Progetto Dallavecchia_Zanobini

June 1, 2022

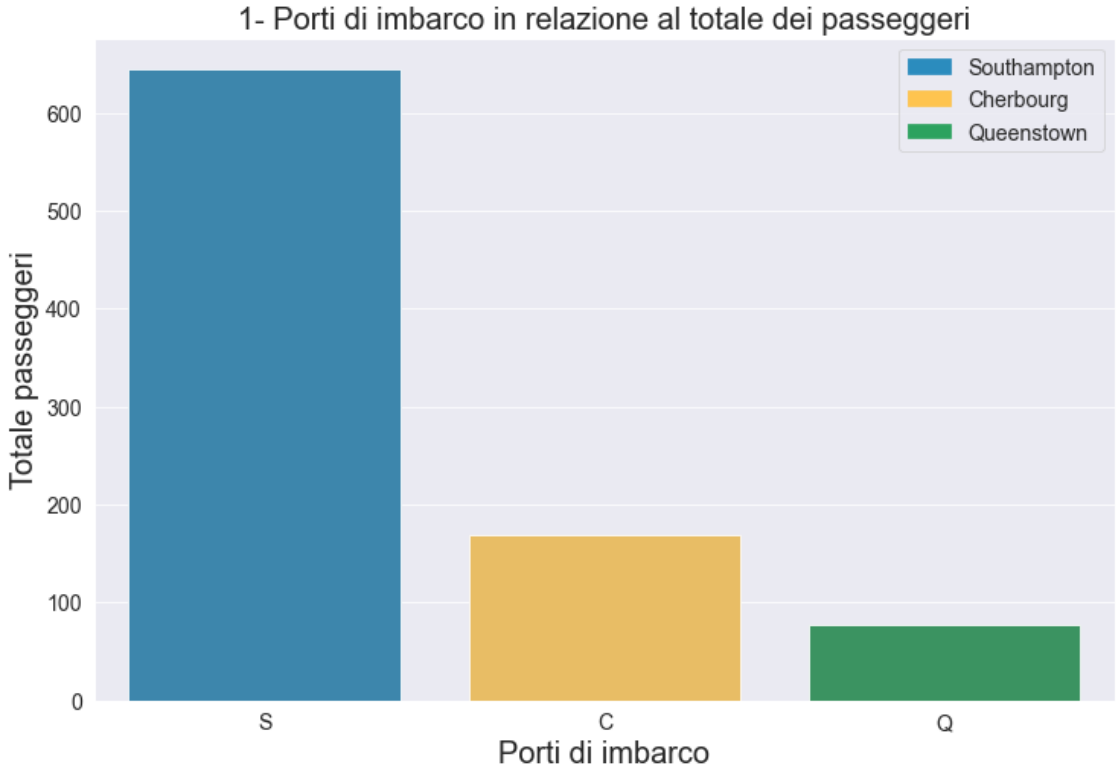
Progetto Dallavecchia Giulia e Zanobini Diego

Progetto Dallavecchia Giulia e Zanobini Diego

Analisi dei passeggeri

[2]:
```
#definizione delle librerie utilizzate
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from numpy import mean
import seaborn as sns
%matplotlib inline
df = pd.read_csv("C:
 ↪\\Users\\giuli\\OneDrive\\Desktop\\MAGISTRALE\\0_ANNUALI\\Programmazione e␣
 ↪analisi dei dati\\Modulo B\\Dataset\\train.csv", encoding='latin')

#Suddivisione per nome del porto di imbarco
S = df[df['Embarked'] == 'S']
C = df[df['Embarked'] == 'C']
Q = df[df['Embarked'] == 'Q']

#Conteggio dei passeggeri imbarcati per porto
IDS = S['PassengerId'].size
IDC = C['PassengerId'].size
IDQ = Q['PassengerId'].size

#crezione di un array contentente il totale dei passeggeri imbarcati per porto
p = [IDS,IDC,IDQ]
#Definizione dei nomi dei porti utilizzati nell'asse delle x
n_porti = ["S", "C", "Q"]


#Definizione dei colori personalizzati: #http://colorbrewer2.org/
porti = ["#2b8cbe", "#fec44f", "#2ca25f"]
#Definizione del layout del grafico
plt.rcParams["figure.figsize"] = [12,8]
sns.set_style("darkgrid")
```

