Splitting Dataset

March 26, 2024

[24]:
```
#identificazione delle righe ocn dati mancanti
righe_con_dati_mancanti = df[df.isnull().any(axis=1)]
righe_con_dati_mancanti
```

[24]:

	Div	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	…	\
278	I1	14/03/10	Parma	Atalanta	1	0	H	0	0	D	…	
286	I1	21/03/10	Chievo	Catania	1	1	D	0	0	D	…	
299	I1	25/03/10	Napoli	Juventus	3	1	H	0	1	A	…	
302	I1	28/03/10	Chievo	Parma	0	0	D	0	0	D	…	
304	I1	28/03/10	Juventus	Atalanta	2	1	H	1	1	D	…	
313	I1	03/04/10	Catania	Palermo	2	0	H	2	0	H	…	
321	I1	10/04/10	Napoli	Parma	2	3	A	1	0	H	…	
322	I1	11/04/10	Bologna	Lazio	2	3	A	2	1	H	…	
331	I1	17/04/10	Chievo	Livorno	2	0	H	1	0	H	…	
339	I1	18/04/10	Udinese	Bologna	1	1	D	0	1	A	…	
342	I1	25/04/10	Bologna	Parma	2	1	H	1	1	D	…	
353	I1	02/05/10	Bari	Genoa	3	0	H	0	0	D	…	
354	I1	02/05/10	Cagliari	Udinese	2	2	D	1	2	A	…	
360	I1	09/05/10	Bologna	Catania	1	1	D	1	0	H	…	
374	I1	16/05/10	Cagliari	Bologna	1	1	D	0	1	A	…	
375	I1	16/05/10	Catania	Genoa	1	0	H	0	0	D	…	

	BbMx>2.5	BbAv>2.5	BbMx<2.5	BbAv<2.5	BbAH	BbAHh	BbMxAHH	BbAvAHH	\
278	2.37	2.17	1.67	1.63	7	0.00	1.47	1.41	
286	2.40	2.25	1.62	1.58	6	-0.25	2.19	2.15	
299	2.20	2.06	1.76	1.70	16	0.00	1.72	1.68	
302	1.89	1.84	2.05	2.02	6	0.00	1.55	1.52	
304	1.86	1.78	2.08	1.98	11	-1.00	2.18	2.11	
313	2.30	2.15	1.75	1.68	13	0.00	1.91	1.84	
321	2.20	2.03	1.79	1.71	14	-1.00	2.08	2.01	
322	2.40	2.38	1.53	1.50	12	0.00	2.12	2.03	
331	2.16	2.03	1.85	1.73	9	-1.25	2.08	2.03	
339	2.10	2.01	1.88	1.79	5	-0.50	2.23	2.20	
342	1.95	1.89	1.91	1.82	7	-0.50	1.56	1.53	
353	1.75	1.68	2.25	2.15	9	0.00	1.87	1.80	
354	1.75	1.69	2.20	2.15	10	0.00	1.80	1.75	
360	2.32	2.27	1.71	1.69	6	-0.25	2.39	2.34	
374	1.50	1.43	2.87	2.56	8	-0.50	1.63	1.61	

```
375      1.53      1.47      2.63      2.45     6  -0.75     1.69     1.67

     BbMxAHA  BbAvAHA
278     3.00     2.70
286     1.78     1.76
299     2.28     2.14
302     2.70     2.55
304     1.79     1.77
313     2.03     1.97
321     1.89     1.83
322     1.85     1.80
331     1.89     1.85
339     1.77     1.75
342     2.66     2.58
353     2.09     2.05
354     2.19     2.14
360     1.68     1.67
374     2.52     2.41
375     2.33     2.30

[16 rows x 70 columns]
```

[19]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#specifico il percorso del tuo file CSV
percorso_file_csv = "/Users/edosido/robotica/pokemons.csv"

#leggi il file csv in un dataframe
df = pd.read_csv(percorso_file_csv)

#mostra le prime righe del dataframe (opzionale)
print(df.head())
```

```
   id        name      rank    generation evolves_from  type1   type2  hp  \
0   1   bulbasaur  ordinary  generation-i      nothing  grass  poison  45
1   2     ivysaur  ordinary  generation-i    bulbasaur  grass  poison  60
2   3    venusaur  ordinary  generation-i      ivysaur  grass  poison  80
3   4  charmander  ordinary  generation-i      nothing   fire    None  39
4   5  charmeleon  ordinary  generation-i   charmander   fire    None  58

   atk  def  spatk  spdef  speed  total  height  weight  \
0   49   49     65     65     45    318       7      69
1   62   63     80     80     60    405      10     130
2   82   83    100    100     80    525      20    1000
3   52   43     60     50     65    309       6      85
4   64   58     80     65     80    405      11     190
```

```
              abilities                                                desc
0  overgrow chlorophyll   A strange seed was planted on its back at birt…
1  overgrow chlorophyll   When the bulb on its back grows large, it appe…
2  overgrow chlorophyll   The plant blooms when it is absorbing solar en…
3     blaze solar-power   Obviously prefers hot places. When it rains, s…
4     blaze solar-power   When it swings its burning tail, it elevates t…
```

[21]:
```
#importa xlsx con fogli
import pandas as pd

#specifica il percorso del tuo file excel
percorso_file_excel = "/Users/edosido/robotica/serieA.xlsx"

#leggi il file excel in un dataframe
df = pd.read_excel(percorso_file_excel, sheet_name='09-10')

