessari (come Streamlit), tramite la dicitura:

```
pip install
```

Nel file *requirements.txt* sono riportate tutte le dipendenze necessarie al corretto funzionamento della dashboard implementata.

### 3.3 Caricamento dataset

A screenshot of a code editor window with a dark background and light-colored text. The code is written in Python and is intended for use in a Streamlit dashboard. It includes comments in Italian and English, and uses pandas for data loading and Streamlit for display. The code is as follows:

```
1 # Carica il dataset
2 dataset_path = "DailyDelhiClimateTrain.csv"
3 df = pd.read_csv(dataset_path)
4 st.title("Daily climate change in the city of
5 Delhi ")
6 # Display the dataset
7 st.write("Dataset completo:")
8 st.write(df)
```

L'output su dashboard consente la visualizzazione del titolo in front page, e il display del dataset intero, come riportato in figura.

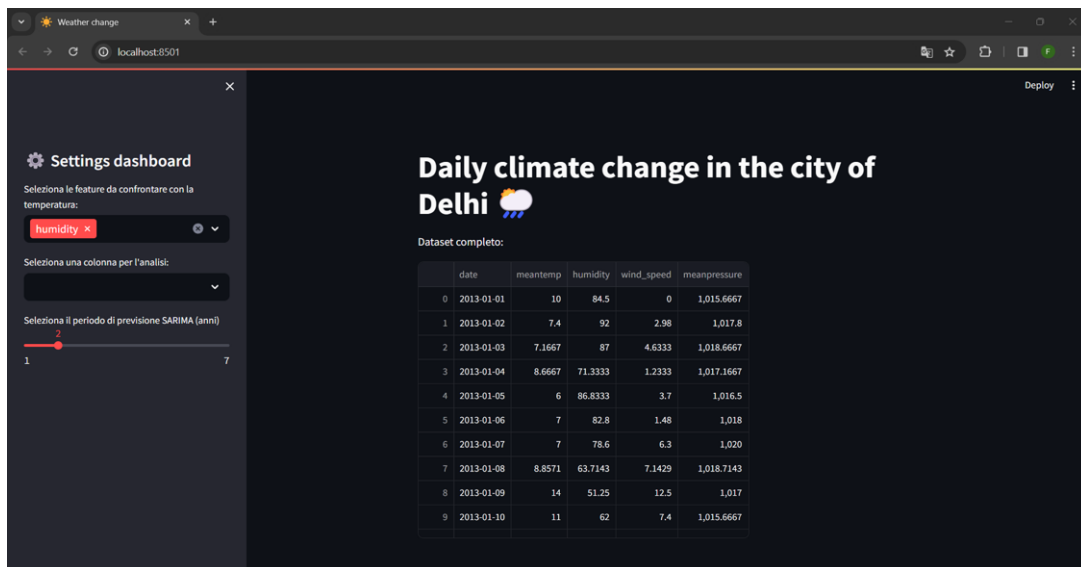


Figure 3.1: Front page Dashboard

### 3.4 Analisi in serie

Si riporta il codice per la visualizzazione dei grafici prodotti per l'analisi in serie del dataset, in particolare delle feature relative a *meantemp*, *humidity*, *wind speed*, *meanpressure*.

```

1 # Analisi in serie
2 st.header("Analisi in serie:")
3 # grafici che si desidera visualizzare in base alle colonne del dataset
4 st.write(" Temperatura Media")
5 st.line_chart(df.set_index('date')['meantemp'])
6 st.write(" Umidita")
7 st.line_chart(df.set_index('date')['humidity'])
8 st.write(" Velocita del Vento")
9 st.line_chart(df.set_index('date')['wind_speed'])
10 st.write(" Pressione Media")
11 st.line_chart(df.set_index('date')['meanpressure'])

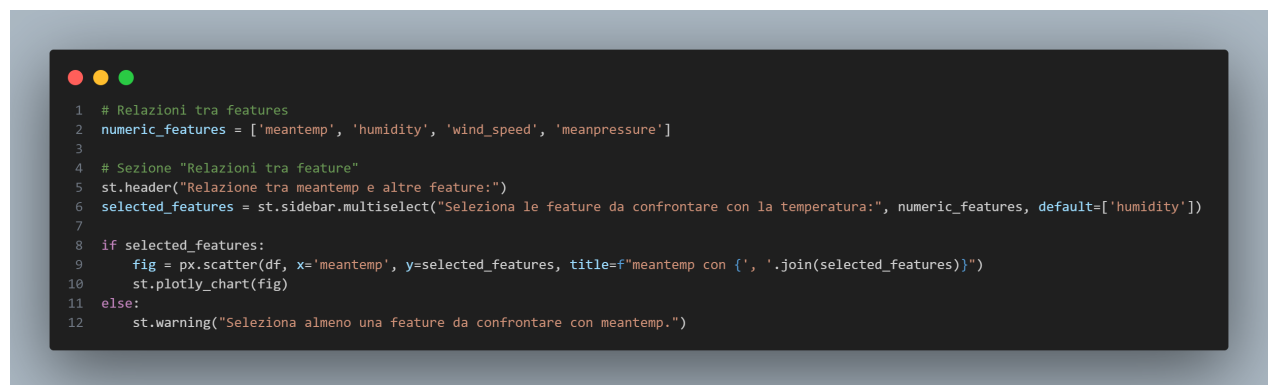
```

