

Progetto Dataset

Componenti del gruppo:

Justin Cadena
Francesco Miraglia
Zhou Zencheng

Indice:

1	Importo	3
2	Modifica	4
3	Salvataggio	6
4	Grafici correlazione & distribuzione	7
5	Missing values	10
6	Outliers	16
7	Encoding & Scaling	21
8	Splitting Dataset	30

Introduzione:

Questo progetto si concentra sull'analisi di un dataset attraverso una serie di fasi ben definite. Iniziamo importando un file CSV contenente i dati che intendiamo esaminare e analizzare. Successivamente, modifichiamo il dataset, apportando eventuali correzioni o pulizie necessarie per garantire la coerenza e la qualità dei dati.

Dopo aver preparato il dataset, passiamo all'analisi dei dati, utilizzando grafici di correlazione e distribuzione per comprendere meglio la struttura e le relazioni tra le variabili presenti nei dati. Questo ci permette di ottenere insight significativi prima di procedere con ulteriori manipolazioni dei dati.

Affrontiamo poi il processo di gestione dei valori mancanti e degli outlier. Identifichiamo i valori mancanti nel dataset e decidiamo come trattarli in modo appropriato per non compromettere l'analisi. Inoltre, individuiamo e gestiamo gli outlier presenti nei dati, poiché possono influenzare negativamente i risultati delle nostre analisi e dei modelli che costruiremo successivamente.

Successivamente, affrontiamo la codifica delle variabili categoriche e il ridimensionamento delle variabili, passaggi essenziali per preparare i dati per l'addestramento dei modelli di machine learning. La codifica delle variabili categoriche è necessaria perché molti algoritmi di machine learning richiedono dati numerici, mentre il ridimensionamento delle variabili può migliorare le prestazioni dei modelli riducendo la scala delle variabili.

Infine, dividiamo il dataset in set di addestramento e test, fondamentale per valutare l'efficacia dei modelli di machine learning e garantire che siano in grado di generalizzare bene su nuovi dati.

Il codice per questo progetto è organizzato in modo modulare e ben commentato per una migliore comprensione. Ogni fase del progetto è rappresentata da uno o più file di script Python, e richiede l'installazione di librerie come pandas, matplotlib, seaborn e scikit-learn.

Questo progetto offre una dimostrazione di come esplorare i dati, applicare tecniche di analisi e la manipolazione dei dati, e costruire modelli di machine learning per ottenere informazioni e prendere decisioni.

1 Importo

```
[1]: import pandas as pd

df = pd.read_csv(r'C:\\Users\\default\\titanic.csv')

print(df)
```

	passenger_id	pclass	name \
0	1216	3	Smyth, Miss. Julia
1	699	3	Cacic, Mr. Luka
2	1267	3	Van Impe, Mrs. Jean Baptiste (Rosalie Paula Go...
3	449	2	Hocking, Mrs. Elizabeth (Eliza Needs)
4	576	2	Veal, Mr. James
..
845	158	1	Hipkins, Mr. William Edward
846	174	1	Kent, Mr. Edward Austin
847	467	2	Kantor, Mrs. Sinai (Miriam Sternin)
848	1112	3	Peacock, Miss. Treasteall
849	425	2	Greenberg, Mr. Samuel

	sex	age	sibsp	parch	ticket	fare	cabin	embarked \
0	female	NaN	0	0	335432	7.7333	NaN	Q
1	male	38.0	0	0	315089	8.6625	NaN	S
2	female	30.0	1	1	345773	24.1500	NaN	S
3	female	54.0	1	3	29105	23.0000	NaN	S
4	male	40.0	0	0	28221	13.0000	NaN	S
..
845	male	55.0	0	0	680	50.0000	C39	S
846	male	58.0	0	0	11771	29.7000	B37	C
847	female	24.0	1	0	244367	26.0000	NaN	S
848	female	3.0	1	1	SOTON/O.Q. 3101315	13.7750	NaN	S
849	male	52.0	0	0	250647	13.0000	NaN	S

	boat	body	home.dest	survived
0	13	NaN	NaN	1
1	NaN	NaN	Croatia	0
2	NaN	NaN	NaN	0
3	4	NaN	Cornwall / Akron, OH	1
4	NaN	NaN	Barre, Co Washington, VT	0
..
845	NaN	NaN	London / Birmingham	0
846	NaN	258.0	Buffalo, NY	0
847	12	NaN	Moscow / Bronx, NY	1
848	NaN	NaN	NaN	0
849	NaN	19.0	Bronx, NY	0

[850 rows x 15 columns]

2 Modifica

```
[2]: # Rinomina una colonna
df = df.rename(columns={
    'passenger_id': 'ID Passeggero',
    'pclass': 'Classe',
    'name': 'Nome',
    'sex': 'Sesso',
    'age': 'Età',
    'sibsp': 'Figli',
    'parch': 'Parenti',
    'ticket': 'Ticket',
    'fare': 'Tariffa',
    'cabin': 'Cabina',
    'embarked': 'Porto Imbarco',
    'boat': 'Scialuppa di salvataggio',
    'body': 'Numero del corpo',
    'home.dest': 'Destinazione',
    'survived': 'Sopravvissuti',
})

# Aggiungi una nuova colonna
df.insert(15, 'Vivi', 0)
df.insert(loc=0, column='Numero Passeggeri', value=range(1, len(df) + 1))

# Elimina una colonna
df = df.drop(columns=['Numero del corpo'])
df = df.drop(columns=['Cabina'])

print(df)
```

	Numero Passeggeri	ID Passeggero	Classe	\
0	1	1216	3	
1	2	699	3	
2	3	1267	3	
3	4	449	2	
4	5	576	2	
...	
845	846	158	1	
846	847	174	1	
847	848	467	2	

848	849	1112	3
849	850	425	2

		Nome	Sesso	Età	Figli	\
0		Smyth, Miss. Julia	female	NaN	0	
1		Cacic, Mr. Luka	male	38.0	0	
2	Van Impe, Mrs. Jean Baptiste (Rosalie Paula Go...		female	30.0	1	
3	Hocking, Mrs. Elizabeth (Eliza Needs)		female	54.0	1	
4	Veal, Mr. James		male	40.0	0	
..	
845	Hipkins, Mr. William Edward		male	55.0	0	
846	Kent, Mr. Edward Austin		male	58.0	0	
847	Kantor, Mrs. Sinai (Miriam Sternin)		female	24.0	1	
848	Peacock, Miss. Treasteall		female	3.0	1	
849	Greenberg, Mr. Samuel		male	52.0	0	

	Parenti	Ticket	Tariffa	Porto	Imbarco	\
0	0	335432	7.7333		Q	
1	0	315089	8.6625		S	
2	1	345773	24.1500		S	
3	3	29105	23.0000		S	
4	0	28221	13.0000		S	
..	
845	0	680	50.0000		S	
846	0	11771	29.7000		C	
847	0	244367	26.0000		S	
848	1 SOTON/O.Q.	3101315	13.7750		S	
849	0	250647	13.0000		S	

	Scialuppa di salvataggio	Destinazione	Sopravvissuti	Vivi
0	13	NaN	1	0
1	NaN	Croatia	0	0
2	NaN	NaN	0	0
3	4	Cornwall / Akron, OH	1	0
4	NaN	Barre, Co Washington, VT	0	0
..
845	NaN	London / Birmingham	0	0
846	NaN	Buffalo, NY	0	0
847	12	Moscow / Bronx, NY	1	0
848	NaN	NaN	0	0
849	NaN	Bronx, NY	0	0

[850 rows x 15 columns]

3 Salvataggio

```
[3]: #Salva il file con le modifiche
df.to_csv(r'C:\\atmosphere\\titanic_saved.csv', index=False)

# Verifica del file salvato
df_verifica = pd.read_csv(r'C:\\atmosphere\\titanic_saved.csv')
print(df_verifica.head())
```

	Numero Passeggeri	ID Passeggero	Classe	\
0	1	1216	3	
1	2	699	3	
2	3	1267	3	
3	4	449	2	
4	5	576	2	