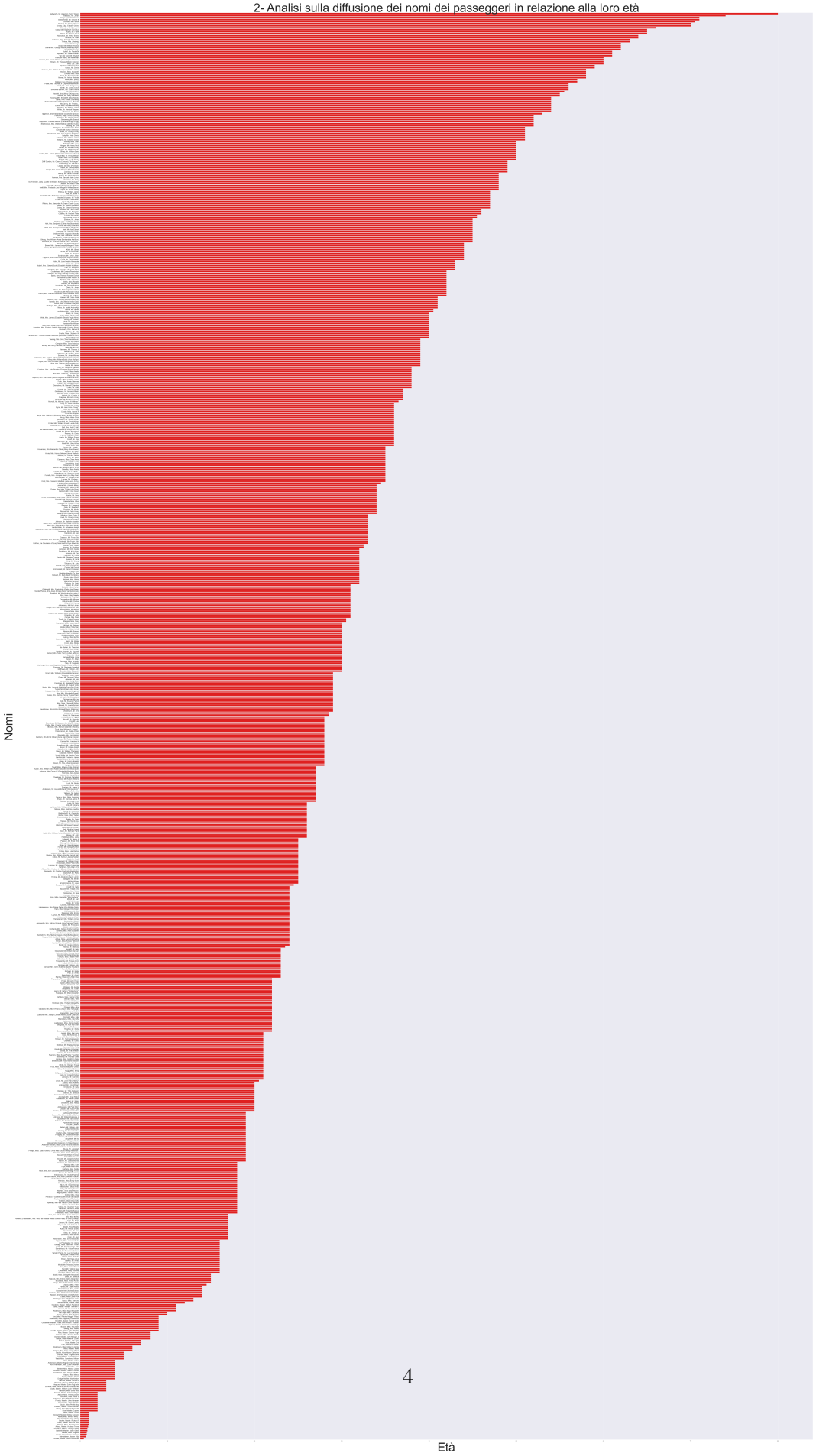
```
sns.set_context("paper", font_scale=1.6, rc={"font.size":20,"axes.titlesize":␣
↪20,"axes.labelsize":20})

#CREAZIONE DEL BAR CHART
sns.barplot(x=n_porti, y=p, palette=sns.color_palette(porti), data=df, ci=None)
#Legenda
colori_barre = {'Southampton':'#2b8cbe', 'Cherbourg':'#fec44f', 'Queenstown':
↪'#2ca25f'}
labels = list(colori_barre.keys())
handles = [plt.Rectangle((0,0),1,1, color=colori_barre[label]) for label in␣
↪labels]
plt.legend(handles, labels, loc="upper right")
#Definizione del titolo e dei nomi delle ascisse e delle ordinate
plt.title("1- Porti di imbarco in relazione al totale dei passeggeri")
plt.xlabel("Porti di imbarco")
plt.ylabel("Totale passeggeri")
plt.show()
```



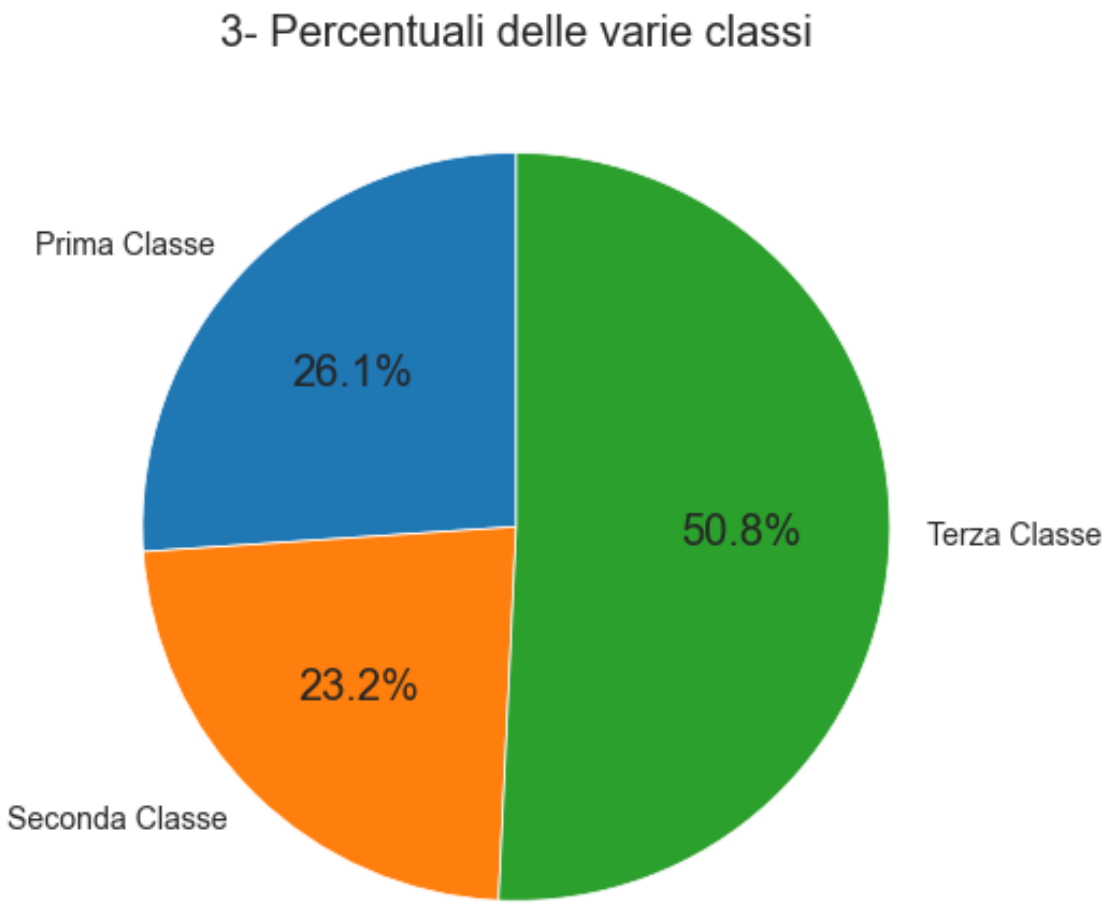
[200]:
```
#Ordinameto dei record per età dei passeggeri
df = df.sort_values('Age',ascending=False).reset_index(drop=True)
#Cancellazione record in cui il valore dell'età non è definito
df = df.dropna(subset=['Age'])
```

```
#Definizione del layout del grafico
plt.rcParams["figure.figsize"] = [100,200]
sns.set_style("darkgrid")
sns.set_context("paper", font_scale=1.5, rc={"font.size":90,"axes.titlesize":
 ↪90,"axes.labelsize":90})

#CREAZIONE DEL PARETOSORT
sns.barplot(x=eta, y="Name", data=df, color="red")
#Definizione del titolo e dei nomi delle ascisse e delle ordinate
plt.title("2- Analisi sulla diffusione dei nomi dei passeggeri in relazione␣
 ↪alla loro età")
plt.ylabel("Nomi")
plt.xlabel("Età")
plt.show()
```

