TPSIT - Esercitazione sui Dataset "Marziani"

Questo notebook contiene una guida dettagliata per l'analisi di dati strutturati utilizzando le librerie Python più comuni nell'ambito della data science: Pandas e Matplotlib. L'esercitazione utilizza un dataset fittizio che rappresenta caratteristiche di creature "marziane".

Indice

1. Introduzione alle librerie

Prima di iniziare l'analisi, importiamo le librerie necessarie:

```
# Pandas: libreria per la manipolazione e l'analisi dei dati
import pandas as pd

# Matplotlib: libreria per la visualizzazione dei dati
import matplotlib.pyplot as plt

# NumPy: libreria per il calcolo numerico (usata implicitamente da pandas)
import numpy as np
```

Pandas offre strutture dati e funzioni per manipolare tabelle numeriche e serie temporali. Le due strutture dati principali sono:

- **Series**: Array monodimensionale etichettato
- **DataFrame**: Tabella bidimensionale con colonne potenzialmente di tipo diverso

Matplotlib è una libreria completa per la creazione di visualizzazioni statiche, animate e interattive in Python.

2. Caricamento dei dati

Il file "marziani.csv" contiene dati relativi a due specie di creature marziane con varie caratteristiche fisiche:

```
# Leggiamo il file CSV e creiamo un DataFrame
marziani = pd.read_csv("marziani.csv")

# Visualizziamo le prime righe del DataFrame
marziani.head()
```

La funzione `pd.read_csv()` è fondamentale per importare dati da file CSV. Alcuni parametri utili sono:

- `delimiter`: carattere usato per separare i campi (default: ',')
- `header`: riga da usare come intestazione delle colonne
- `names`: nomi alternativi per le colonne
- `index_col`: colonna da usare come indice
- `skiprows`: numero di righe da saltare all'inizio del file

3. Esplorazione iniziale dei dati

Esaminiamo la struttura del DataFrame per comprendere meglio i dati:

```
# Visualizzare i tipi di dati di ciascuna colonna
marziani.dtypes

# Ottenere informazioni generali sul DataFrame
marziani.info()

# Statistiche descrittive delle colonne numeriche
marziani.describe()
```

- `dtypes`: mostra il tipo di dato di ciascuna colonna
- `info()`: fornisce un riepilogo conciso del DataFrame
- `describe()`: genera statistiche descrittive (count, mean, std, min, quartili, max)

Il nostro dataset contiene le seguenti colonne:

- **specie**: tipo di creatura marziana (categorico)
- **colore**: colore della creatura (categorico)
- **n_arti**: numero di arti (numerico)
- **peso**: peso della creatura (numerico)
- **altezza**: altezza della creatura (numerico)

- **larghezza**: larghezza della creatura (numerico)

4. Pulizia dei dati

Spesso i dataset contengono valori mancanti o errati. Puliamo il nostro DataFrame:

```
# Rimuoviamo le righe con valori mancanti
marziani_ripulito = marziani.dropna()

# Verifichiamo il risultato
marziani_ripulito.info()
```

Metodi comuni per la pulizia dei dati:

- `dropna()`: elimina righe o colonne con valori mancanti
- `fillna()`: sostituisce i valori mancanti con un valore specificato
- `replace()`: sostituisce valori specifici
- `duplicated()` e `drop_duplicates()`: identifica e rimuove i duplicati

5. Salvataggio dei dati

Dopo la pulizia, possiamo salvare il DataFrame in vari formati:

```
# Salvataggio in formato Excel
marziani_ripulito.to_excel("marziani.xlsx", sheet_name="Marziani")

# Altri formati di salvataggio disponibili
# marziani_ripulito.to_csv("marziani_pulito.csv")
# marziani_ripulito.to_json("marziani.json")
# marziani_ripulito.to_pickle("marziani.pkl")
```

Il metodo `to_excel()` richiede l'installazione del pacchetto `openpyxl` o `xlsxwriter`.