	Nome	Sesso	Età	Figli	\
0	Smyth, Miss. Julia	female	NaN	0	
1	Cacic, Mr. Luka	male	38.0	0	
2	Van Impe, Mrs. Jean Baptiste (Rosalie Paula Go...	female	30.0	1	
3	Hocking, Mrs. Elizabeth (Eliza Needs)	female	54.0	1	
4	Veal, Mr. James	male	40.0	0	

	Parenti	Ticket	Tariffa	Porto Imbarco	Scialuppa di salvataggio	\
0	0	335432	7.7333	Q	13	
1	0	315089	8.6625	S	NaN	
2	1	345773	24.1500	S	NaN	
3	3	29105	23.0000	S	4	
4	0	28221	13.0000	S	NaN	

	Destinazione	Sopravvissuti	Vivi
0	NaN	1	0
1	Croatia	0	0
2	NaN	0	0
3	Cornwall / Akron, OH	1	0
4	Barre, Co Washington, VT	0	0

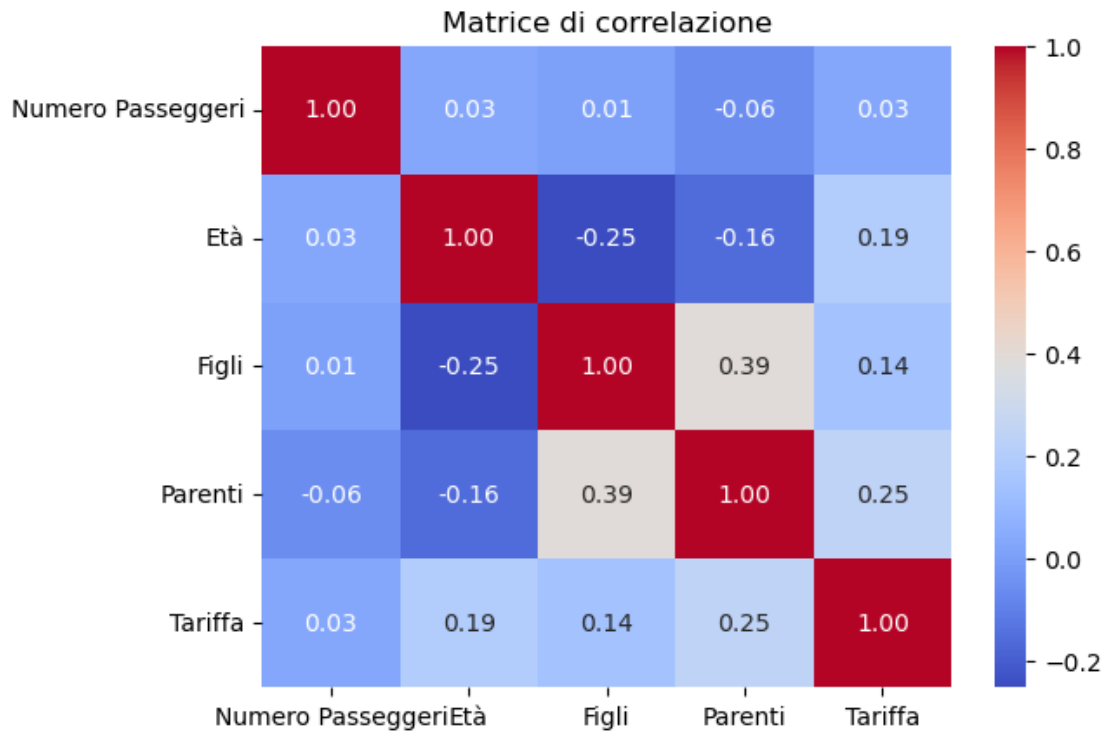
4 Grafici correlazione & distribuzione

```
[4]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Seleziona le colonne desiderate
colonne_selezionate = ['Numero Passeggeri', 'Età', 'Figli', 'Parenti', 'Tariffa']
subset_df = df[colonne_selezionate]

# Matrice di correlazione
correlation_matrix = subset_df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Matrice di correlazione")
plt.show()

# Scatter plot dispersione utilizzando i dati dal dataframe
plt.figure(figsize=(10, 6))
plt.scatter(subset_df['Numero Passeggeri'], subset_df['Età'],
            ↪subset_df['Classe'], label='Numero Passeggeri', color='black', alpha=0.7)
plt.scatter(subset_df['Numero Passeggeri'], subset_df['Età'],
            ↪subset_df['Classe'], label='Età', color='blue', alpha=0.7)
plt.scatter(subset_df['Numero Passeggeri'], subset_df['Età'],
            ↪subset_df['Classe'], label='Classe', color='orange', alpha=0.7)
plt.xlabel('Numero Passeggeri')
plt.ylabel('Età')
plt.title('Relazione tra Numero Passeggeri e Età')
plt.legend()
plt.grid(True)
plt.show()
```

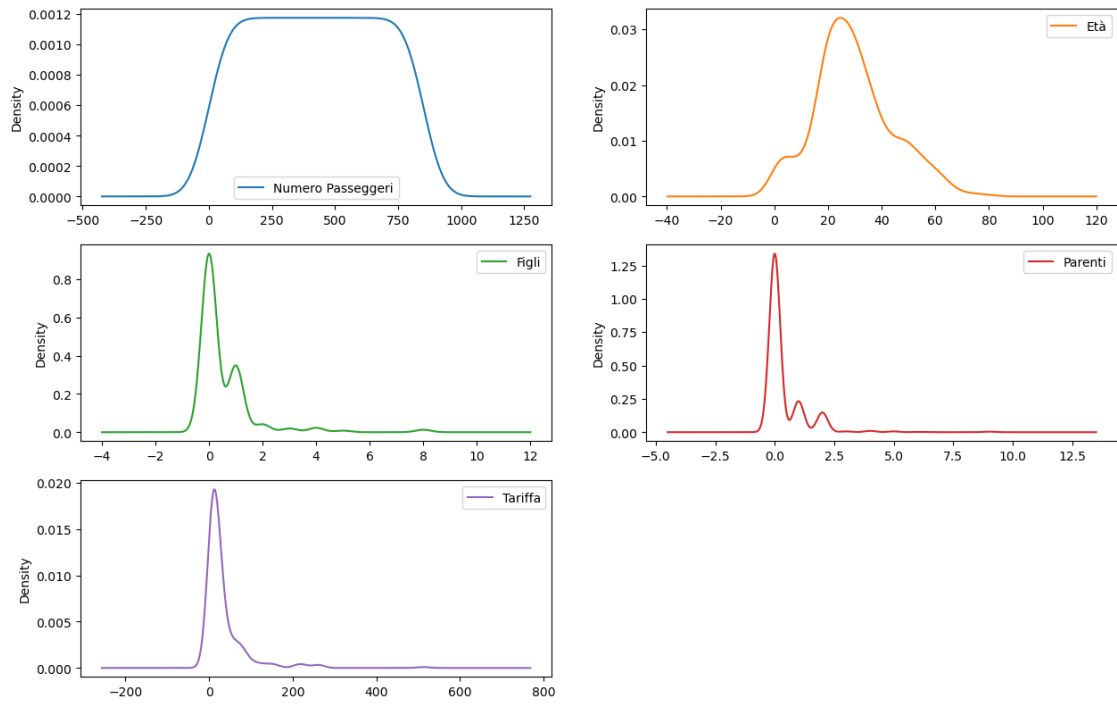


<Figure size 1000x600 with 0 Axes>

```
[5]: # Plotting delle distribuzioni delle colonne
plt.figure(figsize=(6, 5))
subset_df.plot(kind='density', subplots=True, layout=(6, 2), sharex=False,
               ↳figsize=(15, 20))
plt.suptitle("Distribuzioni delle colonne")
plt.show()
```