2- Analisi sulla diffusione dei nomi dei passeggeri in relazione alla loro età

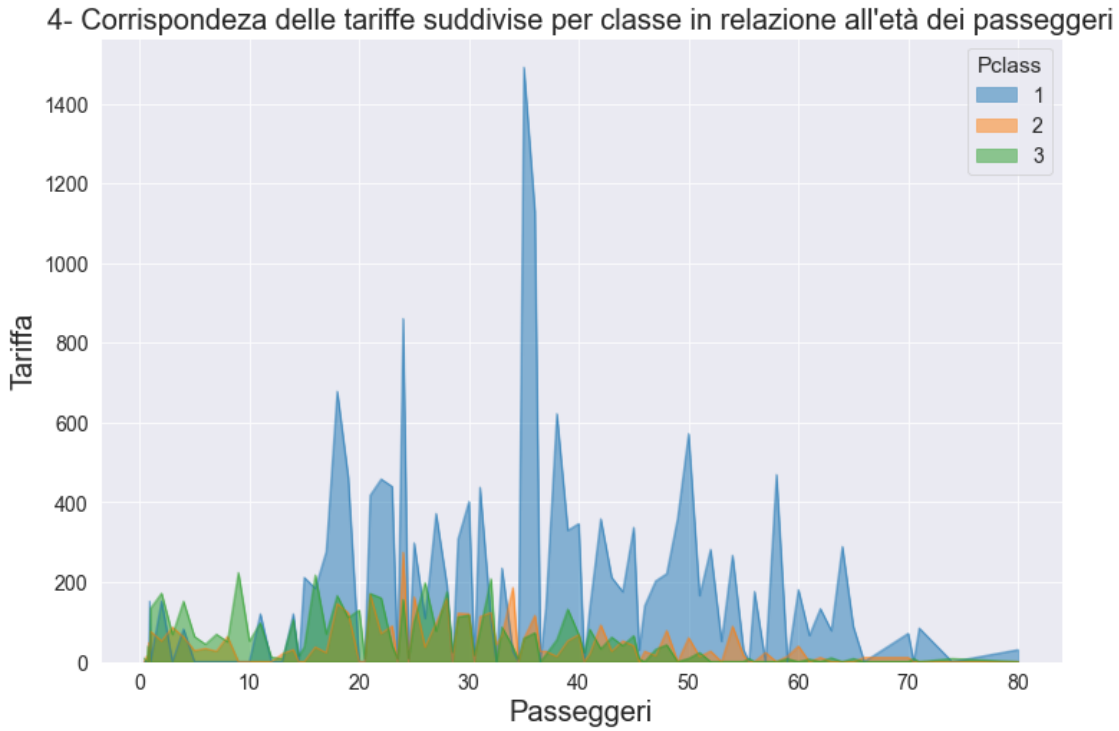


[189]:
```
#Definizione delle librerie utilizzate
from numpy import mean
from numpy import array

#Suddivisione dei passeggeri per classe
Prima_classe = df[df['Pclass'] == 1].size
Seconda_classe = df[df['Pclass'] == 2].size
Terza_classe = df[df['Pclass'] == 3].size
#Conteggio dei passeggeri totali a bordo
conta_passeggeri = df['PassengerId'].size

#Creazione degli array relativi ad ogni classe
p1 = [Prima_classe,conta_passeggeri]
p2 = [Seconda_classe,conta_passeggeri]
p3 = [Terza_classe,conta_passeggeri]

#calcolo della media dei passeggeri presenti per classe sul totale dei␣
 ↪passeggeri
media1 = mean(p1)
media2 = mean(p2)
media3 = mean(p3)

#Definizione del layout del grafico
plt.rcParams["figure.figsize"] = [12,8]
sns.set_context("paper", font_scale=1.6, rc={"font.size":20,"axes.titlesize":␣
 ↪20,"axes.labelsize":20})

#CREAZIONE DEL PIE CHART
labels =['Prima Classe', 'Seconda Classe', 'Terza Classe']
sizes = [media1, media2, media3]
plt.pie(sizes, labels = labels,  autopct = '%1.1f%%', startangle= 90)
#Definizione del titolo
plt.title('3- Percentuali delle varie classi')
plt.show()
```



