



**Università
degli Studi
di Palermo**

Relatore: Davide Taibi

Studente: Manuel Ciancimino

L'utilizzo dei Large Language Model a supporto dei processi di analisi dei dati automatizzati



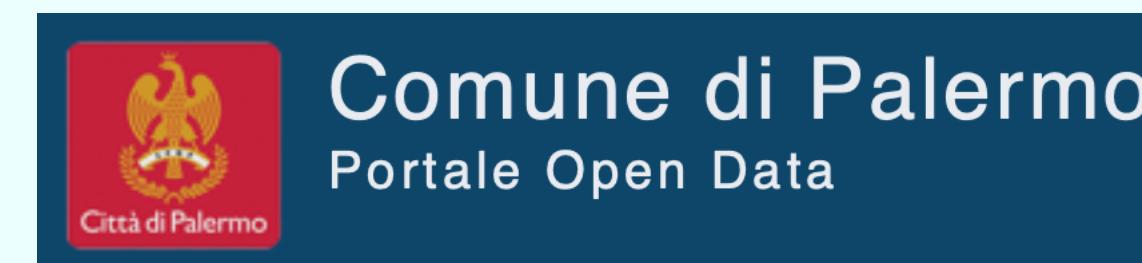
Motivazioni

- Obiettivo: Data la grande potenzialità degli LLM ci siamo posti l'obiettivo di verificare in che modo questi possono intervenire nell'implementazione di una pipeline dell'elaborazione dei dati.
- Per il nostro studio abbiamo utilizzato come caso d'uso gli Open Data. Negli ultimi anni la quantità di dati aperti è in aumento ma non sempre la qualità dei dati pubblicati è alta, rendendo, di conseguenza, necessario una pre-elaborazione per garantire un loro utilizzo efficace.



Università
degli Studi
di Palermo

Datasets



Luoghi del turismo di Palermo



Impianti di Rifornimento



Profili di prompting

- **AUTONOMO**: non si fornisce nessuna indicazioni sul modo in cui svolgere un determinato compito; si chiede semplicemente di risolverlo
- **GUIDATO**: viene data indicazione sommaria di come risolvere un problema; vengono utilizzate nella richiesta competenze di dominio
- **ESPERTO / GUIDATO CON ESEMPI**: viene fornito il modo in cui risolvere un problema, utilizzando una terminologia dettagliata. Viene fornito molto contesto in modo tale che il problema possa risultare il più chiaro possibile.

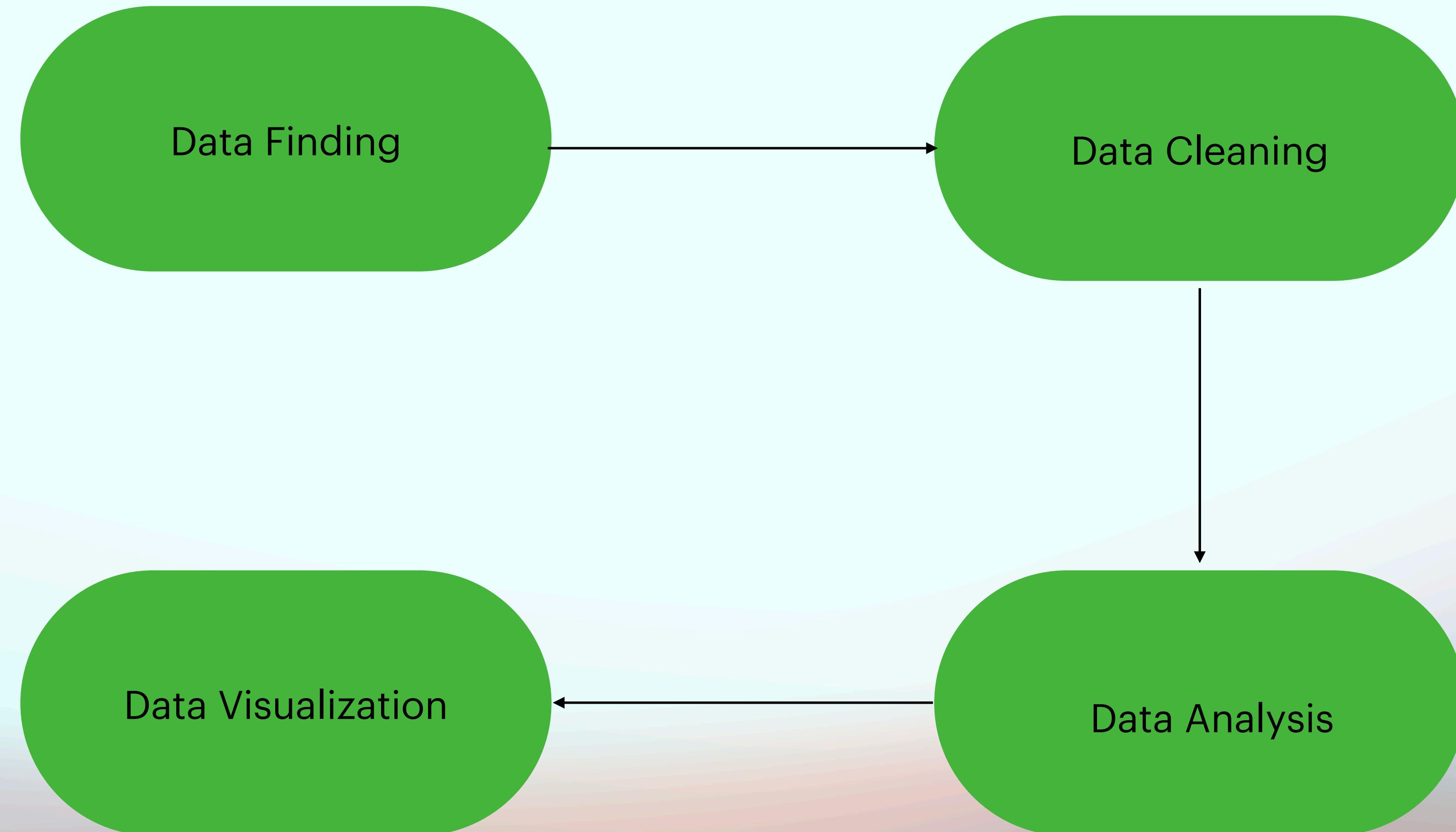
CASI D'USO:



1. **Data journalist**: viene simulato il comportamento che può adottare un individuo non esperto di programmazione ma che comunque conosce quali elaborazioni sono necessarie per realizzare dei grafici da inserire nei propri articoli;
2. **Programmatore**: un individuo esperto di programmazione ma che non ha grande esperienza di utilizzo degli LLM, quindi saprà sicuramente porre domande adeguate e fornire contesto;
3. **Programmatore + esperto di prompt**: un individuo esperto di programmazione e che conosce come funzionano gli LLM e come utilizzarli al meglio. Nelle sue domande inserisce molto contesto, istruisce il modello andando anche a specificare cosa si aspetta nelle risposte. Inoltre va ad inserire quali sono i criteri da rispettare nelle risposte.



Pipeline di elaborazione dati





Data finding



Luoghi del turismo di Palermo

Accesso ai dati: per leggere i dati occorre usare la funzione `read_xml` di Pandas, che richiede come parametro la root dell'albero dei tag, quindi bisogna saperla individuare; in caso contrario non sarà possibile ottenere il dataset.



Impianti di Rifornimento

Accesso ai dati: si tratta di due dataset che possono essere collegati facilmente utilizzando pandas e passando alla funzione `read_csv` il link al sito.

Nota: Occorre pulire l'intestazione del file e prendere come header la seconda riga.





Data Cleaning: differenze di prompt tra profili

Al seguente link è presente il file xml relativo ai luoghi del turismo siciliano <LINK>. Le colonne che contiene il file xml sono:
<COLUMNS>

Obiettivo: effettuare data cleaning del dataset per una futura elaborazione utilizzando python. Ciò che devi fare è analizzare le entità del file xml, riscontrare se ci sono problematiche e porre delle soluzioni.

Contenuto file: <FILE_CONTENT>

Utilizzando Python e Pandas, utilizza la funzione `read_xml` prendendo come parametro il seguente url: <LINK> che restituisce un file xml e memorizza il dataframe nella variabile `df_turismo`. Le prime righe del dataframe sono le seguenti:

<EXAMPLE>

Fatto ciò noto che sono presenti nella colonna latitudine e longitudine le "," piuttosto dei ".", e questo potrebbe comportare dei problemi; forniscimi una soluzione in python. Analizza il contenuto delle righe che rappresentano una porzione del dataframe e se ci sono ulteriori colonne che possono essere problematiche per una futura elaborazione inserisci le soluzioni all'interno di funzioni separate.

Data la seguente tabella di dati in formato xml dal seguente sito: <LINK>, in cui root =//DATA_RECORD esegui le seguenti operazioni di data cleaning utilizzando la libreria Pandas in Python, proponendo ciò che secondo te andrebbe fatto seguendo le seguenti linee guida:

Direttive per l'implementazione del codice:

- Suddividi il codice in funzioni specifiche: ...
- Commenta ogni funzione: ...
- Ottimizza le funzioni per efficienza: ...
- Favorisci la leggibilità e la manutenibilità del codice: ...
- Implementa il multithreading...

Le funzioni che devi implementare sono:

- `parse_xml` ...
- gestione dei valori mancanti: ...
- verifica attraverso l'uso di espressioni regolari se le seguenti colonne contengono delle stringhe adeguate.
- funzione che nelle colonne [LATITUDE] e [LONGITUDE] sostituisce la ',' con ':'

Esempio di struttura del codice:



Data Cleaning

```
def validate_urls(df, columns):
    """
    Validate URLs in specified columns using regular expression.

    :param df: DataFrame containing the data.
    :param columns: List of column names to be validated.
    :return: DataFrame with invalid URLs replaced with np.nan.
    """
    pattern = re.compile(r'^((http://www\.|https://www\.|http://|https://)?[a-z0-9]+([\-\.\.]{1}[a-z0-9]+)*\.[a-z]{2,5}(:[0-9]{1,5})?(/.*?)?$')
    for column in columns:
        df[column] = df[column].where(df[column].apply(lambda x: bool(pattern.match(x))), np.nan)
    return df

def validate_phones(df, columns):
    # <doc>
    for column in columns:
        df[column] = df[column].apply(lambda x: f"+39{x}" if not x.startswith('+39') and len(x) == 10 else x)
    return df
```



```
#2. Funzioni per sostituire "," con "." nelle colonne LATITUDE e LONGITUDE
def replace_commas(df):
    #come caso precedente

#3. Funzioni per verificare la formattazione delle colonne con espressioni regolari
def validate_email(email):
    pattern = r'^[\w\.-]+@[^\s]+\.\w+$'
    return bool(re.match(pattern, email))

def validate_url(url):
    pattern = r'^https?://[^s]+$'
    return bool(re.match(pattern, url))

def validate_phone(phone):
    pattern = r'^\d{10,15}$'
    return bool(re.match(pattern, phone))

def validate_gestore_telefono(phone):
    <simile caso validate_phone>

def validate_columns(df):
    df['EMAIL_VALID'] = df['EMAIL'].apply(validate_email)
    df['GESTORE_WEB_VALID'] = df['GESTORE_WEB'].apply(validate_url)
    df['TELEFONO_VALID'] = df['TELEFONO'].apply(validate_phone)
    df['GESTORE_TELEFONO_VALID'] = df['GESTORE_TELEFONO'].apply(validate_gestore_telefono)
    return df

#4. Gestione dei valori mancanti con np.nan
def handle_missing_values(df):
    df.replace('0', np.nan, inplace=True)
    df.replace('', np.nan, inplace=True)
    return df
```





