Data Splitting e Outliers

March 6, 2024

#

DATASET SPLITTING E OUTLIERS

0.1 INDICE GENERALE

DATASET SLITTING:

OUTLIERS:

1 DATASET SPLITTING

Per poter iniziare questa esercitazione, è necessario installare la libreria "numpy" attraverso il comando qui sotto incollandolo sul terminale di Windows (CMD) per Python "classico" o su una cella di Notebook Jupyter se si sta programmando Python lì, questa libreria permette di gestire diverse operazioni matematiche che di norma Python non riuscirebbe ad eseguire.

[ ]:
```
pip install numpy
```

2 L'APERTURA E LA LETTURA DI UN FILE CSV (COMMA SEPARATED VALUES)

In questo codice viene mostrato come importare da una cartella specifica del PC un file CSV e poi successivamente come leggerlo (tramite la funzione "pd.read_csv"). Infine il programma stampa anche le prime righe del DataFrame utilizzando il metodo "(df.head())" e attraverso l'attributo "shape" si possono visualizzare le proprietà del file, cioè il numero di righe (1017) e il numero di colonne (18). Inoltre quando si parla di DataFrame i rispettivi termini di colonna e riga rappresentano le Feature e le istanze.

[1]:
```
import pandas as pd # per la gestione dei DataFrame
import numpy as np  # per la gestione delle diverse operazioni matematiche
import matplotlib.pyplot as plt # per la creazione di grafici

# Specificare il percorso del file CSV
percorsofilecsv="C:\\Users\\matte\\OneDrive - Scuola Paritaria S. Freud␣
 ↪SRL\\Desktop\\FREUD\\2D\\QUADERNI E ALTRO\\ROBOTICA ED AI\\ESERCIZI IN␣
 ↪CLASSE PYTHON\\pokemons.csv"
# Leggere il file CSV in un DataFrame
df=pd.read_csv(percorsofilecsv) # per i file CSV si scrive "csv" nel pd.read
# Mostrare le prime righe del DataFrame
print(df.head()) # (df.head()) stampa solo le prime righe (istanze) del␣
 ↪DataFrame
df.shape # visualizza le proprietà del DataFrame, quindi il numero totale di␣
 ↪righe (istanze) e di colonne (Feature)
```

```
   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…
```

[1]:
```
(1017, 18)
```

3 L'APERTURA E LA LETTURA DI UN FILE XLSX (FILE "ORIGINALE" DI EXCEL)

Questo codice è equivalente a quello precedente tranne il fatto che in questo caso si sta importando un file XLSX (file classico di un Cartel di Excel). In questo programma viene inserito anche il parametro "sheet_name" della funzione "pd.read_excel" che permette di leggere un particolare foglio del Cartel in questione, in questo caso il foglio di nome "09-10".

[3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Specificare il percorso del file XLSX
percorsofileExcel="C:\\Users\\matte\\OneDrive - Scuola Paritaria S. Freud␣
 ↪SRL\\Desktop\\FREUD\\2D\\QUADERNI E ALTRO\\ROBOTICA ED AI\\ESERCIZI IN␣
 ↪CLASSE PYTHON\\serieA.xlsx"
# Leggere il file XLSX in un DataFrame
df=pd.read_excel(percorsofileExcel, sheet_name='09-10') # per i file XLSX si␣
 ↪scrive "excel" nel pd.read, il nome del programma
# Mostrare le prime righe del DataFrame
print(df.head())
```

```
   position                team  Pt  Played  Won  Net  lose  Goals made  \
0         1          Inter Inter  82      38   24   10     4          75
1         2            Roma Roma  80      38   24    8     6          68
2         3          Milan Milan  70      38   20   10     8          60
3         4  Sampdoria Sampdoria  67      38   19   10     9          49
4         5      Palermo Palermo  65      38   18   11     9          59

   Goals suffered  Difference goals
0              34                41
1              41                27
2              39                21
3              41                 8
4              47                12
```

Qui sotto vengono mostrare tutte le righe dell'ultimo DataFrame importato usando "df".

[4]:
```
# Stampare tutte le righe del DataFrame
df
```

[4]:

	position	team	Pt	Played	Won	Net	lose	Goals made	\
0	1	Inter Inter	82	38	24	10	4	75	
1	2	Roma Roma	80	38	24	8	6	68	
2	3	Milan Milan	70	38	20	10	8	60	
3	4	Sampdoria Sampdoria	67	38	19	10	9	49	
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

    Goals suffered  Difference goals
0               34                41
1               41                27
2               39                21
3               41                 8
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
```

Per poter continuare con questa esercitazione, è necessario installare la libreria "os" attraverso il comando qui sotto incollandolo sul terminale di Windows (CMD) per Python "classico" o su una cella di Notebook Jupyter se si sta programmando Python lì, questa libreria permette di gestire i path di sistema (percorsi file)