[24]:
```
#Definizione delle librerie utilizzate
from numpy import mean
from numpy import array

#Suddivisione del dataset per classe
Prima_classe = df[df['Pclass'] == 1]
Seconda_classe = df[df['Pclass'] == 2]
Terza_classe = df[df['Pclass'] == 3]


#Suddivisione della tariffa del biglietto per classe
Tariffa1 = Prima_classe['Fare']
Tariffa2 = Seconda_classe['Fare']
Tariffa3 = Terza_classe['Fare']

#Calcolo della Tariffa media per classe
```

```
m1 = mean(Tariffa1)
m2 = mean(Tariffa2)
m3 = mean(Tariffa3)

#Definizione del layout del grafico
plt.rcParams["figure.figsize"] = [12,8]
sns.set_style("darkgrid")
sns.set_context("paper", font_scale=1.6, rc={"font.size":20,"axes.titlesize":␣
 ↪20,"axes.labelsize":20})

#Definizione dei colori
colors = ("#67000d", "#fee0d2", "#fcbba1")

#CREAZIONE DELLO STACKED CHART
Summary = pd.crosstab(df['Age'], df['Pclass'], values=df['Fare'], aggfunc=np.
 ↪sum) # qui usiamo la libreria numpy: np.sum
Summary.plot(kind="area", stacked=False)


#Definizione del titolo e dei nomi delle ascisse e delle ordinate
plt.title("4- Corrispondeza delle tariffe suddivise per classe in relazione␣
 ↪all'età dei passeggeri")
plt.ylabel("Tariffa")
plt.xlabel("Passeggeri")
plt.show()
```