### 3.5 Relazioni tra feature

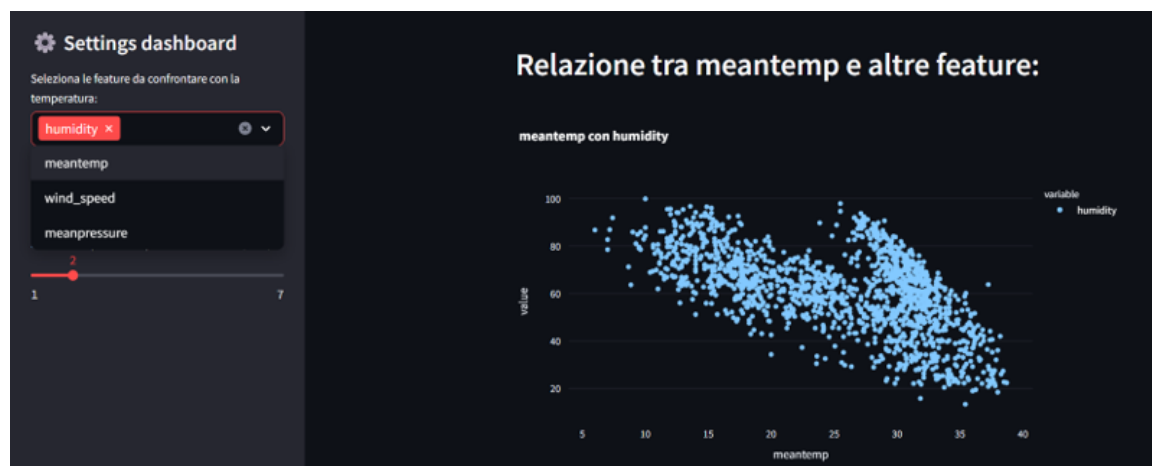
Più interessante è il grafico che riporta le relazioni tra la feature obiettivo (la temperatura media) e le altre nel dataset, per scovarne eventuali dipendenze e correlazioni. Si noti l'utilizzo di

`st.sidebar.multiselect`

che consente la scelta all'utente di quali feature correlare alla temperatura media in modo interattivo e intuitivo, con il grafico che evolve dinamicamente a seconda delle feature scelte.




Si riporta l'output della relazione tra le feature *meantemp* e *humidity*, che sono quelle più interessanti per l'analisi dei dati.



## 3.6 Analisi dei dati con Plotly Express

Si riporta il codice per l'utilizzo della libreria **Plotly** per la realizzazione di un grafico dinamico che, a seguito della feature scelta dall'utente, consente l'analisi della feature stessa.

A screenshot of a code editor window with a dark background and light-colored text. The code is written in Python and uses the Streamlit library. It defines a list of numeric features, creates a sidebar with a multiselect widget, and uses Plotly Express to create a scatter plot based on the selected features. The plot title is dynamically generated based on the selected features. A warning message is shown if no features are selected.

```
1 # Relazioni tra features
2 numeric_features = ['meantemp', 'humidity', 'wind_speed', 'meanpressure']
3
4 # Sezione "Relazioni tra feature"
5 st.header("Relazione tra meantemp e altre feature:")
6 selected_features = st.sidebar.multiselect("Seleziona le feature da confrontare con la temperatura:", numeric_features, default=['humidity'])
7
8 if selected_features:
9     fig = px.scatter(df, x='meantemp', y=selected_features, title=f"meantemp con {' '.join(selected_features)}")
10    st.plotly_chart(fig)
11 else:
12    st.warning("Seleziona almeno una feature da confrontare con meantemp.")
```

## 3.7 Predizioni con Sarima

Anche per questa funzionalità della dashboard è prevista la scelta, da parte dell'utente, della quantità di anni per cui si è interessati a fare predizione (forecasting). L'utente, infatti, deve selezionare tramite lo *slider* presente nella barra laterale un numero, e poi premere il bottone apposito. Il codice di questa sezione finale e l'output su dashboard concludono la documentazione di questo progetto.

