

Lezione 4: Exploratory Data Analysis (EDA)

Davide Evangelista e Dario Lanzoni
`dario.lanzoni6@unibo.it`

Università di Bologna

20 Marzo 2025

*"In statistics, exploratory data analysis (EDA) is an approach of **analyzing data sets to summarize their main characteristics**, often using statistical graphics and other data visualization methods. [...] focuses more narrowly on **checking assumptions required for model fitting** and hypothesis testing, and **handling missing values** and making transformations of variables as needed".*

Un progetto EDA si articola tipicamente in alcuni step fissati:

- **Scelta del dataset:** tramite motori di ricerca come Kaggle o Google Datasets.
- **Esplorazione del dataset:** osservare alcuni dati in esso presenti, interpretare le informazioni a disposizione, pianificare lo studio che si vuole svolgere.
- **Preparazione dei dati (data cleaning):** fondere più datasets (se presenti) per incrementare le informazioni disponibili, aggiustare i tipi di dato (Date, Numeri, Stringhe), standardizzare i valori numerici, gestire i NaN.
- **Indagine statistica:** Utilizzare metodi statistici per estrarre informazioni rilevanti dai dati a disposizione.
- **Visualizzazione** (strettamente collegata con la precedente): Visualizzare attraverso vari tipi di grafici i risultati del punto precedente.

Scelta del dataset

- Esistono motori di ricerca come Kaggle (www.kaggle.com) e Google Datasets (<https://datasetsearch.research.google.com>) in cui è possibile trovare, tramite ricerca con parola chiave) una gran quantità di datasets pubblici.
- Kaggle è di gran lunga il più utilizzato, possiede centinaia di migliaia di datasets, alcuni dei quali ben documentati.
- Nel seguito andremo ad utilizzare principalmente due datasets di esempio:
 - Orange Quality Analysis Dataset: <https://www.kaggle.com/datasets/shruthiiee/orange-quality?resource=download>.
 - House Sales in King County, USA: <https://www.kaggle.com/datasets/harlfoxem/housesalesprediction>.

Esplorazione del dataset

- Da qui in avanti, consideriamo di avere a disposizione un dataset (che indichiamo con X).
- Un dataset è una tabella di valori, in cui le colonne rappresentano le *features*, mentre le righe rappresentano le differenti *osservazioni*. Nel seguito indichiamo con N il numero di righe di X , mentre con d indichiamo il numero di colonne. Dal punto di vista matematico, quindi, un dataset è una matrice di dimensione $N \times d$.
- Abbiamo già osservato che i dataset sono gestiti in Python tramite le funzioni della libreria pandas, i cui oggetti (DataFrame) possono essere caricati con la funzione:

```
> data = pd.read_csv("PATH_TO_CSV.csv")
```
- La libreria seaborn, simile a matplotlib, è molto comoda per visualizzare dati da DataFrame di pandas.

Caricamento dei dati

- Prendiamo in considerazione i dati del file `order_details.csv`, fornito su Virtuale. Carichiamolo in memoria:

```
> data = pd.read_csv("./Orange Quality Data.csv").
```

- Visualizzandone alcuni elementi con la funzione:

```
> print(data.head())
```

osserviamo che alcune colonne possiedono valori numerici, mentre altre sono stringhe.

- Fare particolarmente attenzione quando si lavora con dati che non sono numerici!
- Ricordarsi di visualizzare il numero di righe e di features del dataset.

```
> N, d = data.shape  
> print(f"Shape of data: {N, d}.")  
[1] Shape of data: (241, 11).
```


- E' possibile ottenere maggiori informazioni sulle features del dataset di riferimento tramite il comando `data.info()`.
- Similmente, con il comando `data.describe()` è possibile accedere rapidamente ad informazioni statistiche sul Dataset.
- Possiamo vedere che il dataframe ha 10 colonne (features):
Dimensione (cm), Peso (g), Brix (Dolcezza), pH (Acidità), Morbidezza (1-5), Tempo di raccolta (giorni), Maturazione (1-5), Colore, Varietà, Imperfezioni (Sì/No), Qualità (1-5).