Data Cleaning

Cosa è riuscito a fare

- Replace nelle stringhe
- Utilizzare espressioni regolari per verificare la corretta formattazione dei siti, email, numeri di cellulare
- Individuazione delle problematiche interne al file
- Gestione dei valori mancanti inserendo un placeholder
- Inserire commenti alle funzioni indicando quali sono i parametri e valori di ritorno, inserendo inoltre una descrizione della funzione
- Implementazione multi-threading del codice

Cosa NON è riuscito a fare

- Produrre del codice subito eseguibile



Data Analysis: Differenze di prompt tra profili

Sei un programmatore esperto in Python. Ho memorizzato due file csv: anagrafica_impianti.csv e prezzo_alle_8. Il loro contenuto è:

Anagrafica_impianti: <contenuto dataset>

Prezzo_alle_8:<contenuto dataset>

Forniscimi il codice per eseguire le seguenti operazioni:

1. Unire i due csv
2. Il luogo in cui il costo della benzina 'fai da te' è più alto e dove più basso
3. Ottenerne la bandiera con il prezzo più alto o più basso.
4. Ottenerne la bandiera con il prezzo più alto per ciascuno dei seguenti carburanti: 'Benzina' definita 'fai da te', 'Gpl', 'Gasolio' definita 'fai da te', 'Metano'.
5. Pivot table, in cui i carburanti da considerare sono: 'Benzina', 'Gpl', 'Gasolio', 'Metano'

Sei un programmatore esperto in Python. I due file csv sono:

Anagrafica_impianti: <contenuto dataset>

Prezzo_alle_8:<contenuto dataset>

Direttive per l'implementazione del codice:

- Suddividi il codice in funzioni specifiche
- Commenta ogni funzione
- Ottimizza le funzioni per efficienza
- Favorisci la leggibilità e la manutenibilità del codice

Esempio di struttura del codice:

Forniscimi il codice per eseguire le seguenti operazioni:
<come per programmatore>

Sei un programmatore esperto in Python. Ho memorizzato due file csv: anagrafica_impianti.csv e prezzo_alle_8. Il loro contenuto è:

Anagrafica_impianti: <contenuto dataset>

Prezzo_alle_8:<contenuto dataset>

Forniscimi il codice per eseguire le seguenti operazioni:

Unire i due csv: utilizzare la funzione merge, andando a settare il parametro 'validate' a one-to-many poiché nel dataframe 'prezzo_alle_8' sono presenti più righe con la stessa chiave 'id_impianto'.

Ottenerne il luogo in cui il costo della benzina 'fai da te' è più alto e dove più basso. Considerando che ho già il dataframe (ottenuto al punto precedente), dato dall'unione dei due precedenti, utilizza un filtro per il tipo di carburante 'benzina' e poi utilizza la funzioni max e min. Il risultato finale della funzione è la riga relativa al prezzo max e min, formattata nel seguente modo: 'Gestore', 'Bandiera', 'Provincia', 'Comune', 'Indirizzo', 'descCarburante', 'prezzo'

Ottenerne la bandiera con il prezzo più alto o più basso...

Ottenerne la bandiera con il prezzo più alto per ciascuno dei seguenti carburanti...

Ottenerne una pivot table ...



Data Analysis

```
#Ottenere la bandiera con il prezzo più alto per ciascuno dei seguenti carburanti
def highest_price_bandiera(carburante, is_self):
    filtered = merged_df[(merged_df['descCarburante'] == carburante) & merged_df['isSelf'] == is_self]
    return filtered.loc[filtered['prezzo'].idxmax()][['Bandiera', 'prezzo']]

highest_benzina_self = highest_price_bandiera('Benzina', 1)
#altri caburanti ...

#Pivot table per i carburanti Benzina, Gpl, Gasolio, Metano
pivot_table = merged_df.pivot_table(values='prezzo', index='Bandiera',
    columns='descCarburante', aggfunc='max')[['Benzina', 'GPL', 'Gasolio', 'Metano']]
```



```
#Trova il massimo per ciascun carburante
results = {}
for fuel in 'Benzina', 'GPL', 'Gasolio', 'Metano':
    specific_fuel = merged_df[(merged_df['descCarburante'] == fuel) & merged_df['isSelf'] == 1]
    max_row = specific_fuel.loc[specific_fuel['prezzo'].idxmax()]
    results[fuel] = max_row[['Bandiera', 'Gestore', 'Provincia', 'Comune', 'Indirizzo', 'descCarburante', 'prezzo']]

for key, value in results.items():
    print(f"Massimo per {key}:", value)

pivot_table = pd.pivot_table(merged_df, values='prezzo', index='Provincia', columns='descCarburante', aggfunc='mean')
```



```
def max_price_per_fuel_type(df, fuel_types):
    """
    Ottiene il prezzo massimo per ciascun tipo di carburante specificato.

    :param df: DataFrame unificato.
    :param fuel_types: Lista di tuple contenenti tipo di carburante e modalità self-service.
    :return: DataFrame con i massimi prezzi per tipo di carburante.
    """

    result = pd.DataFrame()
    for desc, is_self in fuel_types:
        df_filtered = df[(df['descCarburante'] == desc) & df['isSelf'] == is_self]
        max_price_row = df_filtered[df_filtered['prezzo'] == df_filtered['prezzo'].max()]
        result = pd.concat([result, max_price_row], axis=0)
    return result[['Bandiera', 'Gestore', 'Provincia', 'Comune', 'Indirizzo']]

def create_pivot_table(df, index, columns, values, aggfunc='mean'):
    """
    Crea una pivot table con i parametri specificati.

    :param df: DataFrame da cui creare la pivot table.
    :param index: Colonna da usare come indice.
    :param columns: Colonna da usare come colonne.
    :param values: Colonna da cui prendere i valori.
    :param aggfunc: Funzione di aggregazione.
    :return: Pivot table risultante.
    """

    return pd.pivot_table(df, index=index, columns=columns, values=values, aggfunc=aggfunc)
```