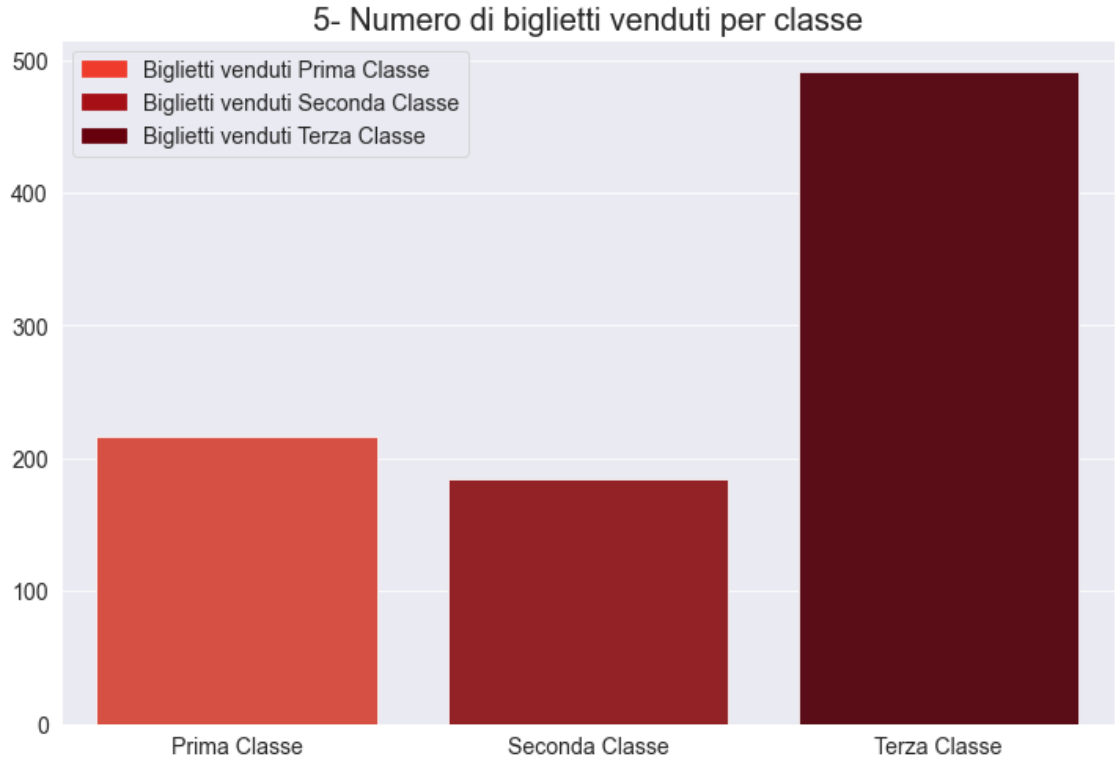


[27]:
```
#Definizione delle librerie utilizzate
from numpy import mean
from numpy import array

#Suddivisione del dataset per classe
Prima_classe = df[df['Pclass'] == 1]
Seconda_classe = df[df['Pclass'] == 2]
Terza_classe = df[df['Pclass'] == 3]

#Calcolo del numero totale dei biglietti per classe
Biglietto1= Prima_classe['Ticket'].size
Biglietto2= Seconda_classe['Ticket'].size
Biglietto3 = Terza_classe['Ticket'].size


#Creazione array contenente il numero totale dei biglietti per classe
Biglietto = [Biglietto1, Biglietto2, Biglietto3]
#Definizone array contenente i valori da inserire nell'asse delle y
Classe = ["Prima Classe", "Seconda Classe", "Terza Classe"]

#Definizione del layout del grafico
plt.rcParams["figure.figsize"] = [12,8]
sns.set_style("darkgrid")
sns.set_context("paper", font_scale=1.6, rc={"font.size":20,"axes.titlesize":␣
↪20,"axes.labelsize":20})
#Definizione colori
classi = ["#ef3b2c", "#a50f15", "#67000d"]

#CREAZIONE DEL BAR CHART
sns.barplot(x=Classe, y=Biglietto, palette=sns.color_palette(classi), data=df,␣
↪ci=None)
#Legenda
colori_barre = {'Biglietti venduti Prima Classe':'#ef3b2c', 'Biglietti venduti␣
↪Seconda Classe':'#a50f15', 'Biglietti venduti Terza Classe':'#67000d'}
labels = list(colori_barre.keys())
handles = [plt.Rectangle((0,0),1,1, color=colori_barre[label]) for label in␣
↪labels]
plt.legend(handles, labels, loc="upper left")

#Definizione del titolo
plt.title("5- Numero di biglietti venduti per classe")
plt.show()
```

216

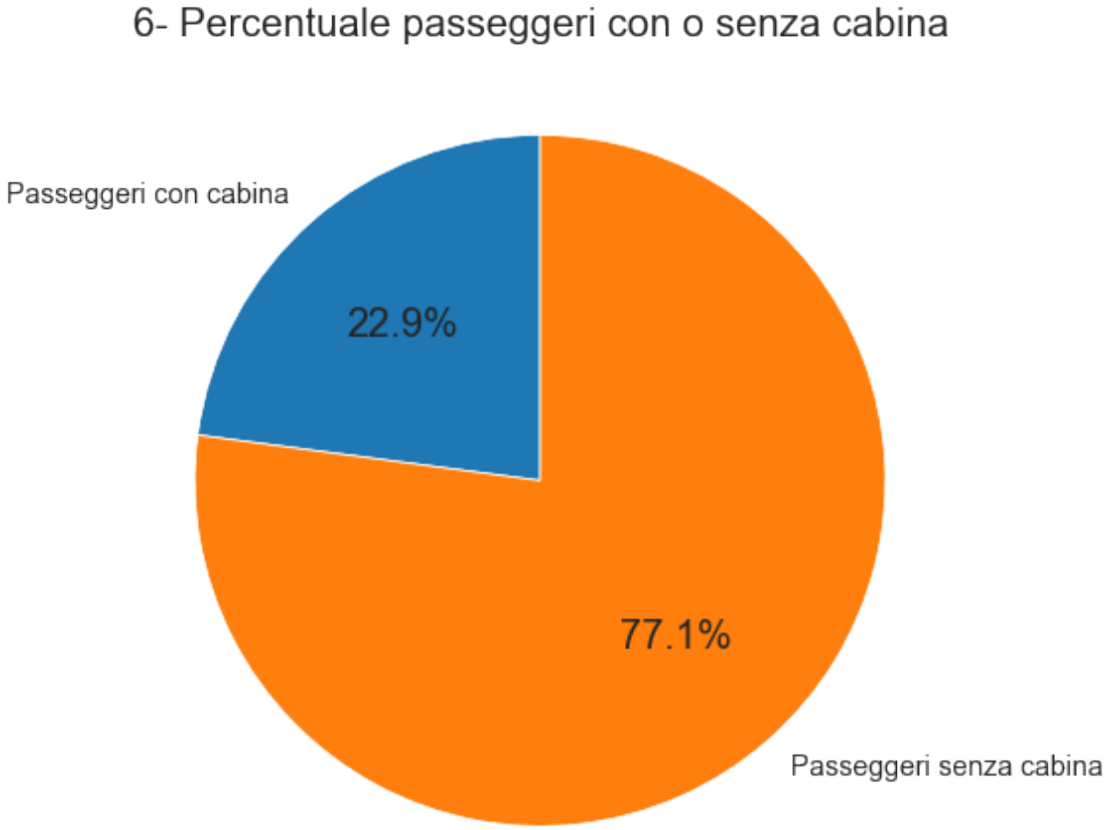
184

491



[186]:
```
#Definizione delle librerie utilizzate
from numpy import mean
from numpy import array
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from numpy import mean
import seaborn as sns
%matplotlib inline

#Definizione del layout del grafico
plt.rcParams["figure.figsize"] = [12,8]
sns.set_context("paper", font_scale=1.6, rc={"font.size":20,"axes.titlesize":␣
↪20,"axes.labelsize":20})

#creazione array selezionando gli elementi della tabella Cabin
Cabine= df['Cabin']
#Conteggio del numero totale delle cabine
conta_cabine = Cabine.size
```