RangeIndex: 241 entries, 0 to 240

Data columns (total 11 columns):

#	Column	Non-Null Count	Dtype
---	-----	-----	-----
0	Size (cm)	241 non-null	float64
1	Weight (g)	241 non-null	int64
2	Brix (Sweetness)	241 non-null	float64
3	pH (Acidity)	241 non-null	float64
4	Softness (1-5)	241 non-null	float64
5	HarvestTime (days)	241 non-null	int64
6	Ripeness (1-5)	241 non-null	float64
7	Color	241 non-null	object
8	Variety	241 non-null	object
9	Blemishes (Y/N)	241 non-null	object
10	Quality (1-5)	241 non-null	float64

dtypes: float64(6), int64(2), object(3)

memory usage: 20.8+ KB

Selezionare le colonne numeriche

- E' spesso una buona idea costruirsi un sub-dataframe del dataset principale contenente solo le variabili numeriche di data. Su questo sub-dataframe sarà possibile eseguire le indagini statistiche tipiche dell'EDA.

```
# Extract numeric values
```

```
> numeric_data = data.select_dtypes(include='number')
```

```
> print(numeric_data.head())
```

	Size (cm)	Weight (g)	...	Ripeness (1-5)	Quality (1-5)
0	7.5	180	...	4.0	4.0
1	8.2	220	...	4.5	4.5
2	6.8	150	...	5.0	5.0
3	9.0	250	...	3.5	3.5
4	8.5	210	...	5.0	4.5

```
[5 rows x 8 columns]
```

Preparazione dei dati

- La preparazione dei dati, o Data Cleaning, è la fase più lunga e delicata in un progetto EDA, e dal suo corretto svolgimento dipende non solo l'analisi statistica successiva, ma anche un eventuale addestramento di modelli di predizione.
- In particolare, è di fondamentale importanza imparare a gestire i dati mancanti (indicati come *Not A Number (NaN)*), come vedremo in seguito.

Merging

- Notiamo che, da Kaggle, possiamo avere accesso a datasets contenenti informazioni differenti. Tuttavia, se i due dataset sono collegati da una colonna comune (come Order ID), questa ci permette di connettere i due datasets.
- Su pandas, i due datasets possono essere uniti mediante la funzione `pd.merge()`, nella quale è necessario specificare il nome della colonna da utilizzare per effettuare la connessione tra i dataset.
- L'operazione con `merge` serve per unire i dataset attraverso la colonna scelta e ordinarli.
[ref]

```
> # Merge the first and the second dataset  
> data2 = pd.read_csv("PATH2_TO_CSV.csv")  
> data = pd.merge(data, data2, on="Order ID")
```

- Dopo il merging, possiamo eliminare la colonna Order ID, ora inutile, tramite il comando `data = data.drop("Order ID")`.

- Si può controllare la tipologia di dato di una colonna di un DataFrame di pandas con il comando `data.dtype`:

```
> print(data["Amount"].dtype)
[1] int64
> print(data["Category"].dtype)
[1] object
```

Variabili Numeriche vs Categorie

- Le variabili numeriche possono essere utilizzate per svolgere operazioni matematiche (per esempio, su variabili numeriche è ben posta la definizione di correlazione).
- Le variabili categoriche contengono valori non numerici, possono essere usate per classificare valori, ma non possono essere utilizzate per funzioni matematiche.
- **Nota:** il dtype object NON indica variabili categoriche per pandas.

```
#Impostiamo variabili categoriche quelle che non sono numeriche
num_type=['float64','int64']
```

```
for col in data.columns:
    print(f"{col} type: {data[col].dtype}.")
    if data[col].dtype not in num_type:
        data[col]=data[col].astype("category")
        print(f"{col} type: {data[col].dtype}.")
print("-"*45)
```


dtype per tutte le colonne

```
> # Check the results ( anche data.info())
```

```
Size (cm) type: float64.
```

```
-----
```

```
. .  
. .  
. .
```

```
Color type: object.
```

```
Color type: category.
```

```
-----
```

```
Variety type: object.
```

```
Variety type: category.
```

```
-----
```

```
Blemishes (Y/N) type: object.
```

```
Blemishes (Y/N) type: category.
```

- Spesso i dataset contengono uno (o più) elementi mancanti, causati per esempio da misurazioni mancate o rimosse poiché irrealistiche, o per via di errori di vario tipo collegati alla collezione dei dati.
- Abbiamo già visto che i dati mancanti vengono mostrati da pandas come dei valori NaN (Not A Number). Per evitare problemi algoritmici, è necessario rimuovere o sostituire i dati mancanti.
- Per controllare la presenza, colonna per colonna, di valori NaN, si può usare il seguente comando:

```
> # Check the presence of NaN values
> print(data.isnull().sum())
```

Il nostro dataset non ne contiene nessuno.