[ ]:
```
pip install os
```

4 L'APERTURA E LA LETTURA DI UNA CARTELLA E DI UN COLLEGAMENTO FILE

In questo programma viene spiegato come aprire e leggere i diversi file all'interno di una cartella (in questo caso solo CSV) per poi poterli unire in un unico DataFrame. Inizialmente viene creata una lista di cui conterrà tutti i DataFrame dei file CSV (un DataFrame per ogni CSV), poi successivamente viene specificato il percorso della cartella (path). A questo punto per automatizzare al meglio il processo di unione dei file CSV in unico DataFrame viene usato un ciclo for con una condizione al suo interno. Nel dettaglio il ciclo for in questione crea una "lista" (non salvata in una variabile) con tutti i nomi dei file nella cartella con la funzione "os.listdir()", in modo che ad ogni iterazione l'indice "nomedelfile" assume un diverso nome di file. Prima che "nomedelfile" assume un

nuovo valore, il ciclo for segue una condizione: se “nomedelfile” finisce con “.csv” allora quest’ultimo viene letto e aggiunto alla lista “listadataframes” creata all’inizio del codice appositamente.

[5]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os

# Creare variabile della lista dei DataFrame
listadataframes=[]
# Specificare il percorso della cartella
percorsofilecartella="C:\\Users\\matte\\OneDrive - Scuola Paritaria S. Freud␣
 ↪SRL\\Desktop\\FREUD\\2D\\QUADERNI E ALTRO\\ROBOTICA ED AI\\ESERCIZI IN␣
 ↪CLASSE PYTHON\\serieAnuovo"
for nomedelfile in os.listdir(percorsofilecartella): # os.listdir crea una␣
 ↪lista contenente tutti i nomi dei file
    # Si inserisce una condizione che permette di selezionare solo un␣
 ↪determinato file all'interno di una cartella
    if nomedelfile.endswith(".csv"): # endswith(".csv") significa che il file␣
 ↪deve terminare con .csv, quindi tutti e solo i file della cartella che␣
 ↪terminano con questa estensione file
        percorsofilecsv=os.path.join(percorsofilecartella, nomedelfile)#join␣
 ↪unisce entrambe le variabili dentro la parentesi
        df=pd.read_csv(percorsofilecsv)
        listadataframes.append(df)
print(listadataframes)
```

```
[    Div      Date   HomeTeam   AwayTeam  FTHG  FTAG FTR
0    I1  29/08/93   Atalanta   Cagliari     5     2   H
1    I1  29/08/93      Genoa       Roma     2     0   H
2    I1  29/08/93      Inter   Reggiana     2     1   H
3    I1  29/08/93   Juventus  Cremonese     1     0   H
4    I1  29/08/93      Lazio     Foggia     0     0   D
..   ..       …          …          …   …   …  ..
301  I1  01/05/94      Lecce   Cagliari     0     1   A
302  I1  01/05/94      Milan   Reggiana     0     1   A
303  I1  01/05/94      Parma   Piacenza     0     0   D
304  I1  01/05/94       Roma     Torino     2     0   H
305  I1  01/05/94  Sampdoria      Lazio     3     4   A

[306 rows x 7 columns],     Div      Date    HomeTeam  AwayTeam  FTHG  FTAG FTR
0    I1  04/09/94        Bari     Lazio     0     1   A
1    I1  04/09/94     Brescia  Juventus     1     1   D
2    I1  04/09/94  Fiorentina  Cagliari     2     1   H
3    I1  04/09/94       Milan     Genoa     1     0   H
4    I1  04/09/94      Napoli  Reggiana     1     0   H
..   ..       …          …         …   …   …  ..
301  I1  04/06/95       Inter    Padova     2     1   H
```

```
302  I1  04/06/95    Juventus  Cagliari     3     1   H
303  I1  04/06/95       Lazio   Brescia     1     0   H
304  I1  04/06/95      Napoli     Parma     1     0   H
305  I1  04/06/95    Reggiana    Foggia     1     1   D

[306 rows x 7 columns],     Div      Date    HomeTeam    AwayTeam  FTHG  FTAG
FTR  HTHG  HTAG HTR
0    I1  27/08/95    Atalanta       Parma     1     1   D     0     0   D
1    I1  27/08/95        Bari      Napoli     1     1   D     1     0   H
2    I1  27/08/95  Fiorentina      Torino     2     0   H     0     0   D
3    I1  27/08/95       Inter     Vicenza     1     0   H     0     0   D
4    I1  27/08/95    Juventus   Cremonese     4     1   H     1     0   H
..   ..       …         …          …   …   …  ..   …   …  ..
301  I1  12/05/96      Napoli     Udinese     2     1   H     1     1   D
302  I1  12/05/96    Piacenza  Fiorentina     0     1   A     0     1   A
303  I1  12/05/96        Roma       Inter     1     0   H     1     0   H
304  I1  12/05/96      Torino       Lazio     0     2   A     0     2   A
305  I1  12/05/96     Vicenza   Sampdoria     2     2   D     2     1   H