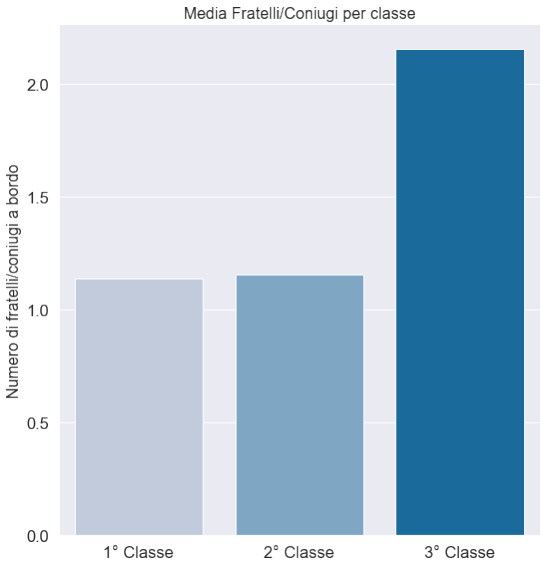
```
#controllo sul valore dei record, se nulli vengono immaginazzinati nella␣
 ↪variabile cabina_no
cabina_no = pd.isnull(Cabine)
#conteggio dei passeggeri che non hanno la cabina
conta_no = np.sum(cabina_no)
#conteggio passeggeri che hanno la cabina sottraendo dal totale dei passeggeri␣
 ↪quello senza cabina
cabina_si = conta_cabine - conta_no

#assegnamento dei nomi alle percentuali del grafico
labels =['Passeggeri con cabina', 'Passeggeri senza cabina']
#assegnamento dei valori percentuali
sizes = [cabina_si , conta_no]

#CREAZIONE DEL PIE CHART
plt.pie(sizes, labels = labels,  autopct = '%1.1f%%', startangle= 90)
#Definizione del titolo
plt.title('6- Percentuale passeggeri con o senza cabina')
plt.show()
```

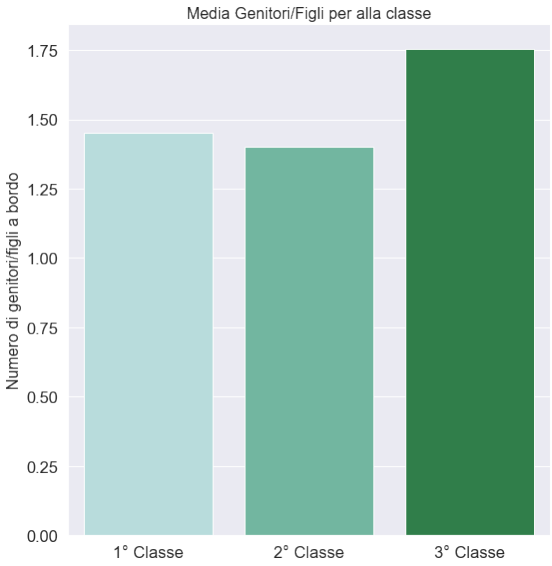


Analisi delle parentele

[153]:
```
#Suddivisione dei passeggeri per classe
Prima_classe = df[df['Pclass'] == 1]
Seconda_classe = df[df['Pclass'] == 2]
Terza_classe = df[df['Pclass'] == 3]

#PARAMETRI PER IL GRAFICO DI CONFRONTO FRATELLI/CONIUGI
#Creazione array per il confronto fratelli/coniugi suddivisi per classe con␣
 ↪controllo sul valore del record
Fratelli_coniugi1 = Prima_classe[Prima_classe['SibSp'] > 0]
Fratelli_coniugi2 = Seconda_classe[Seconda_classe['SibSp'] > 0]
Fratelli_coniugi3 = Terza_classe[Terza_classe['SibSp'] > 0]

#Creazione degli array relativi al valore del record relativo a fratelli/
 ↪coniugi definiti precedentemente
FC1 = Fratelli_coniugi1['SibSp']
FC2 = Fratelli_coniugi2['SibSp']
FC3 = Fratelli_coniugi3['SibSp']

#Calcolo della media di fratelli/coniugi per classe
mediaFC1 = mean(FC1)
mediaFC2 = mean(FC2)
mediaFC3 = mean(FC3)

#Creazione arry contentente i valori delle tre medie
Fratelli_coniugi = [mediaFC1, mediaFC2, mediaFC3]


# PARAMETRI PER IL GRAFICO DI CONFRONTO GENITORI/FIGLI
#Creazione array per il confronto genitori/figli suddivisi per classe con␣
 ↪controllo sul valore del record
Genitori_figli1 = Prima_classe[Prima_classe['Parch'] > 0]
Genitori_figli2 = Seconda_classe[Seconda_classe['Parch'] > 0]
Genitori_figli3 = Terza_classe[Terza_classe['Parch'] > 0]

#Creazione degli array relativi al valore del record relativo a genitori/figli␣
 ↪definiti precedentemente
GF1 = Genitori_figli1['Parch']
GF2 = Genitori_figli2['Parch']
GF3 = Genitori_figli3['Parch']

#Calcolo della media di genitori/figli per classe
mediaGF1 = mean(GF1)
mediaGF2 = mean(GF2)
```