<Figure size 600x500 with 0 Axes>

Distribuzioni delle colonne



5 Missing values

```
[6]: df.isna().any
```

```
[6]: <bound method NDFrame._add_numeric_operations.<locals>.any of
Passeggeri ID Passeggero Classe Nome Sesso Età Figli \
0          False          False False False False False True False
1          False          False False False False False False False
2          False          False False False False False False False
3          False          False False False False False False False
4          False          False False False False False False False
..          ...          ...    ...    ...    ...    ...    ...
845         False          False False False False False False False
846         False          False False False False False False False
847         False          False False False False False False False
848         False          False False False False False False False
849         False          False False False False False False False
```

```

Parenti Ticket Tariffa Porto Imbarco Scialuppa di salvataggio \
0   False  False   False          False          False
1   False  False   False          False          True
2   False  False   False          False          True
3   False  False   False          False          False
4   False  False   False          False          True
..   ...    ...    ...          ...          ...
845  False  False   False          False          True
846  False  False   False          False          True
847  False  False   False          False          False
848  False  False   False          False          True
849  False  False   False          False          True
```

```

Destinazione Sopravvissuti Vivi
0          True          False False
1         False          False False
2          True          False False
3         False          False False
4         False          False False
..          ...          ...    ...
845        False          False False
846        False          False False
847        False          False False
848          True          False False
849        False          False False
```

```
[850 rows x 15 columns]>
```

```
[7]: df.isna().sum()
```

```
[7]: Numero Passeggeri      0
     ID Passeggero          0
     Classe                 0
     Nome                   0
     Sesso                  0
     Età                    174
     Figli                   0
     Parenti                 0
     Ticket                  0
     Tariffa                  1
     Porto Imbarco           1
     Scialuppa di salvataggio 542
     Destinazione            386
     Sopravvissuti           0
     Vivi                   0
     dtype: int64
```

```
[8]: missing_percent = (df.isnull().sum() / len(df)) * 100
     missing_percent
```

```
[8]: Numero Passeggeri      0.000000
     ID Passeggero          0.000000
     Classe                 0.000000
     Nome                   0.000000
     Sesso                  0.000000
     Età                    20.470588
     Figli                   0.000000
     Parenti                 0.000000
     Ticket                  0.000000
     Tariffa                  0.117647
     Porto Imbarco           0.117647
     Scialuppa di salvataggio 63.764706
     Destinazione            45.411765
     Sopravvissuti           0.000000
     Vivi                   0.000000
     dtype: float64
```

```
[9]: import matplotlib.pyplot as plt

     # Creazione della lista dei colori per le barre, utilizzando 'lightblue' se la
     ↳ percentuale di valori mancanti è inferiore a 0.5, altrimenti 'lightcoral'
     colors = ['lightblue' if val < 0.5 else 'lightcoral' for val in missing_percent]

     # Creazione di una nuova figura con dimensioni specifiche
     plt.figure(figsize=(10, 6))
```

```

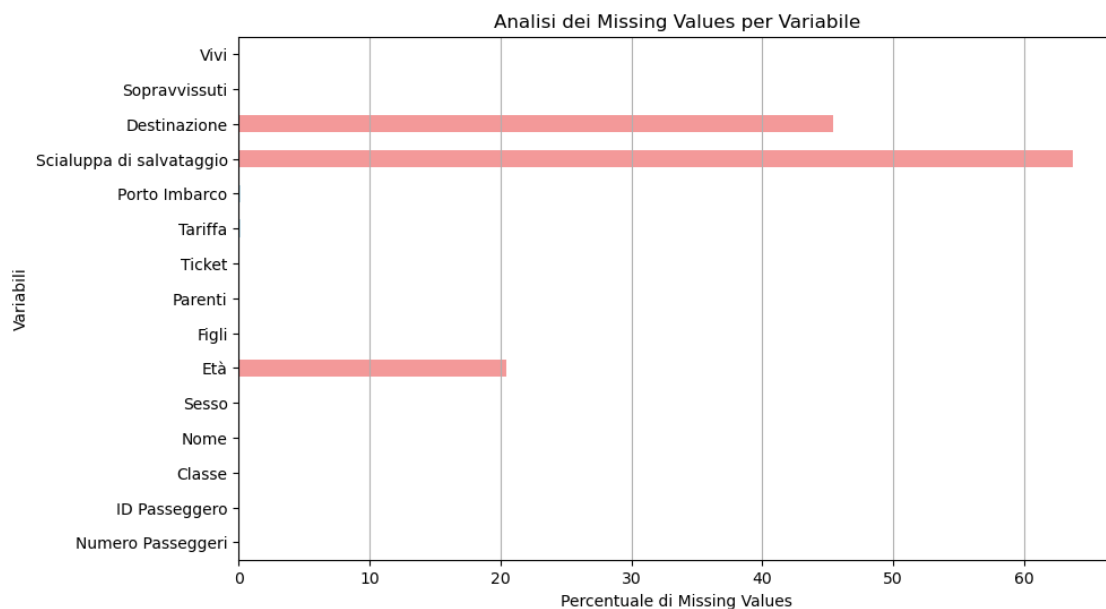
# Creazione del grafico a barre orizzontali, con colori definiti dalla lista
↳ 'colors' e trasparenza impostata a 0.8
missing_percent.plot(kind='barh', color=colors, alpha=0.8)

# Aggiunta di etichette agli assi x e y del grafico e un titolo
plt.xlabel('Percentuale di Missing Values')
plt.ylabel('Variabili')
plt.title('Analisi dei Missing Values per Variabile')

# Aggiunta di una griglia solo sull'asse x del grafico
plt.grid(axis='x')

# Visualizzazione del grafico
plt.show()

```



```

[10]: import pandas as pd

# Carica il dataset
df = pd.read_csv(r'C:\\atmosphere\\titanic_saved.csv')

# Converti la colonna 'Scialuppa di salvataggio' in numeri, ignorando i valori
↳ non convertibili
df['Scialuppa di salvataggio'] = pd.to_numeric(df['Scialuppa di salvataggio'],
↳ errors='coerce')

```

```

# Calcola la media della colonna 'Età' e 'Scialuppa di salvataggio'
eta_media = df['Età'].mean()
scialuppa_media = df['Scialuppa di salvataggio'].mean()

print("Media dell'età:", eta_media)
print("Media della scialuppa di salvataggio:", scialuppa_media)

# Elimina le righe con valori mancanti nelle colonne 'Destinazione', 'Tariffa' e
↳ 'Porto Imbarco'
df.dropna(subset=['Destinazione', 'Tariffa', 'Porto Imbarco'], inplace=True)

# Sostituisci i valori NaN con punti interrogativi
df.fillna('?', inplace=True)

# Visualizza il dataframe senza le righe contenenti NaN nelle colonne specificate
print(df)

```

Media dell'età: 29.519847189349115

Media della scialuppa di salvataggio: 9.421686746987952

	Numero Passeggeri	ID Passeggero	Classe \
1	2	699	3
3	4	449	2
4	5	576	2
7	8	560	2
10	11	313	1
..
844	845	165	1
845	846	158	1
846	847	174	1
847	848	467	2
849	850	425	2

	Nome	Sesso	Età	Figli	Parenti \
1	Cacic, Mr. Luka	male	38.0	0	0
3	Hocking, Mrs. Elizabeth (Eliza Needs)	female	54.0	1	3
4	Veal, Mr. James	male	40.0	0	0
7	Sinkkonen, Miss. Anna	female	30.0	0	0
10	Widener, Mr. Harry Elkins	male	27.0	0	2
..
844	Hoyt, Mr. Frederick Maxfield	male	38.0	1	0
845	Hipkins, Mr. William Edward	male	55.0	0	0
846	Kent, Mr. Edward Austin	male	58.0	0	0
847	Kantor, Mrs. Sinai (Miriam Sternin)	female	24.0	1	0
849	Greenberg, Mr. Samuel	male	52.0	0	0

	Ticket	Tariffa	Porto Imbarco	Scialuppa di salvataggio \
1	315089	8.6625	S	?
3	29105	23.0000	S	4.0

4	28221	13.0000	S	?
7	250648	13.0000	S	10.0
10	113503	211.5000	C	?
..
846	11771	29.7000	C	?
847	244367	26.0000	S	12.0
849	250647	13.0000	S	?