[306 rows x 10 columns],     Div      Date    HomeTeam    AwayTeam  FTHG  FTAG
FTR  HTHG  HTAG HTR
0    I1  08/09/96     Bologna       Lazio     1     0   H     1     0   H
1    I1  08/09/96    Cagliari    Atalanta     2     0   H     1     0   H
2    I1  08/09/96  Fiorentina     Vicenza     2     4   A     1     2   A
3    I1  08/09/96       Milan      Verona     4     1   H     0     1   A
4    I1  08/09/96       Parma      Napoli     3     0   H     1     0   H
..   ..       …         …          …   …   …  ..   …   …  ..
301  I1  01/06/97    Piacenza     Perugia     2     1   H     1     0   H
302  I1  01/06/97    Reggiana    Atalanta     0     3   A     0     1   A
303  I1  01/06/97        Roma     Udinese     0     3   A     0     1   A
304  I1  01/06/97   Sampdoria  Fiorentina     1     1   D     1     1   D
305  I1  01/06/97      Verona       Parma     1     2   A     1     1   D

[306 rows x 10 columns],     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],     Div      Date    HomeTeam     AwayTeam  FTHG  FTAG
```

```
FTR  HTHG  HTAG HTR
0    I1  12/09/98  Fiorentina       Empoli     2     0   H     1     0   H
1    I1  12/09/98       Milan      Bologna     3     0   H     0     0   D
2    I1  12/09/98       Parma      Vicenza     0     0   D     0     0   D
3    I1  12/09/98        Roma  Salernitana     3     1   H     0     1   A
4    I1  12/09/98     Udinese    Sampdoria     2     2   D     2     2   D
..   ..       …          …           …   …   …  ..   …   …  ..
301  I1  23/05/99       Lazio        Parma     2     1   H     1     0   H
302  I1  23/05/99     Perugia        Milan     1     2   A     1     2   A
303  I1  23/05/99    Piacenza  Salernitana     1     1   D     0     0   D
304  I1  23/05/99   Sampdoria         Bari     1     0   H     1     0   H
305  I1  23/05/99     Vicenza         Roma     1     4   A     1     1   D

[306 rows x 10 columns],     Div      Date    HomeTeam  AwayTeam  FTHG  FTAG FTR
HTHG  HTAG HTR
0    I1  28/08/99     Bologna    Torino     0     0   D     0     0   D
1    I1  29/08/99  Fiorentina      Bari     1     0   H     0     0   D
2    I1  29/08/99       Inter    Verona     3     0   H     1     0   H
3    I1  29/08/99    Juventus   Reggina     1     1   D     1     0   H
4    I1  29/08/99       Lecce     Milan     2     2   D     0     0   D
..   ..       …          …       …   …   …  ..   …   …  ..
301  I1  14/05/00       Milan   Udinese     4     0   H     2     0   H
302  I1  14/05/00       Parma     Lecce     4     1   H     1     0   H
303  I1  14/05/00     Perugia  Juventus     1     0   H     0     0   D
304  I1  14/05/00      Torino  Piacenza     2     1   H     2     1   H
305  I1  14/05/00      Verona      Roma     2     2   D     1     2   A

[306 rows x 10 columns],     Div      Date  HomeTeam    AwayTeam  FTHG  FTAG FTR
HTHG  HTAG HTR  …  \
0    I1  30/09/00      Bari      Verona     1     1   D     0     0   D  …
1    I1  30/09/00    Napoli    Juventus     1     2   A     1     0   H  …
2    I1  01/10/00  Atalanta       Lazio     2     2   D     1     1   D  …
3    I1  01/10/00     Milan     Vicenza     2     0   H     1     0   H  …
4    I1  01/10/00     Parma  Fiorentina     2     2   D     1     0   H  …
..   ..       …          …       …   …   …  ..   …   …  ..  …
301  I1  17/06/01     Lecce       Lazio     2     1   H     0     1   A  …
302  I1  17/06/01   Reggina       Milan     2     1   H     0     0   D  …
303  I1  17/06/01      Roma       Parma     3     1   H     2     0   H  …
304  I1  17/06/01   Udinese     Vicenza     2     3   A     2     3   A  …
305  I1  17/06/01    Verona     Perugia     2     1   H     1     0   H  …

      IWA   LBH   LBD    LBA   SBH   SBD    SBA   WHH   WHD   WHA
0    3.40  1.85  3.20   3.75  1.90  3.00   4.00  1.83  3.00   4.0
1    2.10  3.75  3.00   1.90  3.75  3.10   1.91  3.75  2.80   2.0
2    1.75  4.30  3.20   1.70  3.75  3.25   1.80  4.00  3.10   1.8
3    8.00  1.25  4.50  10.00  1.25  4.60  10.00  1.25  4.33  11.0
4    4.00  1.70  3.20   4.30  1.65  3.25   4.50  1.72  3.10   4.5
..    …   …   …    …   …   …    …   …   …   …
```

```
301  2.00   NaN   NaN    NaN  2.60  3.75   2.10  2.65  3.60   2.1
302   NaN   NaN   NaN    NaN  1.53  4.00   4.00   NaN   NaN   NaN
303  7.00  1.28  4.50   8.00  1.29  4.33   8.50  1.33  4.00   8.0
304   NaN  2.87  3.25   2.10  2.80  3.40   2.10  2.75  3.25   2.2
305   NaN  1.28  4.33   9.00  1.29  4.33   8.50  1.22  5.00   9.5