Data Analysis

Cosa è riuscito a fare

- Merge dei due dataframe individuando la colonna in comune
- Min e Max per una colonna rispettando diverse condizioni
- GroupBy e Filter
- Pivot Table
- Individuazione di proprietà di una colonna

Cosa NON è riuscito a fare

- Notare che i due dataframe hanno una relazione uno a molti e dunque è necessario includere un parametro aggiuntivo nella funzione merge
- Utilizzare Nominatim per verificare se coordinate corrispondevano ad un intorno della via e numero civico indicati



Data Visualization: Differenze di prompt tra profili

Sei un esperto nell'utilizzo di python per la data visualization. Ho il seguente file csv che contiene tutti gli impianti di rifornimento d'Italia (ottenuto dalla fase di analisi):

<contenuto file>

Forniscimi il codice per rappresentare graficamente ciò che ti sembra significativo.

<Fase iniziale>

<Grafici da creare>

Voglio i grafici ma nella prossima risposta dimmi se ci sono dei parametri importanti da settare ma che non ho menzionato e che permettono una migliore visualizzazione dei grafici.

Costruisci questi suggerimenti in modo tale da poterli modificare e riscriverteli nella domanda successiva.

Il set di dati è memorizzato nel file impianti.csv con il seguente contenuto:

<contenuto file>

<Fase Iniziale>

<Librerie da utilizzare>

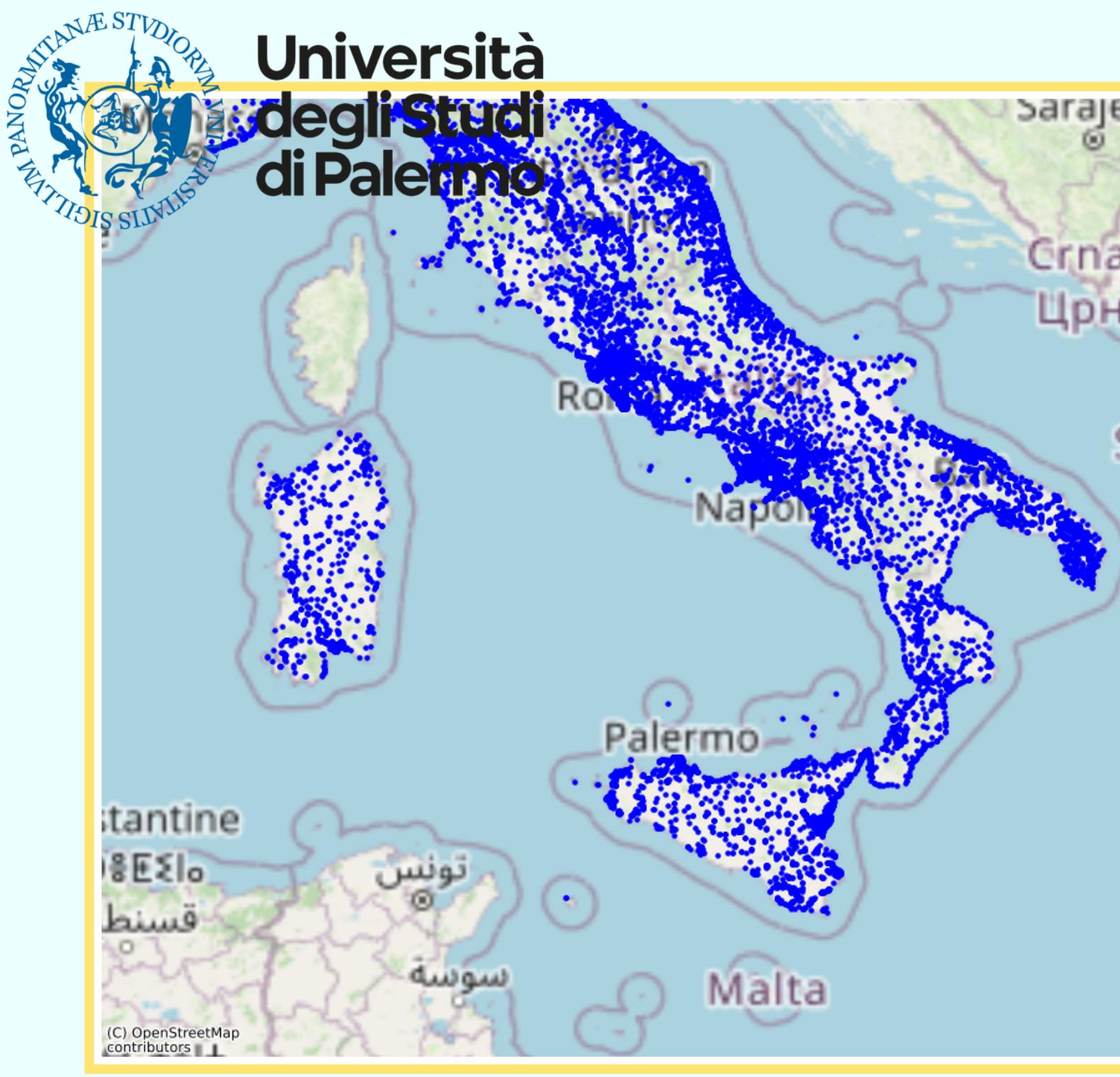
Utilizzando il set di dati a seguire, restituiscimi il codice per poter creare:

- Grouped bar chart in cui le label rappresentano la media dei costi per benzina e gasolio nelle province siciliane
- Bar color in cui metti in evidenza il costo medio della benzina e gasolio nella regione Sicilia
- Horizontal bar plots in cui ogni barra contiene al suo interno il prezzo medio per gasolio e benzina, ogni barra rappresenta una regione italiana
- Grafico a torta in cui ogni porzione contiene il numero di impianti di rifornimento per ogni provincia siciliana.
- Mappa degli impianti siciliani

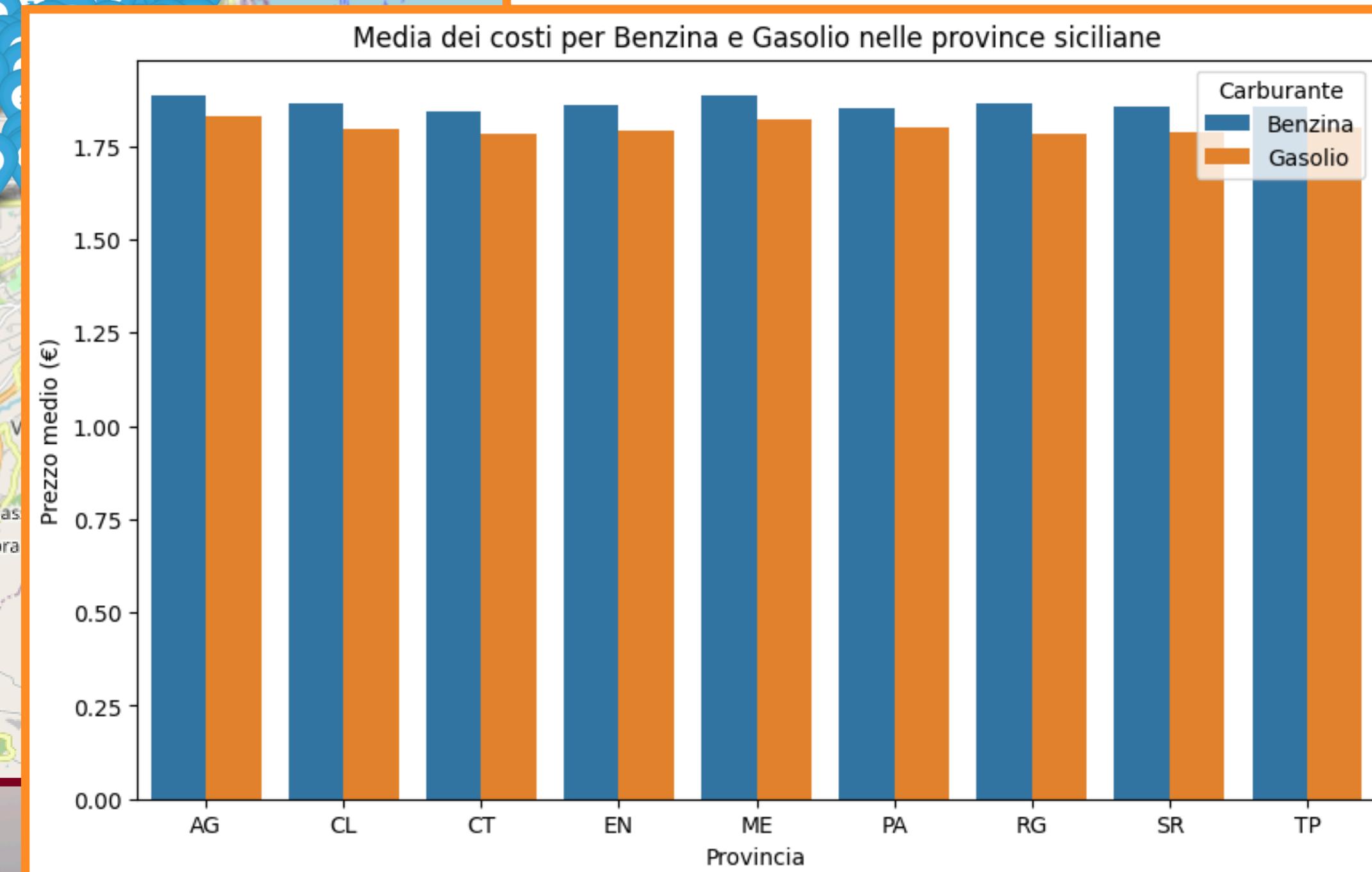