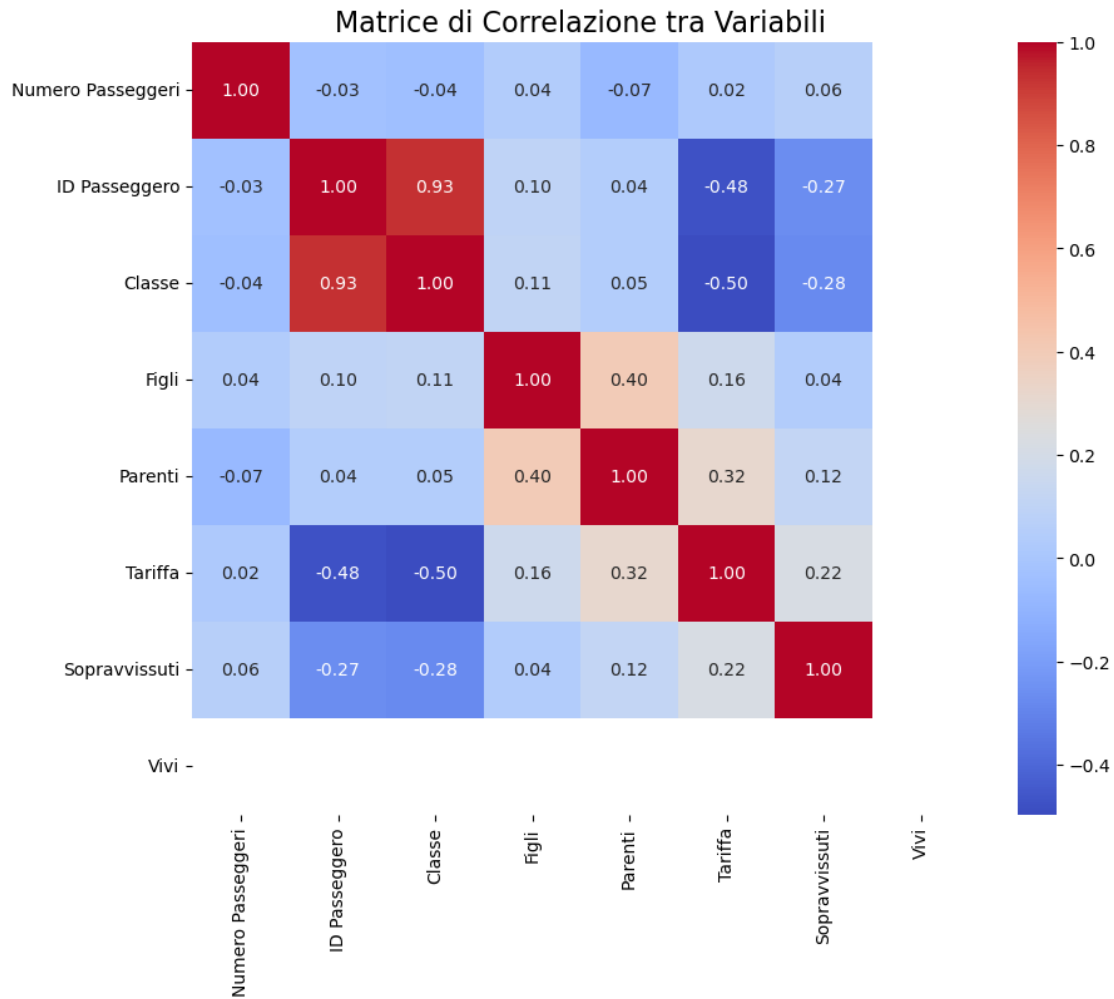
	Destinazione	Sopravvissuti	Vivi
1	Croatia	0	0
3	Cornwall / Akron, OH	1	0
4	Barre, Co Washington, VT	0	0
7	Finland / Washington, DC	1	0
10	Elkins Park, PA	0	0
..
846	Buffalo, NY	0	0
847	Moscow / Bronx, NY	1	0
849	Bronx, NY	0	0

[464 rows x 15 columns]

```
[11]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Calcola la matrice di correlazione
corr_matrix = df.corr()

# Crea una heatmap della matrice di correlazione
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Matrice di Correlazione tra Variabili', fontsize=16)
plt.show()
```



```
[12]: df.isna().sum()
```

```
[12]: Numero Passeggeri      0
      ID Passeggero          0
      Classe                 0
      Nome                   0
      Sesso                  0
      Età                    0
      Figli                  0
      Parenti                0
      Ticket                 0
      Tariffa                 0
      Porto Imbarco          0
      Scialuppa di salvataggio 0
      Destinazione           0
      Sopravvissuti          0
```

Vivi
dtype: int64

0

6 Outliers

```
[13]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv(r'C:\\atmosphere\\titanic_saved.csv')

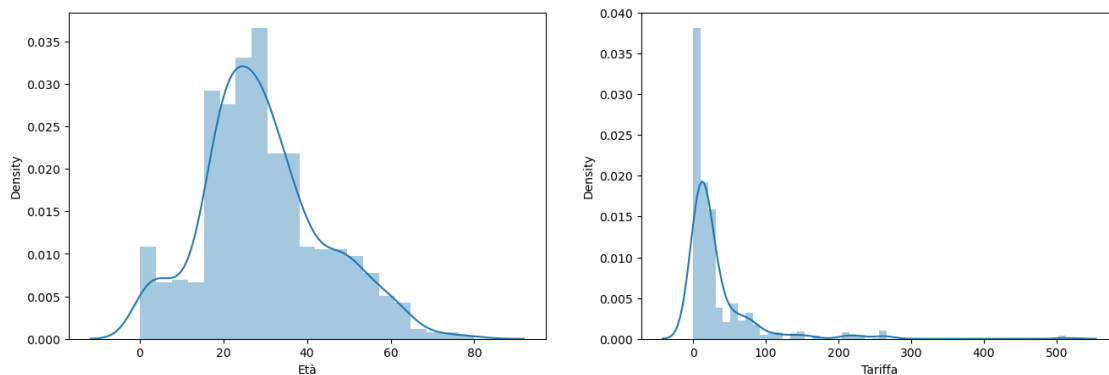
# Crea una nuova figura con dimensioni specifiche
plt.figure(figsize=(16,5))

# Crea un subplot con una riga e due colonne, e seleziona il primo subplot
plt.subplot(1,2,1)

# Utilizza seaborn per creare un grafico della distribuzione dell'età
sns.distplot(df['Età'])

# Sostituisci i valori mancanti nell'intero DataFrame con 0
df.fillna(0, inplace=True)

# Crea il secondo subplot
plt.subplot(1,2,2)
sns.distplot(df['Tariffa'])
plt.show()
```



```
[14]: print("Media Et ",df['Et '].mean())
print("Deviazione Standart Et ", df['Et '].std())
```



```
print("Media Tariffa",df['Tariffa'].mean())
print("Deviazione Standart Tariffa", df['Tariffa'].std())
```

```
Media Età 23.476960823529414
Deviazione Standart Età 17.624806369856714
Media Tariffa 33.972686
Deviazione Standart Tariffa 53.68681811921515
```

```
[15]: import pandas as pd

# Calcola i limiti superiori e inferiori per identificare gli outlier
upper_limit = df['Età'].mean() + 3 * df['Età'].std()
lower_limit = df['Età'].mean() - 3 * df['Età'].std()

# Seleziona le righe che sono al di fuori dei limiti
outliers_eta = df[(df['Età'] > upper_limit) | (df['Età'] < lower_limit)]
outliers_eta
```

```
[15]:      Numero Passeggeri  ID Passeggero  Classe \
177                178                14        1

      Nome Sesso  Età  Figli  Parenti Ticket \
177  Barkworth, Mr. Algernon Henry Wilson  male  80.0        0        0  27042

      Tariffa Porto Imbarco Scialuppa di salvataggio  Destinazione \
177      30.0                S                B  Hessle, Yorks

      Sopravvissuti  Vivi
177                1      0
```

```
[16]: import pandas as pd

# Calcola i limiti superiori e inferiori per identificare gli outlier
upper_limit = df['Tariffa'].mean() + 3 * df['Tariffa'].std()
lower_limit = df['Tariffa'].mean() - 3 * df['Tariffa'].std()

# Seleziona le righe che sono al di fuori dei limiti
outliers_tariffa = df[(df['Tariffa'] > upper_limit) | (df['Tariffa'] <
↳lower_limit)]
outliers_tariffa
```

```
[16]:      Numero Passeggeri  ID Passeggero  Classe \
10                11                313        1
37                38                249        1
60                61                 50        1
62                63                173        1
169               170                193        1
171               172                180        1
```