[306 rows x 25 columns],     Div      Date    HomeTeam    AwayTeam  FTHG  FTAG
FTR  HTHG  HTAG HTR  \
0     I1  25/08/01     Bologna    Atalanta     1     0   H     1     0   H
1     I1  26/08/01     Brescia       Milan     2     2   D     2     0   H
2     I1  26/08/01  Fiorentina      Chievo     0     2   A     0     1   A
3     I1  26/08/01       Inter     Perugia     4     1   H     2     0   H
4     I1  26/08/01    Juventus     Venezia     4     0   H     3     0   H
..    ..       …          …          …   …   …  ..   …   …  ..
301   I1  05/05/02       Parma     Venezia     2     1   H     1     1   D
302   I1  05/05/02     Perugia  Fiorentina     2     0   H     2     0   H
303   I1  05/05/02    Piacenza      Verona     3     0   H     1     0   H
304   I1  05/05/02      Torino        Roma     0     1   A     0     0   D
305   I1  05/05/02     Udinese    Juventus     0     2   A     0     2   A

      …  LBA   SBH    SBD    SBA   SYH   SYD    SYA    WHH   WHD    WHA
0     …  NaN   NaN    NaN    NaN  1.91  3.00   3.75   1.90  3.00   3.75
1     …  NaN   NaN    NaN    NaN  3.20  3.00   2.10   3.30  2.87   2.10
2     …  NaN   NaN    NaN    NaN  1.40  3.75   7.00   1.44  3.75   6.00
3     …  NaN   NaN    NaN    NaN  1.29  4.33   8.50   1.25  4.50  10.00
4     …  NaN   NaN    NaN    NaN  1.20  5.00  11.00   1.20  5.00  11.00
..    …   …    …      …      …    …     …     …      …     …     …
301   …  NaN   NaN    NaN    NaN   NaN   NaN    NaN   1.33  4.00   8.00
302   …  NaN   NaN    NaN    NaN   NaN   NaN    NaN   1.30  4.33   8.00
303   …  NaN   2.0  2.375  4.500   NaN   NaN    NaN   2.00  2.37   5.00
304   …  NaN   8.0  4.000  1.300   NaN   NaN    NaN  10.00  4.50   1.25
305   …  NaN  10.0  4.500  1.182   NaN   NaN    NaN    NaN   NaN    NaN

[306 rows x 28 columns],     Div      Date  HomeTeam  AwayTeam  FTHG  FTAG FTR
HTHG  HTAG HTR  …  \
0     I1  14/09/02   Bologna      Roma     2     1   H     0     1   A  …
1     I1  14/09/02      Como    Empoli     0     2   A     0     1   A  …
2     I1  14/09/02     Inter    Torino     1     0   H     1     0   H  …
3     I1  14/09/02    Modena     Milan     0     3   A     0     1   A  …
4     I1  15/09/02   Brescia  Piacenza     1     2   A     0     0   D  …
..    ..       …          …         …   …   …  ..   …   …  ..  …
301   I1  24/05/03     Inter   Perugia     2     2   D     1     0   H  …
302   I1  24/05/03  Juventus    Chievo     4     3   H     1     0   H  …
303   I1  24/05/03  Piacenza     Milan     4     2   H     3     1   H  …
304   I1  24/05/03      Roma  Atalanta     1     2   A     1     1   D  …
305   I1  24/05/03   Udinese     Lazio     2     1   H     0     0   D  …

       SBH    SBD     SBA   WHH   WHD    WHA  GB>2.5  GB<2.5  B365>2.5  \
```

```
0    3.000  2.875   2.375  3.00  2.80   2.30    2.05    1.75       NaN
1    2.000  3.000   3.750  1.90  3.00   3.75    2.00    1.77       NaN
2    1.300  4.333  10.000  1.30  4.00   9.50    1.67    1.95       NaN
3    4.750  3.100   1.750  4.50  2.90   1.80    1.90    1.86       NaN
4    1.909  3.000   4.000  1.90  2.87   4.00    1.95    1.65       NaN
..     …      …       …     …     …      …       …       …
301  1.333  3.750   8.500  1.25  4.00  13.00     NaN     NaN       NaN
302    NaN    NaN     NaN  3.25  2.50   2.37     NaN     NaN       NaN
303  2.750  2.800   2.400  3.00  2.87   2.25     NaN     NaN       NaN
304    NaN    NaN     NaN  3.20  3.20   2.00     NaN     NaN       NaN
305  1.500  3.750   5.000  1.57  3.75   4.50     NaN     NaN       NaN

     B365<2.5
0         NaN
1         NaN
2         NaN
3         NaN
4         NaN
..        …
301       NaN
302       NaN
303       NaN
304       NaN
305       NaN