#ora puoi lavorare col dataframe df, che contiene i dati dal tuo file excel
df
```

[21]:

	position	team	Pt	Played	Won	Net	lose	Goals made	Goals suffered	Difference goals
0	1	Inter Inter	82	38	24	10	4	75	34	41
1	2	Roma Roma	80	38	24	8	6	68	41	27
2	3	Milan Milan	70	38	20	10	8	60	39	21
3	4	Sampdoria Sampdoria	67	38	19	10	9	49	41	8
4	5	Palermo Palermo	65	38	18	11	9	59		
5	6	Napoli Napoli	59	38	15	14	9	50		
6	7	Juventus Juventus	55	38	16	7	15	55		
7	8	Parma Parma	52	38	14	10	14	46		
8	9	Genoa Genoa	51	38	14	9	15	57		
9	10	Bari Bari	50	38	13	11	14	49		
10	11	Fiorentina Fiorentina	47	38	13	8	17	48		
11	12	Lazio Lazio	46	38	11	13	14	39		
12	13	Catania Catania	45	38	10	15	13	44		
13	14	Chievo Chievo	44	38	12	8	18	37		
14	15	Udinese Udinese	44	38	11	11	16	54		
15	16	Cagliari Cagliari	44	38	11	11	16	56		
16	17	Bologna Bologna	42	38	10	12	16	42		
17	18	Atalanta Atalanta	35	38	9	8	21	37		
18	19	Siena Siena	31	38	7	10	21	40		
19	20	Livorno Livorno	29	38	7	8	23	27		

```
4               47               12
5               43                7
6               56               -1
7               51               -5
8               61               -4
9               49                0
10              47                1
11              43               -4
12              45               -1
13              42               -5
14              59               -5
15              58               -2
16              55              -13
17              53              -16
18              67              -27
19              61              -34
```

[22]:
```
import os
import pandas as pd

#specifica il percorso della cartella contenente i file CSV
percorso_cartella = "/Users/edosido/robotica/serieAnuovo"

#lista dper memorizzare tutti i dataframe letti dati i file CSV
lista_dataframes = []

#scansiona la cartella e legg i file csv
for nome_file in os.listdir(percorso_cartella):
    if nome_file.endswith(".csv"):
        percorso_file_csv = os.path.join(percorso_cartella, nome_file)
        df = pd.read_csv(percorso_file_csv)
    lista_dataframes.append(df)

#ora lista dataframes contiene tutti i dataframe letti.dai file csv nella␣
 ↪cartella
df
```

[22]:
```
    Div      Date   HomeTeam    AwayTeam  FTHG  FTAG FTR  HTHG  HTAG HTR  …  \
0    I1  22/08/09    Bologna  Fiorentina     1     1   D     1     0   H  …
1    I1  22/08/09      Siena       Milan     1     2   A     1     1   D  …
2    I1  23/08/09    Catania   Sampdoria     1     2   A     1     1   D  …
3    I1  23/08/09      Genoa        Roma     3     2   H     0     0   D  …
4    I1  23/08/09      Inter        Bari     1     1   D     0     0   D  …
..   ..       …         …          …   …   …  ..   …   …  ..  …
375  I1  16/05/10    Catania       Genoa     1     0   H     0     0   D  …
376  I1  16/05/10     Chievo        Roma     0     2   A     0     2   A  …
377  I1  16/05/10      Parma     Livorno     4     1   H     1     0   H  …
```

```
378  I1  16/05/10  Sampdoria      Napoli     1     0   H     0     0   D  …
379  I1  16/05/10      Siena       Inter     0     1   A     0     0   D  …

     BbMx>2.5  BbAv>2.5  BbMx<2.5  BbAv<2.5  BbAH  BbAHh  BbMxAHH  BbAvAHH  \
0        2.20      2.03      1.78      1.72    19   0.00     2.57     2.38
1        2.18      2.04      1.80      1.73    19   0.50     1.91     1.85
2        2.31      2.12      1.70      1.66    17   0.00     1.83     1.76
3        2.01      1.94      1.94      1.80    17   0.00     1.75     1.67
4        1.85      1.75      2.15      2.00    17  -1.50     1.84     1.79
..        …         …         …         …     …    …        …        …
375      1.53      1.47      2.63      2.45     6  -0.75     1.69     1.67
376      1.36      1.31      3.40      3.17    11   1.25     1.86     1.84
377      1.55      1.47      3.00      2.56    14  -1.25     2.03     1.98
378      1.57      1.50      2.62      2.42    13  -1.25     1.88     1.85
379      1.48      1.43      3.00      2.66    12   1.75     2.11     2.06

     BbMxAHA  BbAvAHA
0       1.60     1.54
1       2.08     2.02
2       2.12     2.04
3       2.30     2.13
4       2.15     2.05
..       …        …
375     2.33     2.30
376     2.10     2.05
377     1.94     1.89
378     2.08     2.03
379     1.88     1.82

[380 rows x 70 columns]
```

[23]:
```
lista_dataframes[0]
```

[23]:
```
     Div      Date  HomeTeam   AwayTeam  FTHG  FTAG FTR  HTHG  HTAG HTR
0     I1  31/08/97  Atalanta    Bologna     4     2   H     1     0   H
1     I1  31/08/97      Bari      Parma     0     2   A     0     1   A
2     I1  31/08/97    Empoli       Roma     1     3   A     1     1   D
3     I1  31/08/97     Inter    Brescia     2     1   H     0     0   D
4     I1  31/08/97  Juventus      Lecce     2     0   H     0     0   D
..    ..       …         …         …   …   …  ..   …   …  ..
301   I1  16/05/98     Lecce   Piacenza     1     3   A     0     1   A
302   I1  16/05/98    Napoli       Bari     2     2   D     1     2   A
303   I1  16/05/98     Parma    Brescia     1     3   A     1     2   A
304   I1  16/05/98      Roma  Sampdoria     2     0   H     1     0   H
305   I1  16/05/98   Vicenza    Udinese     1     3   A     1     3   A