229	230	112	1
231	232	10	1
247	248	115	1
311	312	250	1
339	340	66	1
369	370	251	1
374	375	23	1
403	404	17	1
436	437	35	1
464	465	285	1
521	522	237	1
623	624	286	1
649	650	11	1
710	711	129	1
714	715	116	1
719	720	314	1
766	767	24	1
790	791	183	1
796	797	49	1
804	805	312	1
826	827	253	1

	Nome	Sesso	Età	Figli \
10	Widener, Mr. Harry Elkins	male	27.0	0
37	Ryerson, Master. John Borie	male	13.0	2
60	Cardeza, Mrs. James Warburton Martinez (Charlo...	female	58.0	0
62	Keeping, Mr. Edwin	male	32.5	0
169	Madill, Miss. Georgette Alexandra	female	15.0	0
171	Kreuchen, Miss. Emilie	female	39.0	0
229	Fortune, Miss. Ethel Flora	female	28.0	3
231	Astor, Col. John Jacob	male	47.0	1
247	Fortune, Mr. Mark	male	64.0	1
311	Ryerson, Miss. Emily Borie	female	18.0	2
339	Chaudanson, Miss. Victorine	female	36.0	0
369	Ryerson, Miss. Susan Parker "Suzette"	female	21.0	2
374	Bidois, Miss. Rosalie	female	42.0	0
403	Baxter, Mrs. James (Helene DeLaudeniére Chaput)	female	50.0	0
436	Bowen, Miss. Grace Scott	female	45.0	0
464	Straus, Mr. Isidor	male	67.0	1
521	Robbins, Mr. Victor	male	0.0	0
623	Straus, Mrs. Isidor (Rosalie Ida Blun)	female	63.0	1
649	Astor, Mrs. John Jacob (Madeleine Talmadge Force)	female	18.0	1
710	Geiger, Miss. Amalie	female	35.0	0
714	Fortune, Mrs. Mark (Mary McDougald)	female	60.0	1
719	Widener, Mrs. George Dunton (Eleanor Elkins)	female	50.0	1
766	Bird, Miss. Ellen	female	29.0	0
790	Lesurer, Mr. Gustave J	male	35.0	0

796	Cardeza, Mr. Thomas Drake Martinez	male	36.0	0
804	Widener, Mr. George Dunton	male	50.0	1
826	Ryerson, Mrs. Arthur Larned (Emily Maria Borie)	female	48.0	1

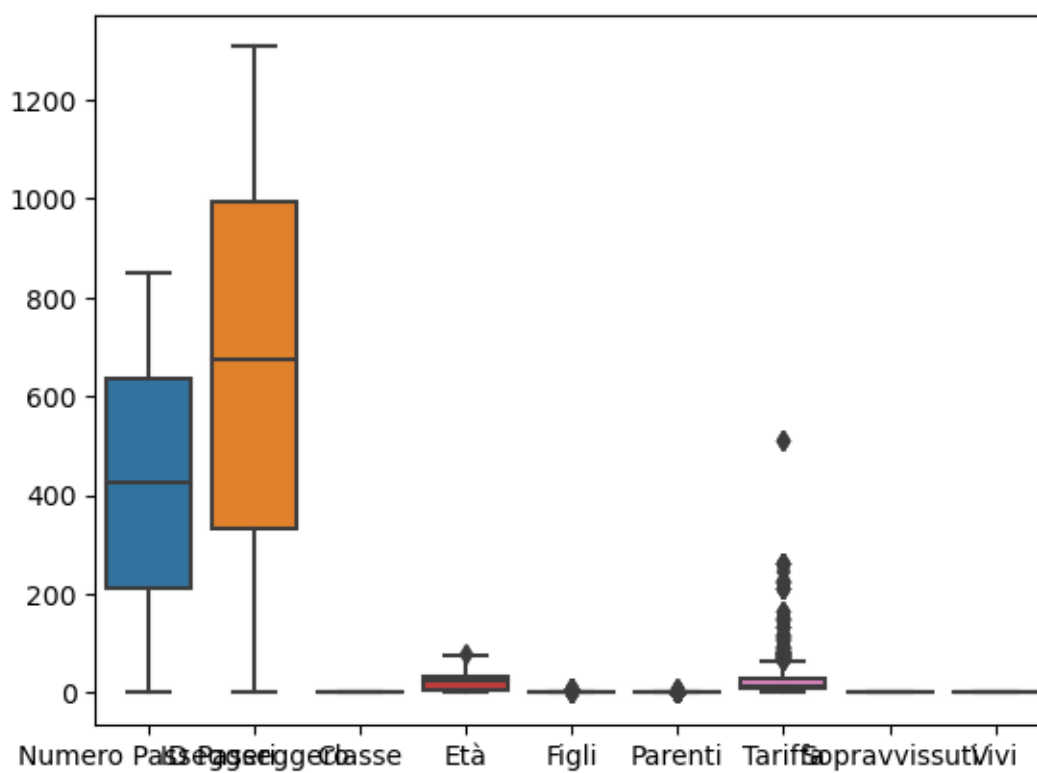
	Parenti	Ticket	Tariffa Porto Imbarco	Scialuppa di salvataggio	\
10	2	113503	211.5000	C	0
37	2	PC 17608	262.3750	C	4
60	1	PC 17755	512.3292	C	3
62	0	113503	211.5000	C	0
169	1	24160	211.3375	S	2
171	0	24160	211.3375	S	2
229	2	19950	263.0000	S	10
231	0	PC 17757	227.5250	C	0
247	4	19950	263.0000	S	0
311	2	PC 17608	262.3750	C	4
339	0	PC 17608	262.3750	C	4
369	2	PC 17608	262.3750	C	4
374	0	PC 17757	227.5250	C	4
403	1	PC 17558	247.5208	C	6
436	0	PC 17608	262.3750	C	4
464	0	PC 17483	221.7792	S	0
521	0	PC 17757	227.5250	C	0
623	0	PC 17483	221.7792	S	0
649	0	PC 17757	227.5250	C	4
710	0	113503	211.5000	C	4
714	4	19950	263.0000	S	10
719	1	113503	211.5000	C	4
766	0	PC 17483	221.7792	S	8
790	0	PC 17755	512.3292	C	3
796	1	PC 17755	512.3292	C	3
804	1	113503	211.5000	C	0
826	3	PC 17608	262.3750	C	4

	Destinazione	Sopravvissuti	Vivi
10	Elkins Park, PA	0	0
37	Haverford, PA / Cooperstown, NY	1	0
60	Germantown, Philadelphia, PA	1	0
62	0	0	0
169	St Louis, MO	1	0
171	0	1	0
229	Winnipeg, MB	1	0
231	New York, NY	0	0
247	Winnipeg, MB	0	0
311	Haverford, PA / Cooperstown, NY	1	0
339	0	1	0
369	Haverford, PA / Cooperstown, NY	1	0
374	0	1	0

403		Montreal, PQ	1	0
436		Cooperstown, NY	1	0
464		New York, NY	0	0
521		0	0	0
623		New York, NY	0	0
649		New York, NY	1	0
710		0	1	0
714		Winnipeg, MB	1	0
719		Elkins Park, PA	1	0
766		0	1	0
790		0	1	0
796	Austria-Hungary / Germantown, Philadelphia, PA		1	0
804		Elkins Park, PA	0	0
826		Haverford, PA / Cooperstown, NY	1	0

```
[17]: sns.boxplot(df)
```

```
[17]: <Axes: >
```



7 Encoding & Scaling

```
[18]: from sklearn.preprocessing import StandardScaler
import pandas as pd

new_data = df[['Numero Passeggeri', 'ID Passeggero', 'Classe', 'Nome', 'Sesso',
↳'Età', 'Figli', 'Parenti', 'Ticket', 'Tariffa', 'Porto Imbarco', 'Scialuppa di
↳salvataggio', 'Destinazione', 'Sopravvissuti', 'Vivi']]
new_data.head()
```

```
[18]:
```

	Numero Passeggeri	ID Passeggero	Classe	\
0	1	1216	3	
1	2	699	3	
2	3	1267	3	
3	4	449	2	
4	5	576	2	