[306 rows x 35 columns],     Div      Date   HomeTeam   AwayTeam  FTHG  FTAG FTR
HTHG  HTAG HTR  …  \
0    I1  30/08/03    Reggina  Sampdoria     2     2   D     2     0   H  …
1    I1  31/08/03    Bologna      Parma     2     2   D     1     1   D  …
2    I1  31/08/03    Brescia     Chievo     1     1   D     0     1   A  …
3    I1  31/08/03      Inter     Modena     2     0   H     0     0   D  …
4    I1  31/08/03   Juventus     Empoli     5     1   H     1     0   H  …
..   ..       …        …        …   …   …  ..   …   …  ..  …
301  I1  16/05/04      Milan    Brescia     4     2   H     2     0   H  …
302  I1  16/05/04      Parma    Udinese     4     3   H     0     0   D  …
303  I1  16/05/04    Perugia     Ancona     1     0   H     0     0   D  …
304  I1  16/05/04  Sampdoria       Roma     0     0   D     0     0   D  …
305  I1  16/05/04      Siena   Juventus     1     3   A     1     2   A  …

     GBAHH  GBAHA  GBAH  LBAHH  LBAHA  LBAH  B365AHH  B365AHA  B365AH  \
0     2.00   1.80 -0.25   2.05   1.80 -0.25    2.075    1.825   -0.25
1     2.10   1.75 -0.25   2.10   1.75 -0.25    1.750    2.150    0.00
2     2.05   1.77 -0.25   2.10   1.75 -0.25    1.750    2.150    0.00
3     1.85   1.95 -1.50   1.90   1.95 -1.50    1.925    1.975   -1.50
4     1.85   1.95 -1.50   1.85   2.00 -1.50    1.875    2.025   -1.50
..     …      …    …      …      …    …        …        …       …
301    NaN    NaN   NaN    NaN    NaN   NaN      NaN      NaN     NaN
302    NaN    NaN   NaN    NaN    NaN   NaN      NaN      NaN     NaN
```

```
303    NaN    NaN   NaN    NaN    NaN   NaN      NaN      NaN     NaN
304    NaN    NaN   NaN    NaN    NaN   NaN      NaN      NaN     NaN
305    NaN    NaN   NaN    NaN    NaN   NaN      NaN      NaN     NaN

     Unnamed: 44
0            NaN
1            NaN
2            NaN
3            NaN
4            NaN
..           …
301          NaN
302          NaN
303          NaN
304          NaN
305          NaN

[306 rows x 45 columns],     Div      Date  HomeTeam  AwayTeam  FTHG  FTAG FTR
HTHG  HTAG HTR  …  \
0    I1  11/09/04    Chievo     Inter     2     2   D     2     1   H  …
1    I1  11/09/04     Milan   Livorno     2     2   D     1     1   D  …
2    I1  12/09/04  Atalanta     Lecce     2     2   D     1     1   D  …
3    I1  12/09/04   Brescia  Juventus     0     3   A     0     2   A  …
4    I1  12/09/04  Cagliari   Bologna     1     0   H     1     0   H  …
..   ..       …       …       …   …   …  ..   …   …  ..  …
375  I1  29/05/05   Messina   Livorno     1     1   D     0     0   D  …
376  I1  29/05/05   Palermo     Lazio     3     3   D     1     1   D  …
377  I1  29/05/05      Roma    Chievo     0     0   D     0     0   D  …
378  I1  29/05/05     Siena  Atalanta     2     1   H     1     0   H  …
379  I1  29/05/05   Udinese     Milan     1     1   D     0     0   D  …

     GBAHH  GBAHA  GBAH  LBAHH  LBAHA  LBAH  B365AHH  B365AHA  B365AH  \
0     2.00   1.80  0.75   1.94   1.90  0.75    1.950    1.950    0.75
1     1.80   2.00 -1.75   1.86   1.98 -1.75    1.850    2.050   -1.75
2     2.10   1.61 -0.50   2.02   1.82 -0.50    2.100    1.800   -0.50
3     1.81   1.80  0.50   2.04   1.80  0.50    2.100    1.800    0.50
4     2.05   1.77 -0.25   2.06   1.78 -0.25    2.125    1.775   -0.25
..     …    …   …    …    …   …       …       …      …
375    NaN    NaN   NaN    NaN    NaN   NaN    2.125    1.775   -0.25
376    NaN    NaN   NaN    NaN    NaN   NaN      NaN      NaN     NaN
377    NaN    NaN   NaN    NaN    NaN   NaN    2.175    1.725    0.00
378    NaN    NaN   NaN    NaN    NaN   NaN    2.125    1.775   -1.50
379    NaN    NaN   NaN    NaN    NaN   NaN    1.875    2.025   -0.75

     Unnamed: 44
0            NaN
1            NaN
2            NaN
```

```
3            NaN
4            NaN
..           …
375          NaN
376          NaN
377          NaN
378          NaN
379          NaN