[306 rows x 10 columns]
```

1

2 Splitting Values

2.1 Cos'è lo Splitting Value?

2.1.1 Splitting Values, come dice dall'inglese, significa Dividere i Valori, di cui una parte verranno utilizzati come test e gli altri invece come allenamento. In questa lezion scopriamo come gestirli questi Valori

Librerie usate :

2.2 #### - Pandas (pd) > [importare, manipolare e analizzare dati, consente di eseguire operazioni come la selezione, il filtraggio, il raggruppamento e il calcolo di statistiche basati su Dataframe, cioè "tabelle".];

2.3 #### - NumPy (np) > [consente di gestire grandi quantità di dati ed offre anche un'ampia gamma di funzioni matematiche e di algebra lineare per manipolare e analizzare questi dati in modo efficiente.];

2.4 #### - Matplotlib.pyplot (plt) > [creare grafici e visualizzare dei dati in modo semplice. E' possibile generare diversi tipi di grafici, come a linee, istogrammi e scatter plot. Si può personalizzare l'aspetto dei grafici, aggiungere titoli, etichette degli assi e colori.];

2.5 #### - from sklearn.model_selection import train_test_split > [sklearn è una libreria usata per il machine learning ed analisi dati e train_test_split è una libreria che permette di, come dice il nome, dividere i dati ed addestrare questo machine learning con dei test_split];

- import random > [consente di generare numeri casuali, selezionare elementi casuali ed altre funzioni che comprendono la casualità di dati];

[ ]:
```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

import random
```

Iniziamo

[5]:
```
import numpy as np
from sklearn.model_selection import train_test_split

#creare dati casuali per altezze (variabile indipendente) e pesi (variabile␣
 ↪dipendente)
```

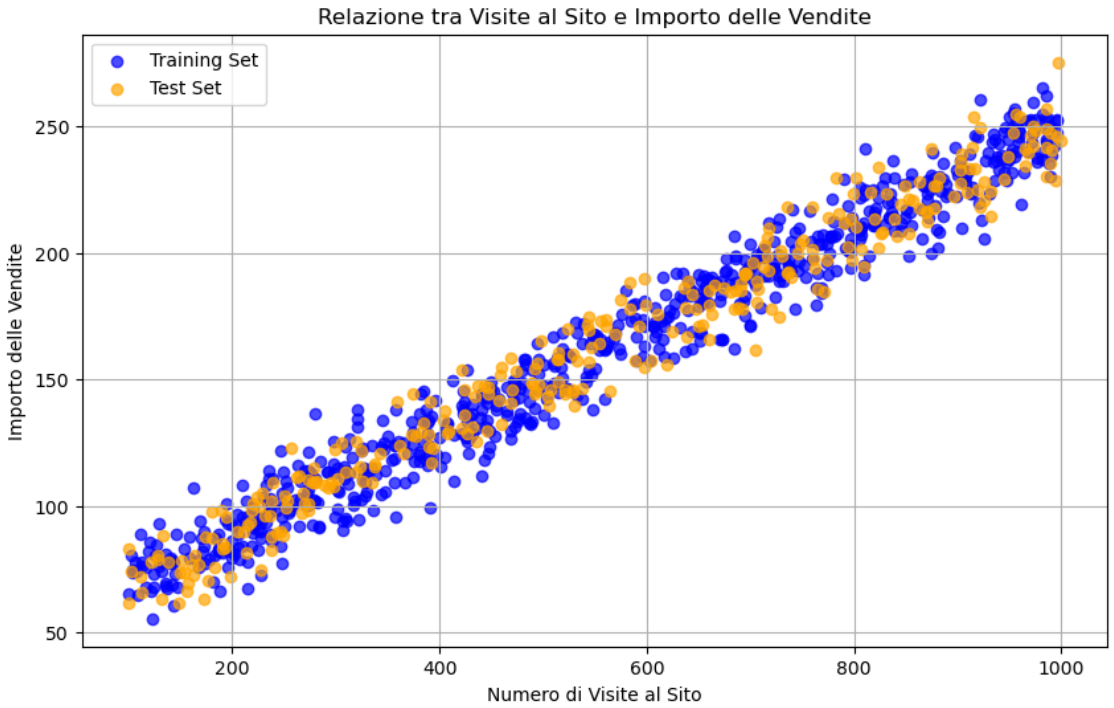
```
np.random.seed(0)
altezze = np.random.normal(160, 10, 100)
pesi = 0.5 * altezze + np.random.normal(0, 5, 100)

#suddividere il dataset in training set (70%) e test set (30%)
X_train, X_test, y_train, y_test = train_test_split(altezze, pesi, test_size=0.
 ↪3, random_state=42)
# test_size = 0.3, 30% dei dati vengono usati come test, il restante 70%␣
 ↪addestramento
# random_state = 42, come np.random.seed()

#stampare le dimensioni dei traning set a test set
# .shape= quanti numeri ci sono in una fila e quanti in una colonna
print("dimensioni del training set (altezze e pesi):", X_train.shape, y_train.
 ↪shape)
print("dimensioni del test set (altezze e pesi:)", X_test.shape, y_test.shape)
```