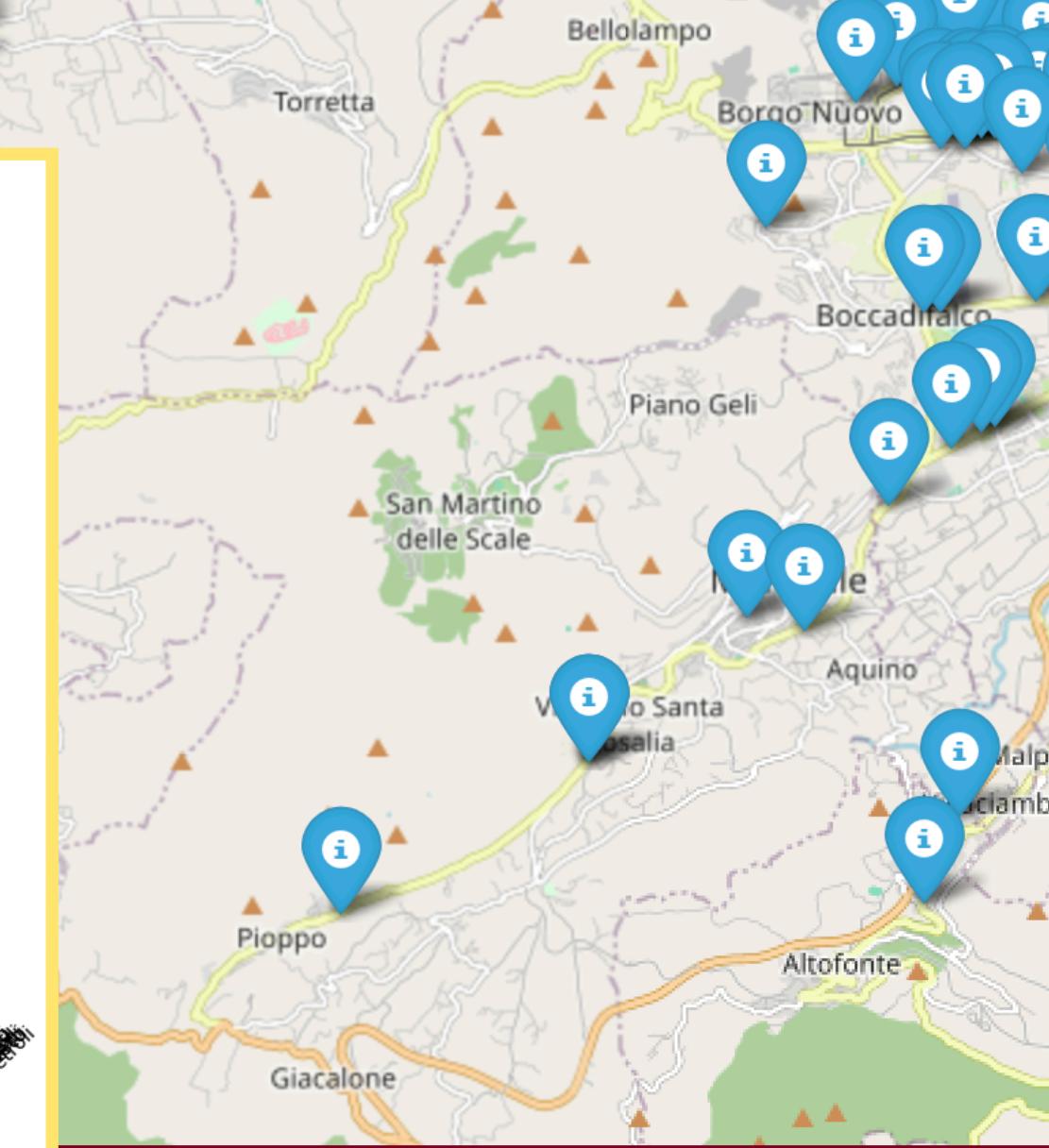
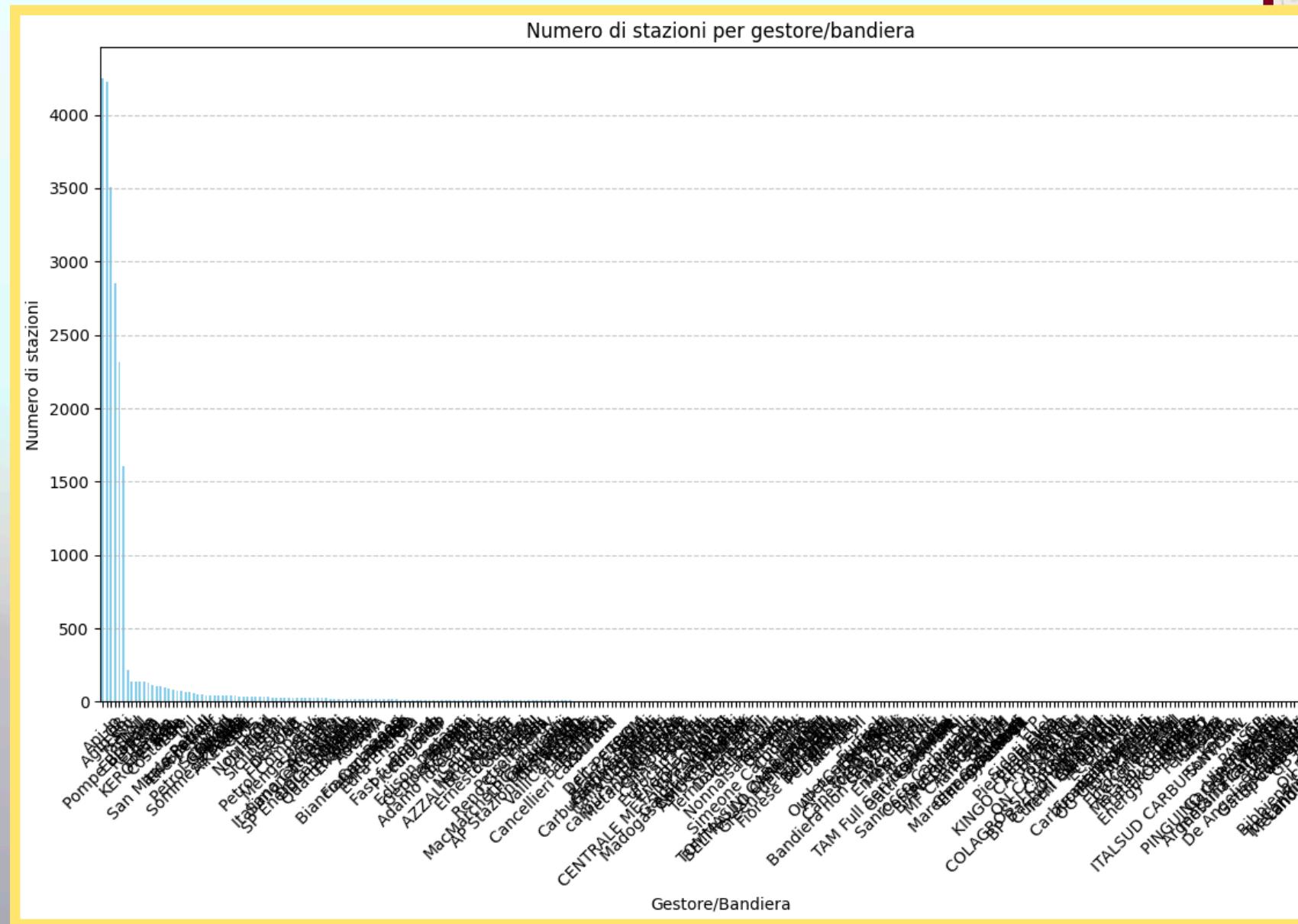
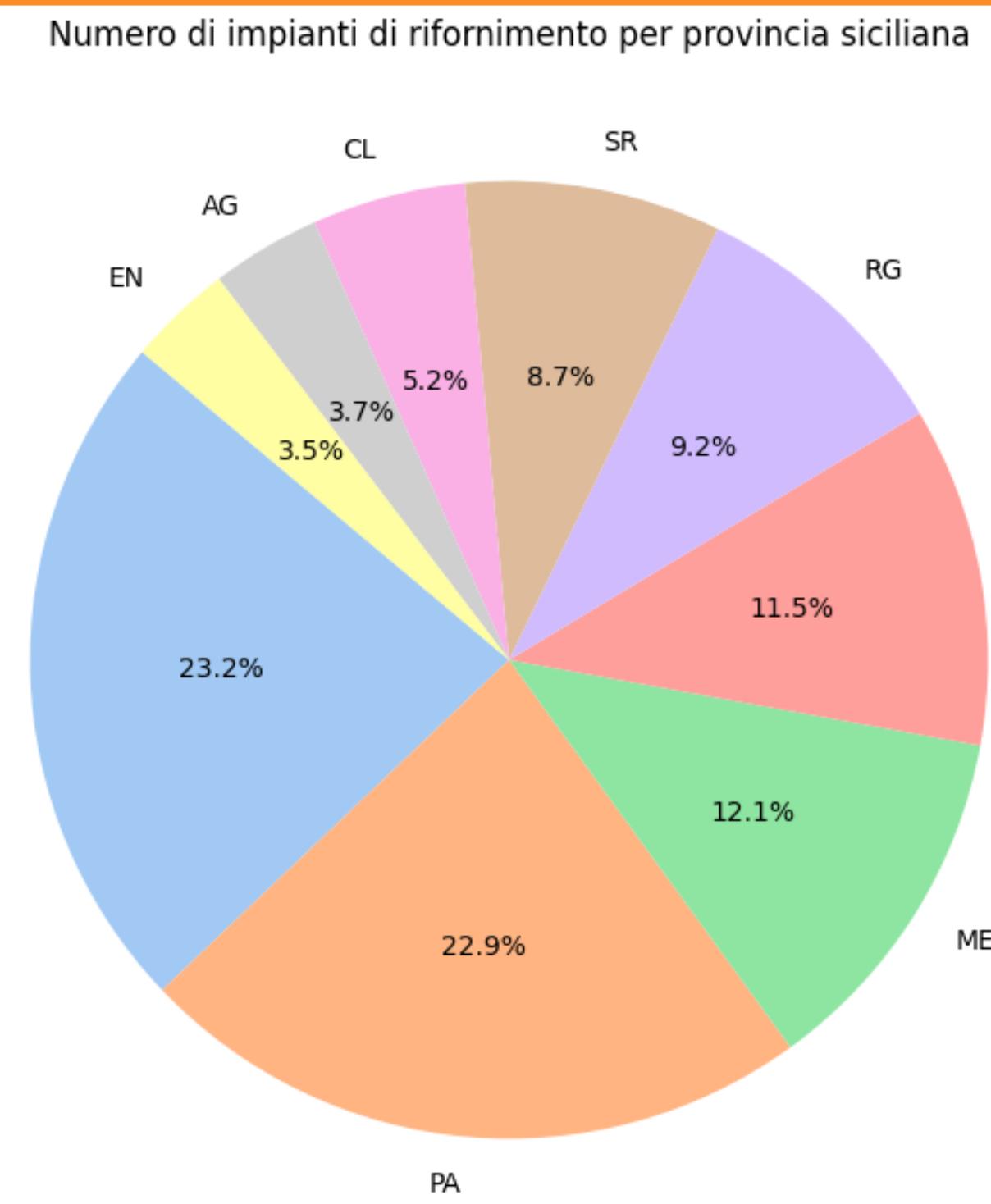
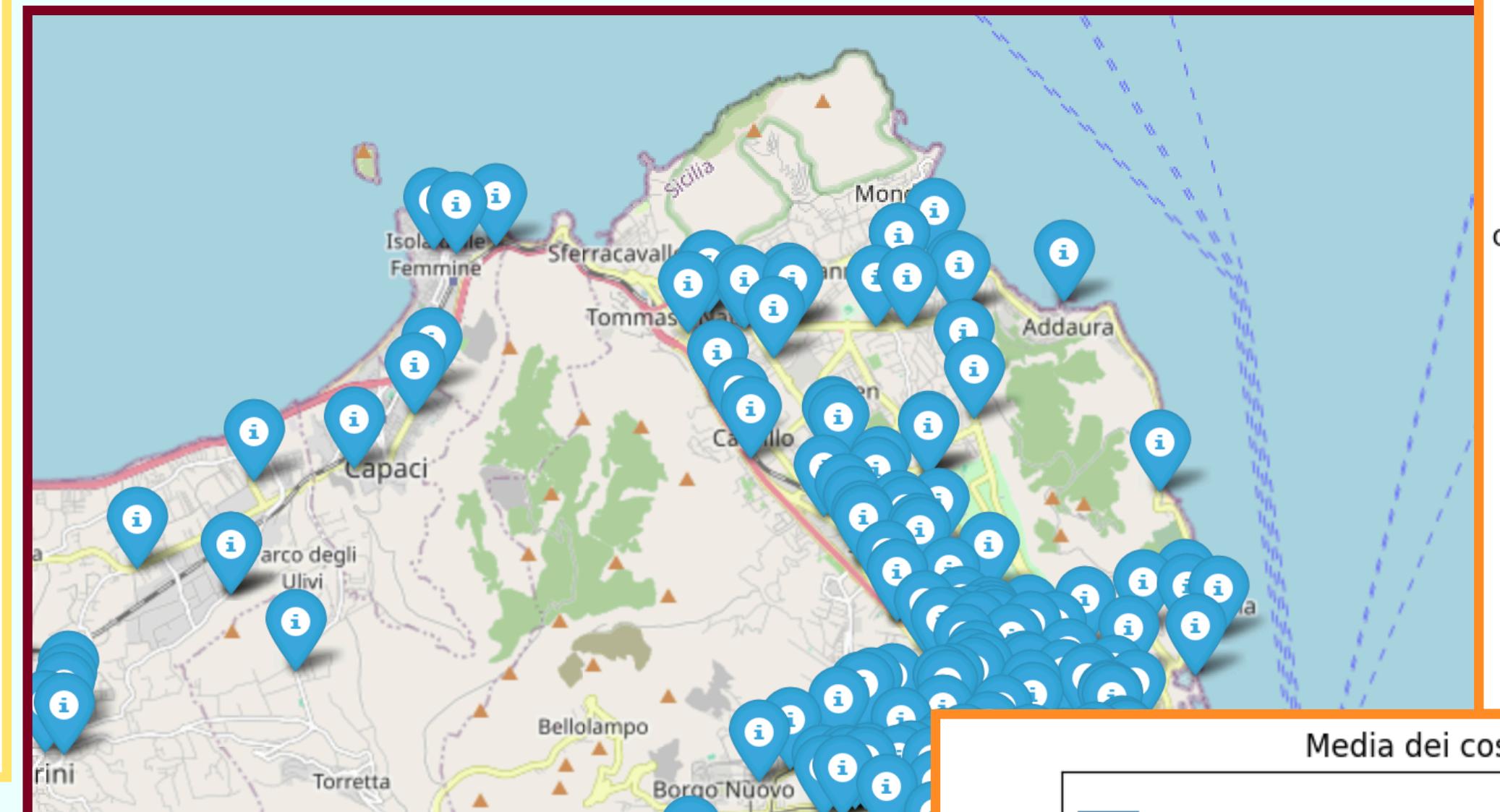
Per ogni grafico aggiungi un breve racconto di cosa rappresentano i grafici, scrivilo in modo accattivante.