[380 rows x 45 columns],     Div      Date    HomeTeam   AwayTeam  FTHG  FTAG
FTR  HTHG  HTAG HTR  …  \
0    I1  27/08/05  Fiorentina  Sampdoria     2     1   H     2     0   H  …
1    I1  27/08/05     Livorno      Lecce     2     1   H     1     1   D  …
2    I1  28/08/05      Ascoli      Milan     1     1   D     0     0   D  …
3    I1  28/08/05       Inter    Treviso     3     0   H     1     0   H  …
4    I1  28/08/05    Juventus     Chievo     1     0   H     1     0   H  …
..   ..       …         …        …   …   …  ..   …   …  ..  …
375  I1  14/05/06     Palermo    Messina     1     0   H     1     0   H  …
376  I1  14/05/06     Reggina   Juventus     0     2   A     0     1   A  …
377  I1  14/05/06   Sampdoria      Lecce     1     3   A     0     1   A  …
378  I1  14/05/06       Siena    Livorno     0     0   D     0     0   D  …
379  I1  14/05/06     Treviso    Udinese     2     1   H     1     1   D  …

    BbMx>2.5  BbAv>2.5  BbMx<2.5  BbAv<2.5  BbAH  BbAHh  BbMxAHH  BbAvAHH  \
0       2.30      2.03      1.75      1.66  16.0  -0.25     2.00     1.95
1       2.00      1.85      2.00      1.84  17.0  -0.50     2.00     1.96
2       1.90      1.78      2.05      1.89  15.0   1.25     2.10     2.05
3       1.65      1.55      2.40      2.23  15.0  -1.75     1.95     1.90
4       1.85      1.71      2.10      1.96  15.0  -1.50     1.90     1.87
..       …       …         …         …   …    …       …        …
375     1.67      1.61      2.35      2.16  17.0  -1.25     2.05     1.99
376     1.71      1.63      2.25      2.12  19.0   1.50     1.99     1.85
377     1.75      1.65      2.25      2.07  15.0  -0.75     1.80     1.77
378     1.80      1.70      2.16      2.03  17.0  -0.25     2.11     2.07
379     1.95      1.81      2.00      1.93  16.0   0.25     1.86     1.83

     BbMxAHA  BbAvAHA
0       1.96     1.92
1       1.95     1.89
2       1.86     1.82
3       2.01     1.97
4       2.07     1.98
..       …      …
375     1.93     1.90
376     2.07     1.98
377     2.16     2.13
378     1.87     1.83
379     2.09     2.07
```

```
[380 rows x 68 columns],     Div      Date    HomeTeam AwayTeam  FTHG  FTAG FTR
HTHG  HTAG HTR  …  \
0    I1  09/09/06  Fiorentina    Inter     2     3   A     0     2   A  …
1    I1  09/09/06       Roma  Livorno     2     0   H     1     0   H  …
2    I1  10/09/06    Atalanta   Ascoli     3     1   H     3     0   H  …
3    I1  10/09/06    Cagliari  Catania     0     1   A     0     0   D  …
4    I1  10/09/06      Chievo    Siena     1     2   A     1     0   H  …
..   ..       …         …       …   …   …  ..   …   …  ..   …
375  I1  27/05/07       Parma   Empoli     3     1   H     2     1   H  …
376  I1  27/05/07     Reggina    Milan     2     0   H     1     0   H  …
377  I1  27/05/07        Roma  Messina     4     3   H     2     1   H  …
378  I1  27/05/07       Siena    Lazio     2     1   H     1     0   H  …
379  I1  27/05/07     Udinese  Palermo     1     2   A     1     1   D  …

    BbMx>2.5  BbAv>2.5  BbMx<2.5  BbAv<2.5  BbAH  BbAHh  BbMxAHH  BbAvAHH  \
0       2.00      1.88      1.90      1.81    19   0.00     2.22     2.10
1       1.96      1.83      1.95      1.89    17  -1.00     1.84     1.79
2       2.25      2.12      1.67      1.62    17  -0.50     1.98     1.94
3       2.20      2.01      1.80      1.70    13  -0.50     1.88     1.81
4       2.10      1.97      1.83      1.73    17  -0.50     1.86     1.81
..       …         …         …         …   …     …        …        …
375     1.90      1.80      2.04      1.92     8  -0.50     1.91     1.87
376     1.70      1.62      2.30      2.12     9  -1.25     2.20     2.15
377     1.45      1.35      3.20      2.81    10  -1.75     1.73     1.71
378     1.70      1.61      2.30      2.11     9  -1.25     2.05     2.00
379     1.65      1.59      2.30      2.13     9  -0.25     2.18     2.14

     BbMxAHA  BbAvAHA
0       1.73     1.69
1       2.11     2.05
2       1.97     1.94
3       2.15     2.06
4       2.15     2.06
..       …        …
375     2.05     2.02
376     1.80     1.77
377     2.28     2.20
378     1.95     1.89
379     1.79     1.76

[380 rows x 68 columns],     Div      Date    HomeTeam    AwayTeam  FTHG  FTAG
FTR  HTHG  HTAG HTR  \
0    I1  25/08/07    Juventus     Livorno     5     1   H     1     0   H
1    I1  25/08/07       Lazio      Torino     2     2   D     0     1   A
2    I1  26/08/07  Fiorentina      Empoli     3     1   H     0     0   D
3    I1  26/08/07       Genoa       Milan     0     3   A     0     3   A
4    I1  26/08/07       Inter     Udinese     1     1   D     1     0   H
```