```
dimensioni del training set (altezze e pesi): (70,) (70,)
dimensioni del test set (altezze e pesi:) (30,) (30,)
```

2.5.1 Questo codice genera dati casuali per rappresentare altezze e pesi di persone, li divide in un gruppo di addestramento e un gruppo di test. Il gruppo di addestramento viene utilizzato per insegnare a un modello di machine learning, mentre il gruppo di test serve per verificare quanto bene il modello è stato addestrato. Alla fine, stampa le dimensioni di entrambi i gruppi di dati.

[6]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

# Creazione di dati casuali per visite al sito web e importo delle vendite
np.random.seed(0)
visite_al_sito = np.random.randint(100, 1000, 1000)
importo_vendite = 50 + 0.2 * visite_al_sito + np.random.normal(0, 10, 1000)

# Suddivisione del dataset in training set (70%) e test set (30%)
X_train, X_test, y_train, y_test = train_test_split(visite_al_sito,␣
 ↪importo_vendite, test_size=0.3, random_state=42)

# Creazione di un grafico a dispersione
plt.figure(figsize=(10, 6))
plt.scatter(X_train, y_train, label='Training Set', color='blue', alpha=0.7)
plt.scatter(X_test, y_test, label='Test Set', color='orange', alpha=0.7)
plt.xlabel('Numero di Visite al Sito')
plt.ylabel('Importo delle Vendite')
plt.title('Relazione tra Visite al Sito e Importo delle Vendite')
```

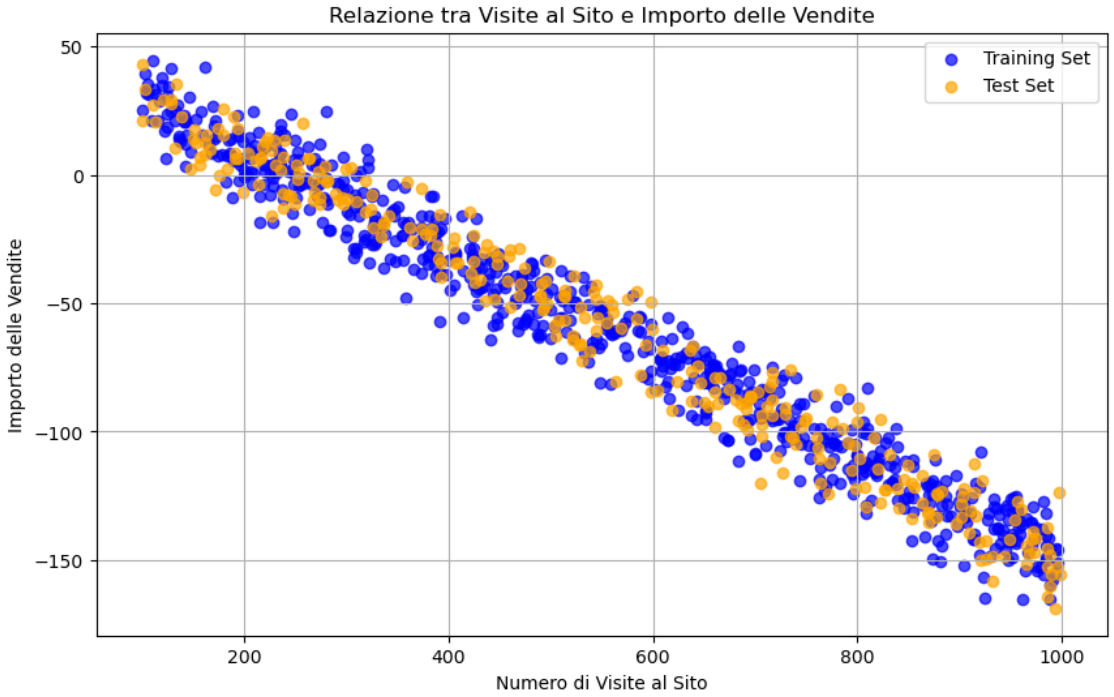
```
plt.legend()
plt.grid(True)
plt.show()

# Stampare le dimensioni dei training set e test set
# .shape= quanti numeri ci sono in una fila e quanti in una colonna
print("Dimensioni del Training Set (visite al sito e importo delle vendite):",␣
 ↪X_train.shape, y_train.shape)
print("Dimensioni del Test Set (visite al sito e importo delle vendite):",␣
 ↪X_test.shape, y_test.shape)
```




```
Dimensioni del Training Set (visite al sito e importo delle vendite): (700,)
(700,)
Dimensioni del Test Set (visite al sito e importo delle vendite): (300,) (300,)
```

2.5.2 Questo codice genera dati casuali per rappresentare le visite al sito web e gli importi delle vendite. Successivamente, suddivide questi dati in un set di addestramento e un set di test per valutare la relazione tra il numero di visite al sito e gli importi delle vendite. Infine, crea un grafico a dispersione per visualizzare questa relazione e stampa le dimensioni dei set di addestramento e di test.