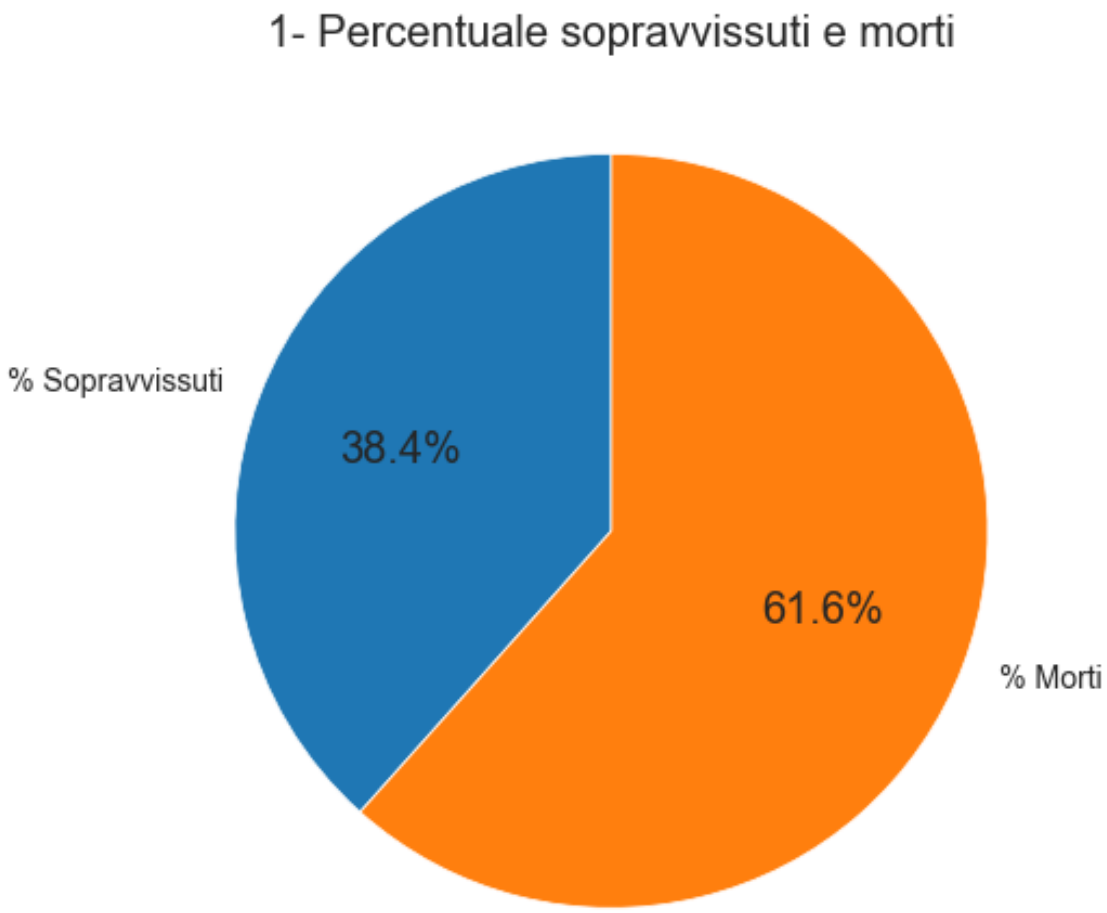
```
mediaGF3 = mean(GF3)

#Creazione arry contentente i valori delle tre medie
Genitori_figli = [mediaGF1, mediaGF2, mediaGF3]

#Definizione valori per l'asse delle x comune a tutti e due i grafici
classi=["1° Classe", "2° Classe", "3° Classe"]
#Definizione del layout dei grafici
plt.rcParams["figure.figsize"] = [12,8]



#Definizione colori per il grafico fratelli/coniugi
c = ["#bdc9e1", "#74a9cf", "#0570b0"]
#CREAZIONE DEL PRIMO BAR CHART
plt.subplot(1,2,1)
sns.barplot(x=classi, y=Fratelli_coniugi, palette=sns.color_palette(c),␣
 ↪data=df, ci=None)
#Definizione del titolo e del nome dell'asse delle y
plt.title("Media Fratelli/Coniugi per classe")
plt.ylabel("Numero di fratelli/coniugi a bordo")


#Definizione colori per il grafico genitori/figli
c1 = ["#b2e2e2", "#66c2a4", "#238b45"]
#CREAZIONE DEL SECONDO BAR CHART
plt.subplot(1,2,2)
sns.barplot(y=Genitori_figli, x=classi, palette=sns.color_palette(c1), data=df,␣
 ↪ci=None)
#Definizione del titolo e del nome dell'asse delle y
plt.title("Media Genitori/Figli per alla classe")
plt.ylabel("Numero di genitori/figli a bordo")

#Definizione del layout dei grafici a confronto
plt.tight_layout()
plt.subplots_adjust(right=1.30)
```





Analisi sui sopravvissuti

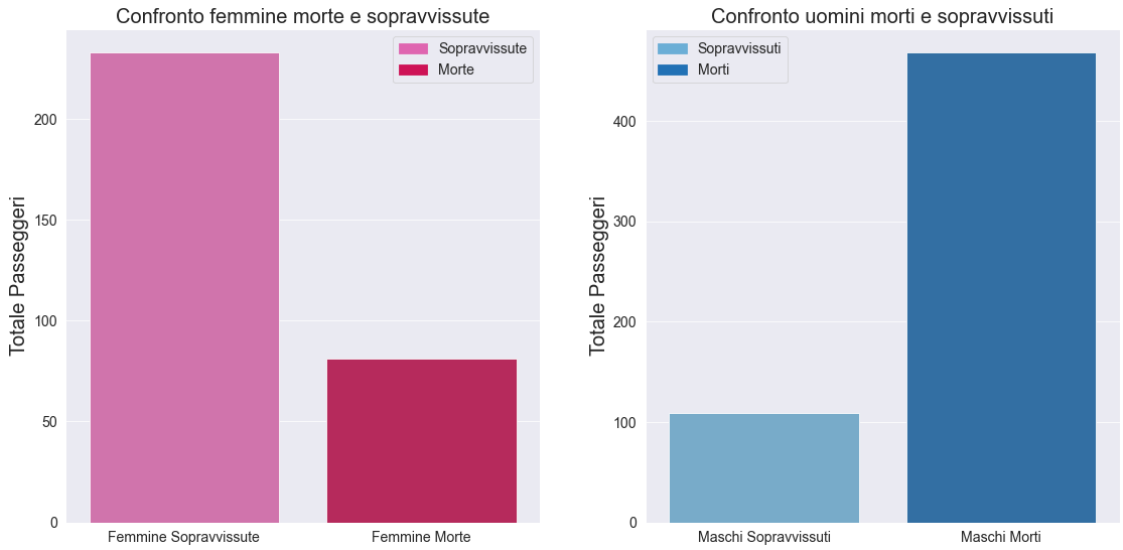
[196]:
```
#Definizione delle librerie utilizzate
from numpy import mean
from numpy import array

#Definizione degli array relativi ai morti e ai sopravvissuti con relativo␣
 ↪conteggio dei record
Sopravvissuti = df[df['Survived'] == 1].size
Morti = df[df['Survived'] == 0].size
#Conteggio del totale dei passeggeri
totale_passeggeri = df['PassengerId'].size

#Definizione del layout del grafico
plt.rcParams["figure.figsize"] = [12,8]
sns.set_context("paper", font_scale=1.6, rc={"font.size":20,"axes.titlesize":␣
 ↪20,"axes.labelsize":20})

#CREAZIONE DEL PIE CHART
labels =['% Sopravvissuti', '% Morti']
sizes = [Sopravvissuti, Morti]
plt.pie(sizes, labels = labels,  autopct = '%1.1f%%', startangle= 90)

#Definizione del titolo
plt.title('1- Percentuale sopravvissuti e morti')
plt.show()
```