## CHAPTER 3. DASHBOARD STREAMLIT

```
1 # Modello SARIMA
2 st.header("Modello SARIMA")
3 time_column_sarima = 'date'
4 temp_column_sarima = 'meantemp'
5
6 df['date'] = pd.to_datetime(df['date'])
7 df_aggregated_sarima = df.groupby(pd.Grouper(key='date', freq='M')).mean().reset_index()
8
9 if len(df_aggregated_sarima) < 2:
10     st.error("Non ci sono abbastanza dati per generare le previsioni.")
11 else:
12     order_sarima = (2, 0, 0)
13     seasonal_order_sarima = (1, 1, 1, 4)
14
15 model_sarima = sm.tsa.SARIMAX(df_aggregated_sarima[temp_column_sarima], order=order_sarima, seasonal_order=seasonal_order_sarima)
16 result_sarima = model_sarima.fit()
17
18 forecast_period_years_sarima = st.sidebar.slider("Seleziona il periodo di previsione SARIMA (anni)", 1, 7, 2)
19
20 st.info("Dopo aver selezionato il numero di anni per il forecast, clicca su 'Esegui previsioni SARIMA'")
21
22 if st.button("Esegui previsioni SARIMA"):
23     last_date_sarima = pd.to_datetime(df_aggregated_sarima[time_column_sarima].max())
24     forecast_period_start_sarima = last_date_sarima + pd.DateOffset(days=-1)
25     forecast_period_end_sarima = forecast_period_start_sarima + pd.DateOffset(years=forecast_period_years_sarima)
26     forecast_index_sarima = pd.date_range(start=forecast_period_start_sarima, end=forecast_period_end_sarima, freq='M')
27
28     forecast_df_sarima = pd.DataFrame({'date': forecast_index_sarima[:-1],
29                                     'Forecast': result_sarima.get_forecast(steps=len(forecast_index_sarima)-1).predicted_mean})
30
31     fig_sarima = px.line(df_aggregated_sarima, x=time_column_sarima, y=temp_column_sarima,
32                         title=f'{temp_column_sarima} con Previsioni SARIMA')
33     fig_sarima.add_scatter(x=forecast_df_sarima['date'], y=forecast_df_sarima['Forecast'],
34                           mode='lines', name='Previsioni Future', line=dict(color='red'))
35     st.plotly_chart(fig_sarima)
```

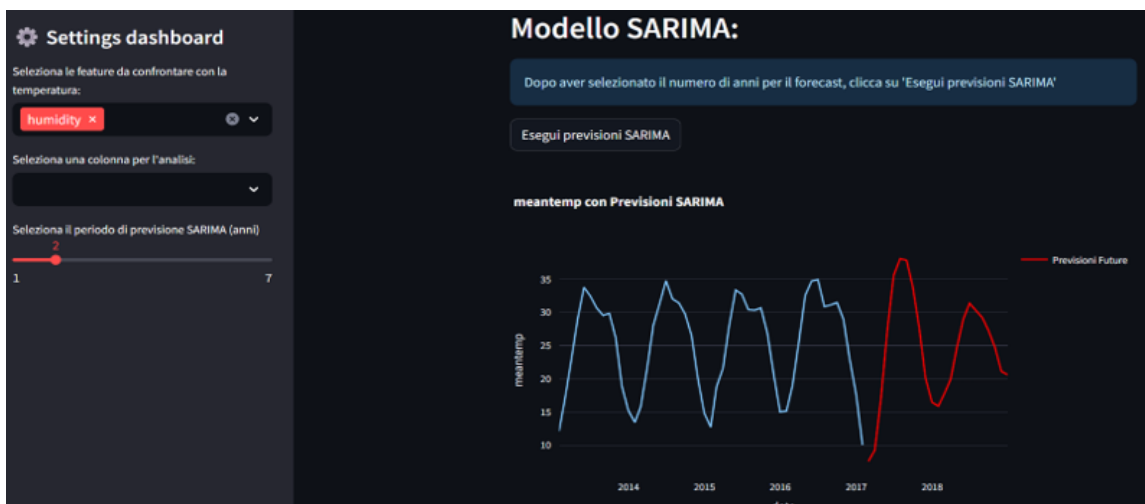


Figure 3.2: Predizioni Dashboard