[8]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

# Creazione di dati casuali per visite al sito web e importo delle vendite
np.random.seed(0)
visite_al_sito = np.random.randint(100, 1000, 1000)
importo_vendite = 50 - 0.2 * visite_al_sito + np.random.normal(0, 10, 1000)

# Suddivisione del dataset in training set (70%) e test set (30%)
X_train, X_test, y_train, y_test = train_test_split(visite_al_sito,␣
 ↪importo_vendite, test_size=0.3, random_state=42)

# Creazione di un grafico a dispersione
plt.figure(figsize=(10, 6))
plt.scatter(X_train, y_train, label='Training Set', color='blue', alpha=0.7)
plt.scatter(X_test, y_test, label='Test Set', color='orange', alpha=0.7)
plt.xlabel('Numero di Visite al Sito')
plt.ylabel('Importo delle Vendite')
plt.title('Relazione tra Visite al Sito e Importo delle Vendite')
plt.legend()
plt.grid(True)
plt.show()

# Stampare le dimensioni dei training set e test set
print("Dimensioni del Training Set (visite al sito e importo delle vendite):",␣
 ↪X_train.shape, y_train.shape)
print("Dimensioni del Test Set (visite al sito e importo delle vendite):",␣
 ↪X_test.shape, y_test.shape)
```




```
Dimensioni del Training Set (visite al sito e importo delle vendite): (700,)
(700,)
Dimensioni del Test Set (visite al sito e importo delle vendite): (300,) (300,)
```

2.5.3 Questo codice genera dati casuali per le visite al sito web e gli importi delle vendite, quindi li suddivide in un set di addestramento e un set di test. Successivamente, crea un grafico a dispersione per visualizzare la relazione tra le visite al sito e gli importi delle vendite nei due set di dati. Infine, stampa le dimensioni dei set di addestramento e di test.

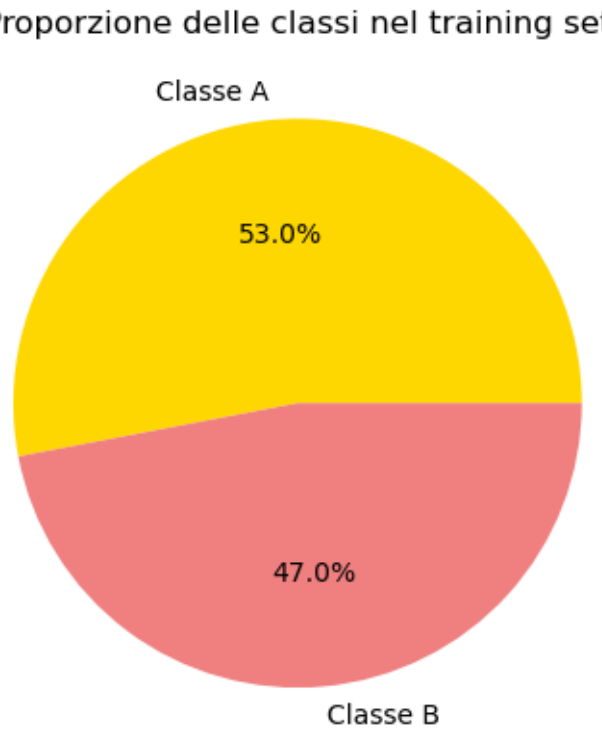
[9]:
```python
from sklearn.model_selection import train_test_split
import numpy as np

#supponiamo di avere un dataset con feature x e target y
X = np.random.rand(100, 2) #dati del dataset (100 campioni, 2 feature)
y = np.random.choice(['A', 'B'], size=100) #etichette di claasse casuali

#calcola le proporzioni delle lcassi dnel dataset origiale
proporzione_classe_A = sum(y == 'A') / len(y)
proporzione_classe_B = 1 - proporzione_classe_A

#eseguire una split stratificato con una proprozionale specificata
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

#calcola le proporzioni delle classi nel training set e nel test set
proporzione_classe_A_train = sum(y_train == 'A') / len(y_train)
proporzione_classe_B_train = 1 - proporzione_classe_A_train

proporzione_classe_A_test = sum(y_test == 'A') / len(y_test)
proporzione_classe_B_test = 1 - proporzione_classe_A_test



#stampa delle proporzioni
print("proporzione classe A nel dataset completo:", proporzione_classe_A)
print("proporzione classe B nel dataset completo:", proporzione_classe_B)
print("proporzione classe A nel training set:", proporzione_classe_A_train)
print("proporzione classe B nel training set:", proporzione_classe_B_train)
print("proporzione classe A nel test set:", proporzione_classe_A_test)
print("proporzione classe B nel test set:", proporzione_classe_B_test)
```

```
proporzione classe A nel dataset completo: 0.53
proporzione classe B nel dataset completo: 0.47
proporzione classe A nel training set: 0.44285714285714284
proporzione classe B nel training set: 0.5571428571428572
proporzione classe A nel test set: 0.7333333333333333
proporzione classe B nel test set: 0.2666666666666667
```

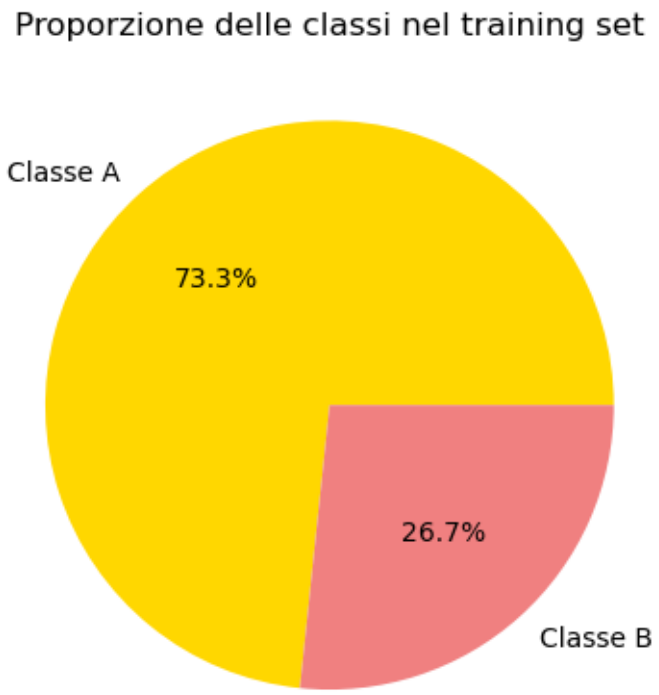
2.5.4 Questo codice genera un dataset casuale con 100 campioni, ognuno con due feature e etichette di classe casuali ('A' o 'B'). Successivamente, suddivide questo dataset in un set di addestramento e un set di test mantenendo la proporzione delle classi originali. Infine, calcola e stampa le proporzioni delle classi nel dataset completo, nel set di addestramento e nel set di test.

[10]:
```
#etichete delle cassi
labels = ['Classe A', 'Classe B']