[77]:
```
#Definizione del layout del grafico
plt.rcParams["figure.figsize"] = [12,8]
sns.set_context("paper", font_scale=1.6, rc={"font.size":20,"axes.titlesize":␣
 ↪20,"axes.labelsize":20})

#Creazione degli array suddividendo i passeggeri per sesso
M = df[df['Sex'] == 'male']
F = df[df['Sex'] == 'female']

#Creazione di array suddividendo i passeggeri per sesso e controllando se sono␣
 ↪sopravvissuti o meno
MS = M[M['Survived'] == 1]
FS = F[F['Survived'] == 1]
MM = M[M['Survived'] == 0]
FM = F[F['Survived'] == 0]
```

```
#Creazione degli array relativi ai passeggeri maschi e femmine morti/
 ↪sopravvissuti
Sopravvissuti_maschi = MS['PassengerId'].size
Sopravvissuti_Femmine = FS['PassengerId'].size
Morti_maschi = MM['PassengerId'].size
Morti_femmine = FM['PassengerId'].size

#Conteggio del totale dei maschi e delle femmine a bordo
Maschi = M['PassengerId'].size
Femmine = F['PassengerId'].size

#Creazione degli array contententi il totale dei maschi e delle femmine␣
 ↪sopravvissuti/morti
Somma_femmine = [Sopravvissuti_Femmine, Morti_femmine]
Somma_maschi = [Sopravvissuti_maschi, Morti_maschi]


#Definizione dei colori per il primo grafico
colori_femmine = ["#df65b0", "#ce1256"]
#Definizione dei valori da inserire nell'assa delle x
x = ['Femmine Sopravvissute', 'Femmine Morte']
#CREAZIONE DEL PRIMO BAR CHART
plt.subplot(1,2,1)
sns.barplot(x=x, y=Somma_femmine, palette=sns.color_palette(colori_femmine),␣
 ↪data=df, ci=None)
#Definizione del titolo e del nome dell'asse delle y
plt.title("Confronto femmine morte e sopravvissute")
plt.ylabel("Totale Passeggeri")
#legenda
colori_b = {'Sopravvissute':'#df65b0', 'Morte':'#ce1256' }
labels = list(colori_b.keys())
handles = [plt.Rectangle((0,0),1,1, color=colori_b[label]) for label in labels]
plt.legend(handles, labels, loc="upper right")

#Definizione dei colori del secondo grafico
colori_maschi = ["#6baed6", "#2171b5"]
#Definizione dei valori da inserire nell'assa delle x
x1 = ['Maschi Sopravvissuti', 'Maschi Morti']
#CREAZIONE DEL SECONDO BAR CHART
plt.subplot(1,2,2)
sns.barplot(x=x1, y=Somma_maschi, palette=sns.color_palette(colori_maschi),␣
 ↪data=df, ci=None)
#Definizione del titolo e del nome dell'asse delle y
plt.title("Confronto uomini morti e sopravvissuti")
plt.ylabel("Totale Passeggeri")
#legenda
```

```
colori_b = {'Sopravvissuti':'#6baed6', 'Morti':'#2171b5' }
labels = list(colori_b.keys())
handles = [plt.Rectangle((0,0),1,1, color=colori_b[label]) for label in labels]
plt.legend(handles, labels, loc="upper left")


#Definizione del layout dei grafici a confronto
plt.tight_layout()
plt.subplots_adjust(right=1.30)
```



[198]:
```
#creazione array che contiene solamente i sopravvissuti
Sopravvissuti = df[df['Survived'] == 1]

#Cancellazione record in cui il valore dell'età non è definito
Sopravvissuti.dropna(subset=['Age'])

#Creazione degli array in base all'età dei passeggeri
Bambini = Sopravvissuti[Sopravvissuti['Age'] < 31].size
Adulti =  Sopravvissuti[Sopravvissuti['Age'] > 31].size
Anziani = Sopravvissuti[Sopravvissuti['Age'] > 65].size
totale_sopravvissuti = Sopravvissuti.size

#Definizione del layout del grafico
plt.rcParams["figure.figsize"] = [12,8]
sns.set_context("paper", font_scale=1.6, rc={"font.size":20,"axes.titlesize":␣
↪20,"axes.labelsize":20})

#CREAZIONE DEL PIE CHART
```

```
labels =['% Sopravvissuti 0-30 anni', '% Sopravvissuti 31-65 anni', '%␣
 ↪Sopravvissuti over 65 anni']
sizes = [Bambini, Adulti, Anziani]
plt.pie(sizes, labels = labels,  autopct = '%1.1f%%', startangle= 90)

#Definizione del titolo
plt.title("3- Percentuale di sopravvissuti per fascia d'età")
plt.show()
```