	Nome	Sesso	Età	Figli	\
0	Smyth, Miss. Julia	female	0.0	0	
1	Cacic, Mr. Luka	male	38.0	0	
2	Van Impe, Mrs. Jean Baptiste (Rosalie Paula Go...	female	30.0	1	
3	Hocking, Mrs. Elizabeth (Eliza Needs)	female	54.0	1	
4	Veal, Mr. James	male	40.0	0	

	Parenti	Ticket	Tariffa	Porto Imbarco	Scialuppa di salvataggio	\
0	0	335432	7.7333	Q		13
1	0	315089	8.6625	S		0
2	1	345773	24.1500	S		0
3	3	29105	23.0000	S		4
4	0	28221	13.0000	S		0

	Destinazione	Sopravvissuti	Vivi
0	0	1	0
1	Croatia	0	0
2	0	0	0
3	Cornwall / Akron, OH	1	0
4	Barre, Co Washington, VT	0	0

```
[19]: from sklearn.preprocessing import LabelEncoder

lb = LabelEncoder() # Crea un'istanza della classe LabelEncoder

# Definizione di una funzione personalizzata per codificare una colonna usando
↳LabelEncoder
def myfuc(x, c):
    x = x.astype(str) # Converti tutti i valori della colonna in stringhe
```

```

    m = lb.fit(x) # Addestra il LabelEncoder sulla colonna
    p = lb.transform(x) # Trasforma i valori della colonna
    df = pd.DataFrame(p, columns=[c]) # Crea un DataFrame con i valori
    ↪trasformati
    return df

mylist = [] # Inizializza una lista vuota per contenere i DataFrame trasformati

# Iterazione su tutte le colonne del DataFrame 'new_data'
for item in new_data.columns:
    new = myfuc(new_data[item], item) # Applica la funzione personalizzata a
    ↪ciascuna colonna
    mylist.append(new) # Aggiungi il DataFrame risultante alla lista 'mylist'

mylist = tuple(mylist) # Converti la lista in una tupla
new_encode = pd.concat(mylist, axis=1) # Concatena tutti i DataFrame nella
    ↪tupla lungo l'asse delle colonne

new_encode # Mostra il DataFrame risultante

```

```

[19]:
    Numero Passeggeri  ID Passeggero  Classe  Nome  Sesso  Età  Figli  \
0                0          159        2   736      0    0      0
1               111          626        2   123      1   45      0
2               222          197        2   783      0   35      1
3               333          452        1   360      0   66      1
4               444          534        1   792      1   49      0
..              ...          ...      ...   ...      ...   ...      ...
845             829          246        0   353      1   67      0
846             830          260        0   426      1   71      0
847             831          467        1   411      0   26      1
848             832           79        2   613      0   34      1
849             834          436        1   309      1   64      0

```

```

    Parenti  Ticket  Tariffa  Porto Imbarco  Scialuppa di salvataggio  \
0         0     283     177             2              5
1         0     257     212             3              0
2         1     307      77             3              0
3         3     237      73             3             14
4         0     228      20             3              0
..        ...     ...     ...           ...             ...
845        0     488     130             3              0
846        0      58      99             1              0
847        0     145      84             3              4
848        1     630      23             3              0
849        0     160      20             3              0

```

```

    Destinazione  Sopravvissuti  Vivi

```

0	0	1	0
1	57	0	0
2	0	0	0
3	53	1	0
4	12	0	0
..
845	140	0	0
846	36	0	0
847	167	1	0
848	0	0	0
849	28	0	0

[850 rows x 15 columns]

```
[21]: import pandas as pd
import matplotlib.pyplot as plt

# Seleziona le colonne di interesse nel DataFrame 'df'
selected_columns = ['Numero Passeggeri', 'Classe', 'Età', 'Tariffa',
                    ↳ 'Sopravvissuti']

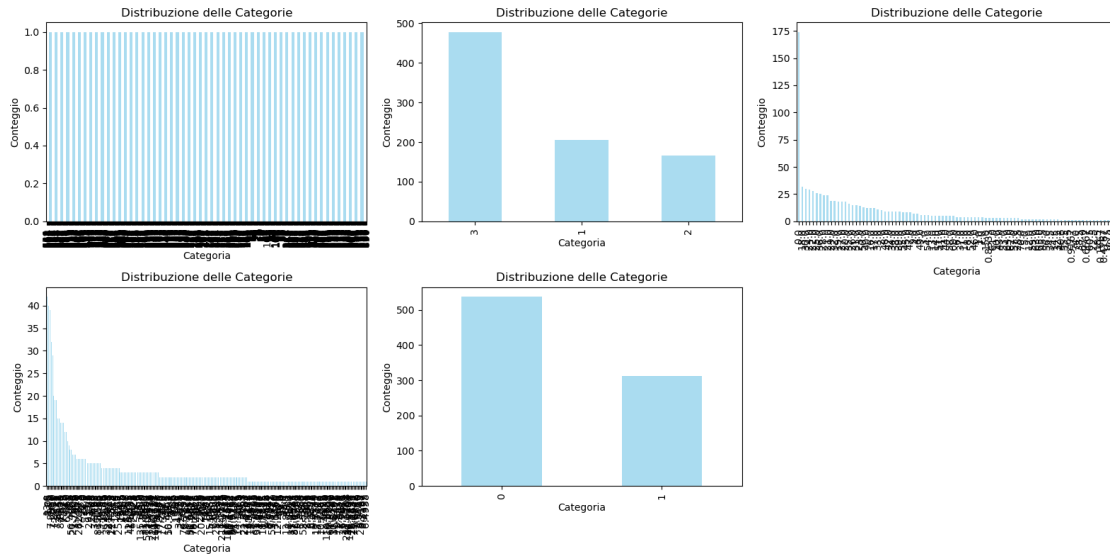
# Inizializza un dizionario per memorizzare il conteggio delle diverse categorie
↳ per ogni colonna selezionata
category_counts = {}

# Calcola il conteggio delle diverse categorie per ogni colonna selezionata
for col in selected_columns:
    category_counts[col] = df[col].value_counts()

# Crea un nuovo grafico con dimensioni specificate
plt.figure(figsize=(16, 8))

# Itera su tutte le colonne selezionate e crea un subplot per ciascuna colonna
for i, col in enumerate(selected_columns):
    plt.subplot(2, 3, i+1) # Crea un subplot in una griglia 2x3, selezionando
↳ l'indice 'i+1'
    category_counts[col].plot(kind='bar', color='skyblue', alpha=0.7, label=col)
↳ # Crea un grafico a barre per il conteggio delle categorie
    plt.title('Distribuzione delle Categorie') # Aggiungi un titolo al subplot
    plt.xlabel('Categoria') # Aggiungi un'etichetta all'asse x
    plt.ylabel('Conteggio') # Aggiungi un'etichetta all'asse y

plt.tight_layout() # Ottimizza la disposizione dei subplot
plt.show()
```



```
[22]: from sklearn.preprocessing import StandardScaler

# Seleziona solo le colonne numeriche dal DataFrame
numeric_columns = df.select_dtypes(include=['float64', 'int64']).columns

# Inizializza lo scaler
st = StandardScaler()

# Adatta lo scaler ai dati delle colonne numeriche
st.fit(df[numeric_columns])

# Esegui lo scaling dei dati
scaled_data = st.transform(df[numeric_columns])

# Crea un nuovo DataFrame con i dati scalati
scaled_df = pd.DataFrame(scaled_data, columns=numeric_columns)

print("DataFrame originale:")
print(df)
print("\nDataFrame con dati scalati:")
print(scaled_df)
```

DataFrame originale:

	Numero Passeggeri	ID Passeggero	Classe \
0	1	1216	3
1	2	699	3
2	3	1267	3
3	4	449	2
4	5	576	2

..
845	846	158	1
846	847	174	1
847	848	467	2
848	849	1112	3
849	850	425	2

		Nome	Sesso	Età	Figli	\
0		Smyth, Miss. Julia	female	0.0	0	
1		Cacic, Mr. Luka	male	38.0	0	
2	Van Impe, Mrs. Jean Baptiste (Rosalie Paula Go...		female	30.0	1	
3	Hocking, Mrs. Elizabeth (Eliza Needs)		female	54.0	1	
4	Veal, Mr. James		male	40.0	0	
..		
845	Hipkins, Mr. William Edward		male	55.0	0	
846	Kent, Mr. Edward Austin		male	58.0	0	
847	Kantor, Mrs. Sinai (Miriam Sternin)		female	24.0	1	
848	Peacock, Miss. Treasteall		female	3.0	1	
849	Greenberg, Mr. Samuel		male	52.0	0	