Il set di dati è memorizzato nel file impianti.csv con il seguente contenuto:

<contenuto file>



Data Visualization





Data Visualization

Cosa è riuscito a fare

- È forse la parte che riesce a fare meglio, utilizza abilmente le funzioni di ogni libreria
- Determinare Latitudine e Longitudine della regione
- Il motivo per cui funziona bene potrebbe essere dato dal fatto che sono processi molto simili tra loro e la varietà di ragionamento logico può essere inferiore rispetto alla data analisi

Cosa NON è riuscito a fare

- Non riesce a raccogliere le province per regioni (no colonna Regione), richiesta: Horizontal bar plots in cui ogni barra contiene al suo interno il prezzo medio per gasolio e benzina, ogni barra rappresenta una regione italiana



Considerazioni Finali

È uno strumento veramente utile?

Gli LLM hanno un impatto sulla nostra vita professionale?

Gli LLM possono davvero aiutarci o dobbiamo averne timore?

In realtà: Se la nostra conoscenza di un dato argomento non è ampia allora è ovvio che ciò che ci verrà suggerito ci apparirà sbalorditivo, ma per ottenere dei risultati ottimali è necessaria avere conoscenza del dominio. Quanta più dimestichezza si ha con un certo dominio, e quanta più conoscenza verticale si ha su di esso, tanto più è possibile interagire con LLM per ottenere risposte efficaci.

E' possibile utilizzare gli LLM in modo proficuo?

Si, è possibile utilizzare gli LLM per svolgere una grande varietà di compiti ma non è consigliabile affidarsi totalmente ad essi. L'intervento umano è ancora necessario.

Gli LLM potranno sostituire i programmati?

Semplicemente no. Ciò che riescono a fare in modo eccellente è adattare le nostre richieste a ciò che è già stato fatto.



**Università
degli Studi
di Palermo**

Grazie per l'attenzione