#colori delle fette del grafico
colors = ['gold', 'lightcoral']

#crea un grafico a torta con etichette
plt.pie([proporzione_classe_A, proporzione_classe_B], labels=labels,␣
 ↪colors=colors, autopct='%1.1f%%')
plt.title('Proporzione delle classi nel training set')
plt.show()
```



2.5.5 Questo codice genera un grafico a torta che visualizza la proporzione delle classi nel set di addestramento. Le etichette delle fette del grafico corrispondono alle classi ('Classe A' e 'Classe B') e i colori delle fette sono specificati come oro e corallo chiaro. Ogni fetta del grafico rappresenta la proporzione di una classe rispetto al totale delle classi nel set di addestramento. Infine, il grafico viene mostrato.

[11]:
```
#etichete delle cassi
labels = ['Classe A', 'Classe B']

#colori delle fette del grafico
colors = ['gold', 'lightcoral']

#crea un grafico a torta con etichette
plt.pie([proporzione_classe_A_test, proporzione_classe_B_test], labels=labels,␣
↪colors=colors, autopct='%1.1f%%')
plt.title('Proporzione delle classi nel training set')
plt.show()
```



2.5.6 Questo codice genera un grafico a torta che visualizza la proporzione delle classi nel set di test. Le etichette delle fette del grafico corrispondono alle classi ('Classe A' e 'Classe B') e i colori delle fette sono specificati come oro e corallo chiaro. Ogni fetta del grafico rappresenta la proporzione di una classe rispetto al totale delle classi nel set di test. Infine, il grafico viene mostrato.

[12]:
```
import random
import numpy as np

dataset=[]
# Creazione di un dataset di 1000 elementi (ad esempio, dati casuali)
for i in range(1000):
    dataset.append(random.randint(1, 100))

# Estrazione di un campione casuale semplice di 50 elementi dal dataset
campione_casuale = random.sample(dataset, 300)

# Calcolo della media e della deviazione standard del campione
media_campione = np.mean(campione_casuale)
deviazione_standard_campione = np.std(campione_casuale)
```

```
# Calcolo della media e della deviazione standard del dataset completo
media_dataset = np.mean(dataset)
deviazione_standard_dataset = np.std(dataset) # .std = deviazione

print(f"Media del campione casuale: {media_campione: .2f}")
print(f"Deviazione standard del campione casuale: {deviazione_standard_campione:
 ↪ .2f}")
print(f"Media del dataset completo: {media_dataset: .2f}")
print(f"Deviazione standard del dataset completo: {deviazione_standard_dataset:␣
 ↪.2f}")
```

```
Media del campione casuale:  50.59
Deviazione standard del campione casuale:  27.98
Media del dataset completo:  50.89
Deviazione standard del dataset completo:  28.71
```

2.5.7 Questo codice genera un dataset casuale di 1000 elementi, estrae casualmente un campione di 300 elementi da questo dataset e calcola la media e la deviazione standard sia per il campione che per l'intero dataset. Infine, stampa i valori della media e della deviazione standard per entrambi i casi.

[13]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

# Impostare il seed per la riproducibilità
np.random.seed(42)
# Numero totale di elementi nel DataFrame
num_elementi = 1000
# Percentuale di "A"
percentuale_A = 0.7
# Generare la colonna con distribuzione desiderata
colonna = np.random.choice(['A', 'B'], size=num_elementi, p=[percentuale_A, 1 -␣
 ↪percentuale_A])

# Creare il DataFrame
df = pd.DataFrame({'ColonnaAB': colonna})
df
```

[13]:

	ColonnaAB
0	A
1	B
2	B
3	A

```
4           A
..        …
995         A
996         B
997         A
998         B
999         A

[1000 rows x 1 columns]
```

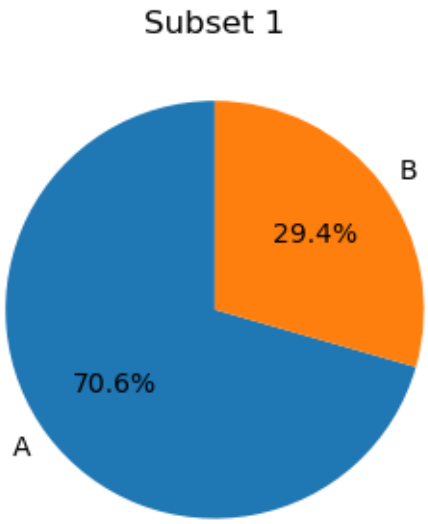
2.5.8 Questo codice utilizza la libreria Pandas per creare un DataFrame con una colonna chiamata 'ColonnaAB'. La colonna contiene valori casuali 'A' e 'B' con una percentuale specificata. Il DataFrame ha un totale di 1000 elementi e viene generato utilizzando un seed per la riproducibilità.