	Parenti	Ticket	Tariffa	Porto	Imbarco	\
0	0	335432	7.7333		Q	
1	0	315089	8.6625		S	
2	1	345773	24.1500		S	
3	3	29105	23.0000		S	
4	0	28221	13.0000		S	
..	
845	0	680	50.0000		S	
846	0	11771	29.7000		C	
847	0	244367	26.0000		S	
848	1 SOTON/O.Q.	3101315	13.7750		S	
849	0	250647	13.0000		S	

	Scialuppa di salvataggio	Destinazione	Sopravvissuti	Vivi
0	13	0	1	0
1	0	Croatia	0	0
2	0	0	0	0
3	4	Cornwall / Akron, OH	1	0
4	0	Barre, Co Washington, VT	0	0
..
845	0	London / Birmingham	0	0
846	0	Buffalo, NY	0	0
847	12	Moscow / Bronx, NY	1	0
848	0	0	0	0
849	0	Bronx, NY	0	0

[850 rows x 15 columns]

DataFrame con dati scalati:

	Numero Passeggeri	ID Passeggero	Classe	Età	Figli	Parenti	\
0	-1.730014	1.453727	0.811420	-1.332825	-0.469963	-0.434989	
1	-1.725939	0.095088	0.811420	0.824496	-0.469963	-0.434989	
2	-1.721863	1.587751	0.811420	0.370323	0.429741	0.702675	
3	-1.717788	-0.561894	-0.381845	1.732842	0.429741	2.978005	
4	-1.713713	-0.228147	-0.381845	0.938040	-0.469963	-0.434989	
..	
845	1.713713	-1.326621	-1.575110	1.789614	-0.469963	-0.434989	
846	1.717788	-1.284575	-1.575110	1.959929	-0.469963	-0.434989	
847	1.721863	-0.514592	-0.381845	0.029694	0.429741	-0.434989	
848	1.725939	1.180422	0.811420	-1.162510	0.429741	0.702675	
849	1.730014	-0.624965	-0.381845	1.619299	-0.469963	-0.434989	

	Tariffa	Sopravvissuti	Vivi
0	-0.489037	1.309830	0.0
1	-0.471719	-0.763458	0.0
2	-0.183070	-0.763458	0.0
3	-0.204504	1.309830	0.0
4	-0.390879	-0.763458	0.0
..
845	0.298709	-0.763458	0.0
846	-0.079632	-0.763458	0.0
847	-0.148591	1.309830	0.0
848	-0.376435	-0.763458	0.0
849	-0.390879	-0.763458	0.0

[850 rows x 9 columns]

```
[23]: import matplotlib.pyplot as plt

# Visualizzazione delle distribuzioni delle colonne selezionate prima dello
↳scaling
selected_columns = ['Età', 'Tariffa']

# Crea una nuova figura con dimensioni specificate
plt.figure(figsize=(6, 6))

# Itera su tutte le colonne selezionate e crea un subplot per ciascuna colonna
for i, col in enumerate(selected_columns):
    plt.subplot(2, len(selected_columns)//2, i+1) # Crea un subplot in una
↳griglia 2x2, selezionando l'indice 'i+1'
    plt.hist(df[col], bins=20, color='blue', alpha=0.5) # Crea un istogramma
↳dei valori della colonna
    plt.title(col + ' (Prima dello Scaling)') # Aggiungi un titolo al subplot
    plt.xlabel(col) # Aggiungi un'etichetta all'asse x
    plt.ylabel('Frequenza') # Aggiungi un'etichetta all'asse y
```

```

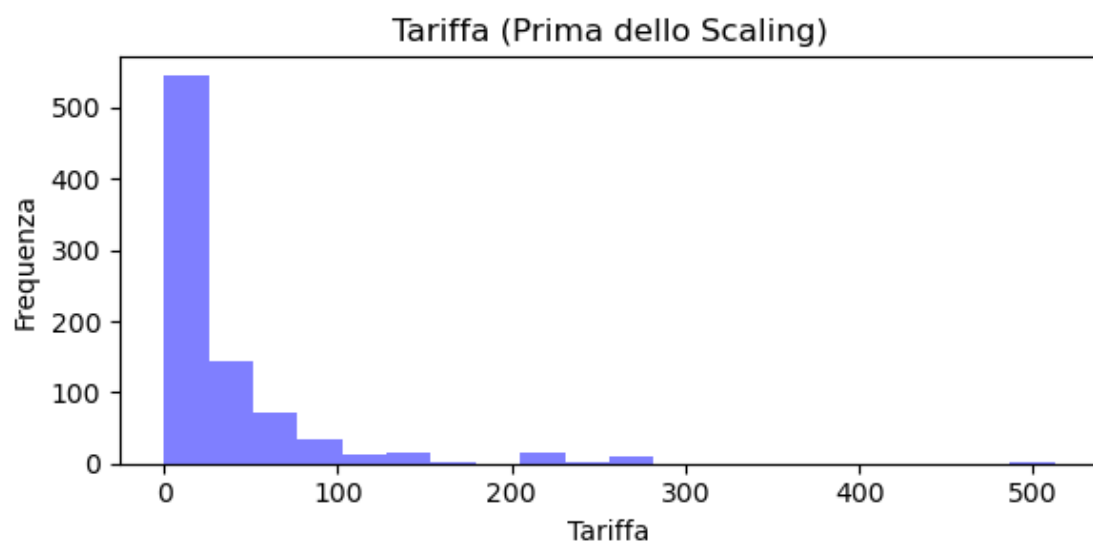
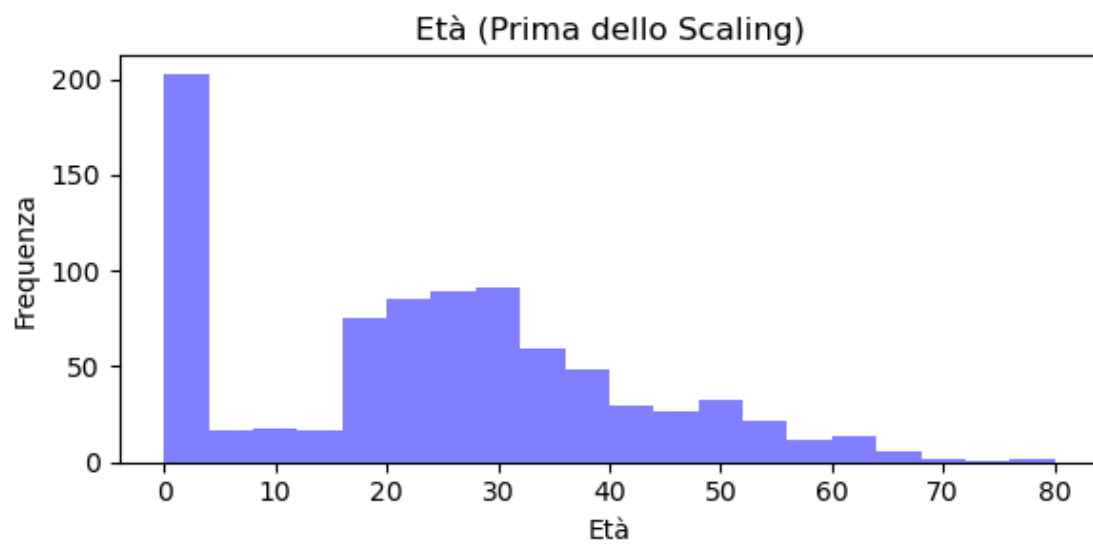
plt.tight_layout() # Ottimizza la disposizione dei subplot
plt.show() # Mostra il grafico

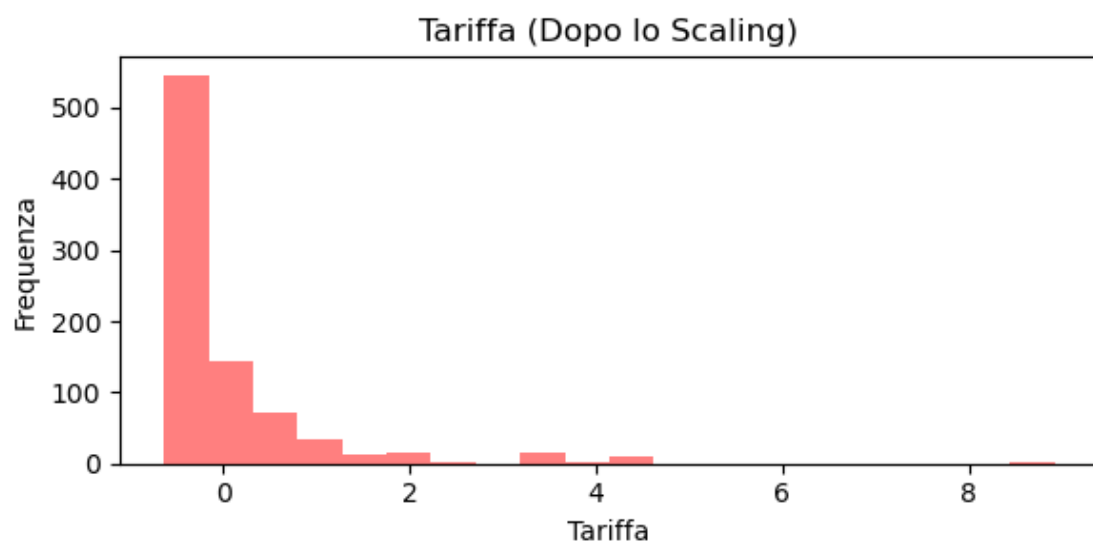
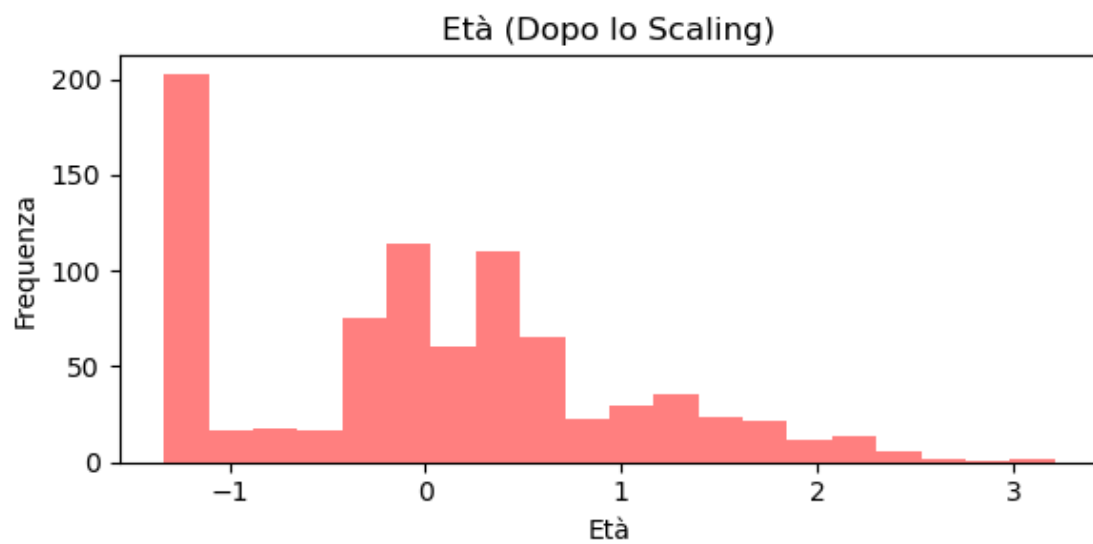
# Visualizzazione delle distribuzioni delle colonne selezionate dopo lo scaling
plt.figure(figsize=(6, 6))

# Itera su tutte le colonne selezionate e crea un subplot per ciascuna colonna
for i, col in enumerate(selected_columns):
    plt.subplot(2, len(selected_columns)//2, i+1) # Crea un subplot in una
    ↪griglia 2x2, selezionando l'indice 'i+1'
    plt.hist(scaled_df[col], bins=20, color='red', alpha=0.5) # Crea un
    ↪istogramma dei valori della colonna dopo lo scaling
    plt.title(col + ' (Dopo lo Scaling)') # Aggiungi un titolo al subplot
    plt.xlabel(col) # Aggiungi un'etichetta all'asse x
    plt.ylabel('Frequenza') # Aggiungi un'etichetta all'asse y

plt.tight_layout()
plt.show()