```
..   ..       …          …          …   …   …  ..   …   …  ..
375  I1  18/05/08       Lazio      Napoli     2     1   H     1     0   H
376  I1  18/05/08       Milan     Udinese     4     1   H     0     1   A
377  I1  18/05/08       Parma       Inter     0     2   A     0     0   D
378  I1  18/05/08       Siena     Palermo     2     2   D     1     2   A
379  I1  18/05/08      Torino  Fiorentina     0     1   A     0     0   D

     …  BbMx>2.5  BbAv>2.5  BbMx<2.5  BbAv<2.5  BbAH  BbAHh  BbMxAHH  \
0    …       2.02      1.92      1.92      1.83    19  -1.25     1.93
1    …       2.34      2.17      1.67      1.62    19  -0.75     2.19
2    …       2.02      1.93      1.90      1.82    20  -1.00     2.13
3    …       2.10      1.97      1.86      1.76    19   0.75     1.95
4    …       1.78      1.72      2.17      2.04    20  -1.50     2.17
..   …        …        …         …         …    …    …       …
375  …       1.65      1.60      2.35      2.21    17  -0.25     2.00
376  …       1.60      1.56      2.50      2.29    16  -1.50     2.09
377  …       1.70      1.60      2.40      2.21    22   1.00     2.02
378  …       1.70      1.67      2.23      2.10    16  -0.25     1.97
379  …       1.83      1.70      2.16      2.05    17   1.00     1.91

     BbAvAHH  BbMxAHA  BbAvAHA
0       1.86     2.08     2.02
1       2.11     1.83     1.77
2       2.04     1.85     1.81
3       1.90     2.05     1.97
4       2.11     1.82     1.77
..       …       …        …
375     1.94     1.97     1.93
376     2.03     1.88     1.83
377     1.94     1.93     1.90
378     1.92     2.02     1.95
379     1.87     2.07     1.97

[380 rows x 70 columns],     Div      Date   HomeTeam   AwayTeam  FTHG  FTAG FTR
HTHG  HTAG HTR  …  \
0    I1  30/08/08  Sampdoria      Inter     1     1   D     0     1   A  …
1    I1  30/08/08    Udinese    Palermo     3     1   H     2     0   H  …
2    I1  31/08/08   Atalanta      Siena     1     0   H     1     0   H  …
3    I1  31/08/08   Cagliari      Lazio     1     4   A     1     0   H  …
4    I1  31/08/08    Catania      Genoa     1     0   H     0     0   D  …
..   ..       …        …         …   …   …  ..   …   …  ..  …
375  I1  31/05/09     Napoli     Chievo     3     0   H     3     0   H  …
376  I1  31/05/09    Palermo  Sampdoria     2     2   D     2     1   H  …
377  I1  31/05/09    Reggina      Siena     1     1   D     1     0   H  …
378  I1  31/05/09       Roma     Torino     3     2   H     1     1   D  …
379  I1  31/05/09    Udinese   Cagliari     6     2   H     3     0   H  …

     BbMx>2.5  BbAv>2.5  BbMx<2.5  BbAv<2.5  BbAH  BbAHh  BbMxAHH  BbAvAHH  \
```

```
0       2.17      2.00      1.83      1.74  17.0   0.50     1.97     1.92
1       2.00      1.92      1.92      1.81  17.0   0.00     1.50     1.45
2       2.05      1.95      1.88      1.79  17.0  -0.50     2.10     2.06
3       2.22      2.12      1.70      1.65  18.0   0.00     2.10     2.03
4       2.25      2.12      1.70      1.65  17.0   0.00     1.70     1.64
..       …         …         …         …    …     …        …        …
375     1.70      1.62      2.35      2.18  14.0  -0.75     1.90     1.86
376     1.55      1.48      2.67      2.48  14.0  -0.50     1.90     1.87
377     1.73      1.67      2.20      2.09  14.0  -0.25     1.99     1.96
378     1.50      1.45      2.85      2.65  10.0  -0.75     1.90     1.87
379     1.55      1.48      2.70      2.51  14.0  -0.50     1.88     1.85

     BbMxAHA  BbAvAHA
0       1.99     1.96
1       2.75     2.57
2       1.87     1.82
3       1.86     1.79
4       2.31     2.21
..       …        …
375     2.08     2.01
376     2.05     2.00
377     1.94     1.92
378     2.05     2.02
379     2.08     2.02

[380 rows x 70 columns],     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],     Div      Date    HomeTeam   AwayTeam  FTHG  FTAG
FTR  HTHG  HTAG HTR  …  \
0    I1  28/08/10        Roma     Cesena     0     0   D     0     0   D  …
1    I1  28/08/10     Udinese      Genoa     0     1   A     0     0   D  …
2    I1  29/08/10        Bari   Juventus     1     0   H     1     0   H  …
3    I1  29/08/10      Chievo    Catania     2     1   H     1     1   D  …
4    I1  29/08/10  Fiorentina     Napoli     1     1   D     0     1   A  …
..   ..       …         …        …   …   …  ..   …   …  ..  …
375  I1  22/05/11    Juventus     Napoli     2     2   D     0     1   A  …
376  I1  22/05/11       Lecce      Lazio     2     4   A     2     2   D  …
377  I1  22/05/11     Palermo     Chievo     1     3   A     1     1   D  …
378  I1  22/05/11        Roma  Sampdoria     3     1   H     1     1   D  …
379  I1  22/05/11     Udinese      Milan     0     0   D     0     0   D  …