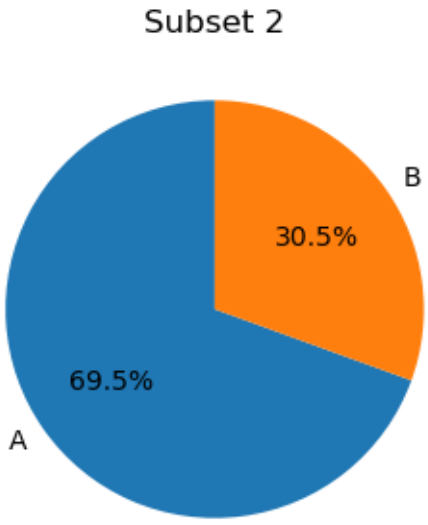
[25]:
```
#Creare tre subset di dimensioni simili
subset1 = df.sample(frac=1/3)
df = df.drop(subset1.index)

subset2 = df.sample(frac=1/2)
df = df.drop(subset2.index)

subset3 = df #l'ultimo subset com il rimanente
```

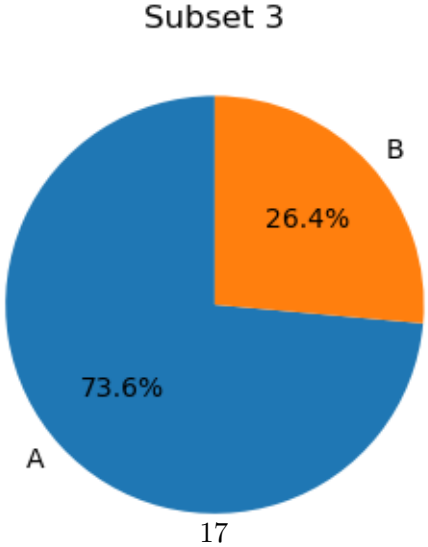
2.5.9 Questo codice suddivide un DataFrame in tre subset di dimensioni simili. Il primo subset (subset1) è composto da un terzo delle righe del DataFrame originale. Il secondo subset (subset2) è composto dalla metà delle righe rimanenti dopo l'estrazione di subset1. Il terzo subset (subset3) comprende le righe rimanenti non utilizzate per creare i primi due subset.

[15]:
```
#Calcolare le percentuali di "A" e "B" per ogni subset
## normalize=True, da % e non valore assoluto
percentuali_subset1 = subset1['ColonnaAB'].value_counts(normalize=True)
percentuali_subset2 = subset2['ColonnaAB'].value_counts(normalize=True)
percentuali_subset3 = subset3['ColonnaAB'].value_counts(normalize=True)

# Creare i grafici a torta
fig, axs = plt.subplots(3, 1, figsize=(6, 12))

# Subset 1
axs[0].pie(percentuali_subset1, labels=percentuali_subset1.index, autopct='%1.
 ↪1f%%', startangle=90)
axs[0].set_title('Subset 1')
```

```
# Subset 2
axs[1].pie(percentuali_subset2, labels=percentuali_subset2.index, autopct='%1.
  ↪1f%%', startangle=90)
axs[1].set_title('Subset 2')

# Subset 3
axs[2].pie(percentuali_subset3, labels=percentuali_subset3.index, autopct='%1.
  ↪1f%%', startangle=90)
axs[2].set_title('Subset 3')

# Mostrare il grafico
plt.show()
```


Subset 1
B
29.4%
70.6%
A

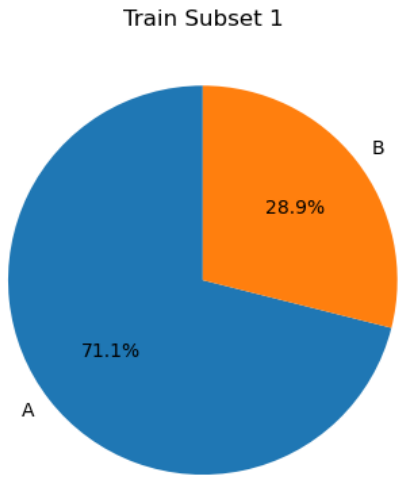

Subset 2
B
30.5%
69.5%
A

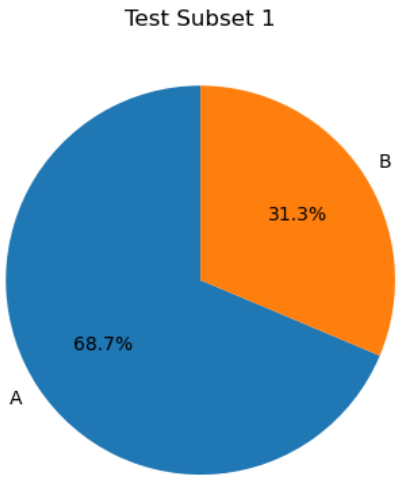

Subset 3
B
26.4%
73.6%
A

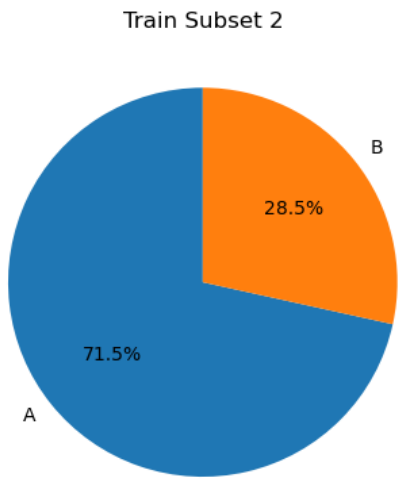
2.5.10 Questo codice calcola le percentuali di 'A' e 'B' per ciascuno dei tre subset creati in precedenza. Successivamente, crea tre grafici a torta, uno per ciascun subset, che mostrano la distribuzione delle classi ('A' e 'B') all'interno di ciascun subset. Infine, mostra i grafici a torta.