```





8 Splitting Dataset

```
[27]: from sklearn.model_selection import train_test_split

# Dividi il dataset in set di addestramento e set di test
X_train, X_test, y_train, y_test = train_test_split(df[['Età', 'Sesso']],
    →df['Sopravvissuti'], test_size=0.25, random_state=42)

# Calcola il numero di sopravvissuti per età e sesso nel set di addestramento
survived_train = X_train.join(y_train).groupby(['Età',
    →'Sesso'])['Sopravvissuti'].sum()

# Calcola il numero di sopravvissuti per età e sesso nel set di test
survived_test = X_test.join(y_test).groupby(['Età', 'Sesso'])['Sopravvissuti'].
    →sum()

print("Numero di sopravvissuti per età e sesso nel set di addestramento:")
print(survived_train)
print("\nNumero di sopravvissuti per età e sesso nel set di test:")
print(survived_test)
```

Numero di sopravvissuti per età e sesso nel set di addestramento:

Età	Sesso	
0.0000	female	22
	male	10
0.1667	female	1
0.4167	male	1
0.6667	male	1
	..	
67.0000	male	0
70.0000	male	0
74.0000	male	0
76.0000	female	1
80.0000	male	1

Name: Sopravvissuti, Length: 139, dtype: int64

Numero di sopravvissuti per età e sesso nel set di test:

Età	Sesso	
0.0000	female	10
	male	5
0.9167	male	1
1.0000	male	1
2.0000	male	1
	..	
60.0000	male	0
61.0000	male	0

```
62.0000  male      0
63.0000  male      0
64.0000  male      0
Name: Sopravvissuti, Length: 87, dtype: int64
```

```
[28]: import matplotlib.pyplot as plt

# Calcola il numero di sopravvissuti per età e sesso nel set di addestramento
survived_train = X_train.join(y_train).groupby(['Età',
→ 'Sesso'])['Sopravvissuti'].sum().reset_index()

# Calcola il numero di sopravvissuti per età e sesso nel set di test
survived_test = X_test.join(y_test).groupby(['Età', 'Sesso'])['Sopravvissuti'].
→ sum().reset_index()

# Crea un subplot con due grafici a barre
fig, axes = plt.subplots(1, 2, figsize=(15, 6))

# Grafico per il set di addestramento
axes[0].bar(survived_train.index, survived_train['Sopravvissuti'], color='blue',
→ label='Sopravvissuti')
axes[0].set_title('Numero di sopravvissuti nel set di addestramento') # Titolo
→ del grafico
axes[0].set_xlabel('Età e Sesso') # Etichetta asse x
axes[0].set_ylabel('Numero di sopravvissuti') # Etichetta asse y
axes[0].set_xticks(survived_train.index) # Imposta le posizioni delle etichette
→ sull'asse x
axes[0].set_xticklabels(survived_train['Età'].astype(str) + '-' +
→ survived_train['Sesso']) # Imposta le etichette sull'asse x
axes[0].legend() # Mostra la legenda nel grafico

# Grafico per il set di test
axes[1].bar(survived_test.index, survived_test['Sopravvissuti'], color='orange',
→ label='Sopravvissuti')
axes[1].set_title('Numero di sopravvissuti nel set di test') # Titolo del
→ grafico
axes[1].set_xlabel('Età e Sesso') # Etichetta asse x
axes[1].set_ylabel('Numero di sopravvissuti') # Etichetta asse y
axes[1].set_xticks(survived_test.index) # Imposta le posizioni delle etichette
→ sull'asse x
axes[1].set_xticklabels(survived_test['Età'].astype(str) + '-' +
→ survived_test['Sesso']) # Imposta le etichette sull'asse x
axes[1].legend() # Mostra la legenda nel grafico

plt.tight_layout() # Ottimizza la disposizione dei subplot
plt.show()
```