- Quando i valori NaN appaiono in un numero relativamente basso di righe, la cosa più semplice da fare è rimuovere tutte le righe che li contengono. Questo può essere fatto semplicemente con la funzione `data.dropna()`.

Sostituire i valori NaN

- Un metodo molto più efficiente di quello appena descritto, ma molto più complesso da utilizzare, è quello di sostituire i dati mancanti.
- L'opzione più semplice è quella di utilizzare la funzione `data.fillna(VALORE)`, che sostituisce tutti i valori NaN con il valore inserito. La stessa funzione si può utilizzare per sostituire i NaN anche, ad esempio, con il valore del dato precedente.
- Esistono tecniche più avanzate per sostituire i valori NaN, utilizzando informazioni presenti nel dataset per prevedere un valore realistico da inserire al posto del NaN.

Indagine Statistica

Matrice di correlazione (1/4)

- Un'operazione comune praticamente a tutti i progetti di EDA, è la computazione della **Matrice di Correlazione di Pearson**.
- Si basa sulla definizione di correlazione tra variabili aleatorie, definita da:

$$\rho_{X,Y} := \text{Corr}(X, Y) = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}(X)\text{Var}(Y)}},$$

dove:

$$\text{Cov}(X, Y) := \mathbb{E} \left[(X - \mathbb{E}[X]) (Y - \mathbb{E}[Y])^T \right].$$

- Si definisce la matrice di correlazione di Pearson $C \in \mathbb{R}^{d \times d}$ dove:

$$C_{i,j} = \rho_{X_i, X_j}.$$

Matrice di correlazione (2/4)

Si osserva che:



$$-1 \leq C_{i,j} \leq 1, \quad \forall i, j = 1, \dots, d.$$

- Un valore positivo di $C_{i,j}$ indica una correlazione **positiva** tra X_i e X_j . Un valore negativo di $C_{i,j}$ indica una correlazione **negativa** tra X_i e X_j .
- $C_{i,j} = 1$ significa che X_i e X_j sono correlate **deterministicamente**.
- **Nota:** $C_{i,i} = 1$ per ogni $i = 1, \dots, d$.

Matrice di correlazione (3/4)

- La matrice di correlazione su pandas si calcola con il comando `data.corr()`.

```
> numerical_col=data.select_dtypes(include='number')  
> C=numerical_col.corr()  
> print(C.shape)  
[1] (8,8)
```

- E' possibile visualizzare la matrice di correlazione in modo colorato, così da visualizzare le relazioni statistiche tra le features più agevolmente.
- Questa operazione può essere fatta sia con `matplotlib` che con `seaborn`.

```
> sns.heatmap(C, annot=True, cbar=True, cmap='coolwarm', fmt='.2f')
```

Matrice di correlazione (4/4)

- E' possibile visualizzare la matrice di correlazione con il comando `matshow` di `matplotlib`.

```
> plt.matshow(data.corr(), vmin=-1, vmax=1)
> plt.xticks(np.arange(0, data.shape[1]), data.columns, rotation=45)
> plt.yticks(np.arange(0, data.shape[1]), data.columns)
> plt.title("A visualization of the correlation Matrix")
> plt.colorbar()
> plt.show()
```


- Oltre alle informazioni ottenibili dalla matrice di correlazione, e alle statistiche sul dataset visibili tramite la funzione `data.describe()`, informazioni interessanti possono essere ottenute *condizionando* su una delle features del dataset (ovvero, filtrando quegli elementi che rispettano una data condizione).
- Ad esempio, è possibile calcolare la media del qualità delle arance del dataset, e confrontarla con la media della qualità delle arance condizionata al fatto che $\text{pH} > \text{pH}_{\text{medio}}$.
- Analizzando le statistiche delle distribuzioni condizionate rispetto a quelle non condizionate, si possono fare interessanti osservazioni sui dati!

Scatter Plot

- La matrice di Correlazione studia le relazioni tra tutte le features del dataset (*analisi multivariata*). Per studiare le relazioni tra coppie di esse (*analisi bivariata*) è possibile usare lo *scatter plot*.

```
# Creiamo uno scatter plot tra pH (Acidity) e Quality (1-5)
> import matplotlib.pyplot as plt

> plt.scatter(data["pH (Acidity)"], data["Quality (1-5)"], alpha=0.5,
color='red')
> plt.title("Scatter plot tra pH e Qualità")
> plt.xlabel('pH')
> plt.ylabel('Quality')
> plt.show()
```