6. Filtri e query sui dati

Pandas offre potenti funzionalità per filtrare i dati:

```
# Filtro semplice: marziani con più di 27 arti
marziani_con_molti_arti = marziani_ripulito[marziani_ripulito["n_arti"] >
27]
print(marziani_con_molti_arti)

# Conteggio: quanti marziani hanno più di 28 arti?
num_marziani_28_arti = len(marziani_ripulito[marziani_ripulito["n_arti"] >
28])
print(f"Numero di marziani con più di 28 arti: {num_marziani_28_arti}")
```

```
# Filtri complessi con operatori logici (AND, OR)
# Marziani della specie Robby, di colore blu, con peso o altezza superiore
alla media

# Calcoliamo prima le medie
media_peso = marziani_ripulito["peso"].mean()
media_altezza = marziani_ripulito["altezza"].mean()

# Applichiamo il filtro combinato
marziani_filtrati = marziani_ripulito[
    (marziani_ripulito["specie"] == "Robby") &
    (marziani_ripulito["colore"] == "blu") &
    ((marziani_ripulito["peso"] > media_peso) |
(marziani_ripulito["altezza"] > media_altezza))
]

print(f"Numero di Robby blu con peso o altezza superiore alla media:
{len(marziani_filtrati)}")
```

Operatori di confronto e logici in Pandas:

- `==`, `!=`, `>`, `<`, `>=`, `<=`: confronto
- `&`: AND logico (le parentesi intorno alle condizioni sono necessarie)
- `|`: OR logico
- `~`: NOT logico

7. Statistiche descrittive

Calcoliamo alcune statistiche di base:

```
# Massimo e minimo del numero di arti
max_arti = marziani_ripulito["n_arti"].max()
min_arti = marziani_ripulito["n_arti"].min()

print(f"Numero massimo di arti: {max_arti}")
print(f"Numero minimo di arti: {min_arti}")

# Altre statistiche utili
# Media
media_peso = marziani_ripulito["peso"].mean()
# Mediana
mediana_peso = marziani_ripulito["peso"].median()
# Deviazione standard
std_peso = marziani_ripulito["peso"].std()
# Quartili
quartili_peso = marziani_ripulito["peso"].quantile([0.25, 0.5, 0.75])
```

```
print(f"Statistiche per il peso:")
print(f"Media: {media_peso:.2f}")
print(f"Mediana: {mediana_peso:.2f}")
print(f"Deviazione standard: {std_peso:.2f}")
print(f"Quartili: {quartili_peso.to_dict()}")
```

8. Aggregazione e raggruppamento

Il metodo `groupby()` permette di raggruppare i dati in base a una o più colonne:

```
# Media delle caratteristiche raggruppate per specie e colore
medie_per_gruppo = marziani_ripulito[["n_arti", "altezza", "peso",
"larghezza", "specie", "colore"]].groupby(["specie", "colore"]).mean()
print(medie_per_gruppo)

# Conteggio per specie e colore
conteggio_per_gruppo = marziani_ripulito[["specie",
"colore"]].value_counts()
print(conteggio_per_gruppo)

# Altre operazioni di aggregazione disponibili
# .sum(), .min(), .max(), .count(), .std(), .var(), .sem(), .describe(),
.first(), .last()
```

Possiamo anche usare `agg()` per applicare diverse funzioni a diverse colonne:

```
# Aggregazione personalizzata
aggregazione = marziani_ripulito.groupby(["specie", "colore"]).agg({
    "n_arti": ["mean", "max", "min"],
    "peso": ["mean", "median", "std"],
    "altezza": ["mean", "max"]
})
print(aggregazione)
```