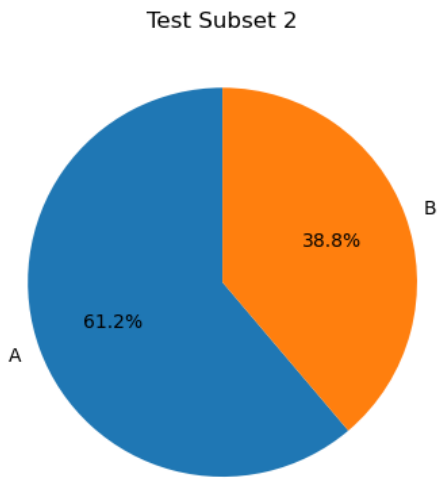
[18]:
```
# Dividere ciascun subset in training set e test set
train_subset1, test_subset1 = train_test_split(subset1, test_size=0.2,␣
 ↪random_state=42)
train_subset2, test_subset2 = train_test_split(subset2, test_size=0.2,␣
 ↪random_state=42)
train_subset3, test_subset3 = train_test_split(subset3, test_size=0.2,␣
 ↪random_state=42)

# Creare il grafico con 6 torte
fig, axs = plt.subplots(3, 2, figsize=(10, 12))

# Funzione per disegnare una torta con etichette
def draw_pie(ax, data, title):
    ax.pie(data, labels=data.index, autopct='%1.1f%%', startangle=90)
    ax.set_title(title)
    ## autopct='%1.1f%%'= formatta i valori per il grafico a torta
    # %f indica che il valore verrà formattato come un numero decimale.
    # 1.1 specifica che il numero decimale avrà una cifra intera e una cifra␣
 ↪decimale.
    # %% è utilizzato per rappresentare il simbolo percentuale.


## normalize=True, da % e non valore assoluto

# Prima riga di torte (Subset 1)
draw_pie(axs[0, 0], train_subset1['ColonnaAB'].value_counts(normalize=True),␣
 ↪'Train Subset 1')
draw_pie(axs[0, 1], test_subset1['ColonnaAB'].value_counts(normalize=True),␣
 ↪'Test Subset 1')

# Seconda riga di torte (Subset 2)
draw_pie(axs[1, 0], train_subset2['ColonnaAB'].value_counts(normalize=True),␣
 ↪'Train Subset 2')
draw_pie(axs[1, 1], test_subset2['ColonnaAB'].value_counts(normalize=True),␣
 ↪'Test Subset 2')

# Terza riga di torte (Subset 3)
```

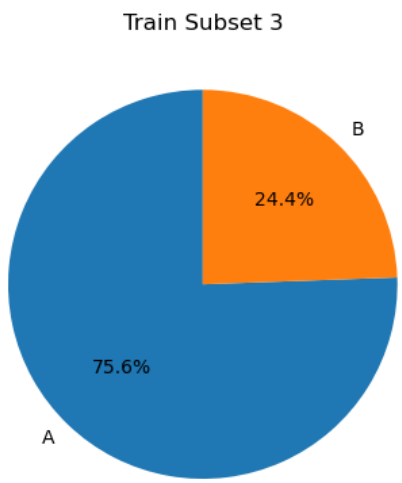
```
draw_pie(axs[2, 0], train_subset3['ColonnaAB'].value_counts(normalize=True),␣
 ↪'Train Subset 3')
draw_pie(axs[2, 1], test_subset3['ColonnaAB'].value_counts(normalize=True),␣
 ↪'Test Subset 3')

# Regolare lo spaziamento tra i subplots
plt.tight_layout()

# Mostrare il grafico
plt.show()
```

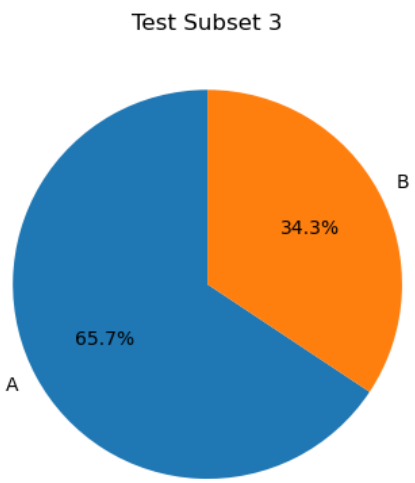

Train Subset 1
B
28.9%
71.1%
A


Test Subset 1
B
31.3%
68.7%
A


Train Subset 2
B
28.5%
71.5%
A


Test Subset 2
B
38.8%
61.2%
A


Train Subset 3
B
24.4%
75.6%
A


Test Subset 3
B
34.3%
65.7%
A

2.5.11 Questo codice divide ciascun subset in un set di addestramento e un set di test utilizzando la funzione train_test_split() dalla libreria scikit-learn. Successivamente, crea un grafico composto da 6 torte, organizzate in 3 righe e 2 colonne, per visualizzare la distribuzione delle classi ('A' e 'B') nei set di addestramento e di test per ciascun subset. Infine, mostra il grafico.