Lezione 4: Exploratory Data Analysis (EDA)

Davide Evangelista
davide.evangelista5@unibo.it

Università di Bologna

18 Marzo 2024

EDA: Definizione

"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".*

Gestione di un progetto EDA

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

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 principalemente due datasets di esempio:
  - Ice Cream sales: `https://www.kaggle.com/datasets/raphaelmanayon/temperature-and-ice-cream-sales`.
  - Online sales data: `https://www.kaggle.com/datasets/samruddhi4040/online-sales-data/data`.

Esplorazione del dataset

Datasets

- 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 i differenti dati. 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")
```

Caricamento dei dati

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

  ```
  > data = pd.read_csv("./data/order_details.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: (1500, 7).
  ```

Descrizione dei dati

- 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 sulle sole variabili numeriche.

```
[1] RangeIndex: 1500 entries, 0 to 1499
Data columns (total 7 columns):
#   Column         Non-Null Count  Dtype
---  ------         --------------  -----
0   Order ID       1500 non-null   object
1   Amount         1500 non-null   int64
2   Profit         1500 non-null   int64
3   Quantity       1500 non-null   int64
4   Category       1500 non-null   object
5   Sub-Category   1500 non-null   object
6   PaymentMode    1500 non-null   object
```

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())
[1]    Amount  Profit  Quantity
0    1096     658         7
1    5729      64        14
2    2927     146         8
3    2847     712         8
4    2617    1151         4
```

Preparazione dei dati

Data Cleaning

- 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, abbiamo a disposizione due datasets contenenti informazioni differenti: `Orders.csv` e `Details.csv`.
- I due dataset sono collegati tramite la colonna `Order ID`, che 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 connesione tra i dataset.

```
> # Merge the first and the second dataset
> data2 = pd.read_csv("./data/order_details2.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")`.

Tipi di dati su pandas

- 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 Categoriche

- Le variabili numeriche possono essere utilizzate per svolgere operazioni matematiche (per esempio, su variabili numeriche è ben posta la definizione di correlazione).
- Le varabili 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`.

```
> # Transform categorical columns from "object" into "category"
> data["Sub-Category"] = data["Sub-Category"].astype("category")
> data["Category"] = data["Category"].astype("category")
> data["PaymentMode"] = data["PaymentMode"].astype("category")
> data["Order Date"] = data["Order Date"].astype("category")
> data["CustomerName"] = data["CustomerName"].astype("category")
> data["State"] = data["State"].astype("category")
> data["City"] = data["City"].astype("category")
```

dtype per tutte le colonne

```
> # Check if categorical columns are considered "category"
> for col in data.columns:
>  print(f"{col} type: {data[col].dtype}.")
[1] Amount type: int64.
Profit type: int64.
Quantity type: int64.
Category type: category.
Sub-Category type: category.
PaymentMode type: category.
Order Date type: category.
CustomerName type: category.
State type: category.
City type: category.
```

- Alternativamente, lo stesso risultato può essere ottenuto tramite la funzione `data.info()`

Gestione dei valori NaN

- 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()`.

Sostiuire 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 sostiuisce 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 sostiuire 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 varabili aleatorie, definita da:

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

dove:

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

- 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:

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

Matrice di correlazione (3/4)

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

```
> data = pd.read_csv('./data/ice_cream.csv')
> print(data.shape)
[1] (365, 2)
> C = data.corr()
> print(C.shape)
[1] (2, 2)
> print(C)
[1]                      Temperature  Ice Cream Profits
Temperature             1.000000           0.988446
Ice Cream Profits       0.988446           1.000000
```

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()
```

Distribuzioni condizionate

- 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 profitto sugli ordini presenti nel dataset, e confrontarla con la media dei profitti condizionata al fatto che la categoria sia una tra `Electronics`, `Furniture` o `Clothing`.
- Analizzando le statistiche delle distribuzioni condizionate rispetto a quelle non condizionate, su possono fare interessanti osservazioni sui dati!

Visualizzazione (seaborn)

Visualizzare DataFrame con pandas

- La libreria `seaborn`, simile a `matplotlib`, è molto comoda per visualizzare dati da `DataFrame` di `pandas`.
- Scelta la tipologia di grafico che si vuole utilizzare, la sintassi è sempre la stessa (riporto sotto un esempio con scatterplot).

```
> import seaborn as sns
> sns.scatterplot(data=data, x="Quantity", y="Profit",
                  hue="Sub-Category", col="Category")
```

dove:

- `data` è il nome del DataFrame in considerazione,
- `x` e `y` sono i nomi delle colonne di `data` da rappresentare sull'asse x e y, rispettivamente,
- `hue` è il nome della colonna (categorica) su cui i dati vengono fattorizzati,
- `col` è il nome della colonna (categorica) su cui il grafico viene spezzato in sotto-grafici.