9. Visualizzazione dei dati

Matplotlib ci permette di creare visualizzazioni efficaci:

```
# Grafico a dispersione per peso e larghezza, colorato per specie
fig, ax = plt.subplots(figsize=(10, 8))

# Plot per i Robby (in rosso)
marziani_ripulito[marziani_ripulito["specie"] == "Robby"].plot.scatter(
    x="peso",
    y="larghezza",
    c="r",
    label="Robby",
```

```
    ax=ax
)

# Plot per i Simmy (in verde)
marziani_ripulito[marziani_ripulito["specie"] == "Simmy"].plot.scatter(
    x="peso",
    y="larghezza",
    c="g",
    label="Simmy",
    ax=ax
)

# Personalizzazione del grafico
ax.set_xlabel("Peso", fontsize=16)
ax.set_ylabel("Larghezza", fontsize=16)
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)
ax.set_title("Relazione tra Peso e Larghezza dei Marziani", fontsize=20)
ax.grid(True)
ax.legend(fontsize=16)

# Mostra il grafico
plt.tight_layout()
plt.show()
```

Altri tipi di grafici comuni:

- Istogrammi: `plt.hist()` o `df.hist()`
- Diagrammi a barre: `plt.bar()` o `df.plot.bar()`
- Diagrammi a torta: `plt.pie()` o `df.plot.pie()`
- Boxplot: `plt.boxplot()` o `df.plot.box()`
- Heatmap: `plt.imshow()` o con `seaborn.heatmap()`

10. Esercizi aggiuntivi

Ecco alcuni esercizi aggiuntivi per approfondire l'analisi:

1. Crea un istogramma della distribuzione del numero di arti per specie
2. Calcola la correlazione tra le variabili numeriche
3. Crea un boxplot per confrontare la distribuzione del peso tra le due specie
4. Trova i marziani outlier (con valori molto distanti dalla media)
5. Crea una nuova colonna "BMI" definita come peso/(altezza^2)
6. Applica una trasformazione di normalizzazione alle variabili numeriche

```
# Esempio di soluzione per l'esercizio 1
plt.figure(figsize=(12, 6))
```

```
# Istogramma per Robby
plt.subplot(1, 2, 1)
marziani_ripulito[marziani_ripulito["specie"] == "Robby"]
["n_arti"].hist(bins=10, color='red', alpha=0.7)
plt.title("Distribuzione n° arti - Robby")
plt.xlabel("Numero di arti")
plt.ylabel("Frequenza")

# Istogramma per Simmy
plt.subplot(1, 2, 2)
marziani_ripulito[marziani_ripulito["specie"] == "Simmy"]
["n_arti"].hist(bins=10, color='green', alpha=0.7)
plt.title("Distribuzione n° arti - Simmy")
plt.xlabel("Numero di arti")
plt.ylabel("Frequenza")

plt.tight_layout()
plt.show()

# Esempio di soluzione per l'esercizio 2
matrice_correlazione = marziani_ripulito[["n_arti", "peso", "altezza",
"larghezza"]].corr()
print("Matrice di correlazione:")
print(matrice_correlazione)

# Visualizzazione della matrice di correlazione (richiede seaborn)
import seaborn as sns
plt.figure(figsize=(10, 8))
sns.heatmap(matrice_correlazione, annot=True, cmap='coolwarm', vmin=-1,
vmax=1)
plt.title("Matrice di correlazione delle caratteristiche")
plt.tight_layout()
plt.show()
```

Conclusioni

Questo notebook ha illustrato i concetti fondamentali dell'analisi dati con Python, utilizzando pandas per la manipolazione dei dati e matplotlib per la visualizzazione. Le competenze acquisite sono applicabili a qualsiasi dataset strutturato e rappresentano la base per analisi più avanzate in ambito data science e machine learning.

I concetti chiave includono:

- Caricamento e pulizia dei dati
- Esplorazione e analisi esplorativa
- Filtri e selezione condizionale
- Statistiche descrittive

- Aggregazione e raggruppamento
- Visualizzazione efficace dei dati