     BbAv<2.5  BbAH  BbAHh  BbMxAHH  BbAvAHH  BbMxAHA  BbAvAHA  Unnamed: 70  \
0        2.02    14  -1.25     1.90     1.86     2.05     2.01          NaN
1        1.84    23   0.00     1.75     1.68     2.23     2.12          NaN
2        1.70    17   0.00     3.00     2.72     1.50     1.41          NaN
3        1.65    21   0.00     1.63     1.54     2.55     2.40          NaN
4        1.75    21   0.00     1.67     1.61     2.43     2.23          NaN
..        …   …    …       …        …        …      …          …
375      2.27    20  -0.50     1.92     1.87     2.07     2.01          NaN
376      2.26    16   0.75     2.11     2.06     1.85     1.81          NaN
377      2.46    18  -0.75     2.11     2.07     1.84     1.81          NaN
378      2.60    20  -1.50     2.08     2.02     1.92     1.85          NaN
379      1.81    16   0.00     1.52     1.47     2.81     2.73          NaN

     Unnamed: 71  Unnamed: 72
0            NaN          NaN
1            NaN          NaN
2            NaN          NaN
```

```
3            NaN           NaN
4            NaN           NaN
..           …           …
375          NaN           NaN
376          NaN           NaN
377          NaN           NaN
378          NaN           NaN
379          NaN           NaN

[380 rows x 73 columns],     Div      Date    HomeTeam  AwayTeam  FTHG  FTAG FTR
HTHG  HTAG HTR  …  \
0    I1  09/09/11       Milan     Lazio     2     2   D     2     2   D  …
1    I1  10/09/11      Cesena    Napoli     1     3   A     1     1   D  …
2    I1  11/09/11     Catania     Siena     0     0   D     0     0   D  …
3    I1  11/09/11      Chievo    Novara     2     2   D     2     1   H  …
4    I1  11/09/11  Fiorentina   Bologna     2     0   H     1     0   H  …
..   ..       …         …       …   …   …  ..   …   …  ..  …
375  I1  13/05/12    Juventus  Atalanta     3     1   H     2     0   H  …
376  I1  13/05/12       Lazio     Inter     3     1   H     0     1   A  …
377  I1  13/05/12       Milan    Novara     2     1   H     0     1   A  …
378  I1  13/05/12      Napoli     Siena     2     1   H     2     1   H  …
379  I1  13/05/12       Parma   Bologna     1     0   H     1     0   H  …

     BbMx>2.5  BbAv>2.5  BbMx<2.5  BbAv<2.5  BbAH  BbAHh  BbMxAHH  BbAvAHH  \
0        2.05      1.95      1.91      1.82    20  -1.00     2.11     2.06
1        2.38      2.21      1.70      1.63    20   0.50     1.76     1.69
2        2.43      2.29      1.62      1.58    22  -0.50     2.14     2.07
3        2.43      2.25      1.66      1.61    23  -0.50     1.93     1.88
4        2.25      2.12      1.76      1.68    21  -0.75     1.86     1.83
..        …       …       …       …   …    …       …       …
375      1.55      1.48      2.78      2.57    19  -1.50     1.97     1.92
376      1.70      1.63      2.41      2.21    17   0.00     2.21     2.07
377      1.50      1.45      2.90      2.65    17  -1.75     1.99     1.93
378      1.64      1.57      2.56      2.32    20  -1.50     2.03     1.98
379      1.63      1.59      2.41      2.27    18  -0.50     1.99     1.93

     BbMxAHA  BbAvAHA
0       1.86     1.81
1       2.28     2.21
2       1.83     1.78
3       2.01     1.95
4       2.08     2.04
..       …      …
375     2.00     1.94
376     1.85     1.76
377     1.99     1.96
378     1.98     1.88
379     2.03     1.93
```

```
[380 rows x 70 columns],     Div      Date    HomeTeam    AwayTeam  FTHG  FTAG
FTR  HTHG  HTAG HTR  \
0    I1  25/08/12  Fiorentina     Udinese     2     1   H   0.0   1.0   A
1    I1  25/08/12    Juventus       Parma     2     0   H   0.0   0.0   D
2    I1  26/08/12    Atalanta       Lazio     0     1   A   0.0   1.0   A
3    I1  26/08/12      Chievo     Bologna     2     0   H   0.0   0.0   D
4    I1  26/08/12       Genoa    Cagliari     2     0   H   0.0   0.0   D
..   ..       …         …          …   …   …  ..   …   …  ..
375  I1  19/05/13     Palermo       Parma     1     3   A   0.0   3.0   A
376  I1  19/05/13     Pescara  Fiorentina     1     5   A   0.0   3.0   A
377  I1  19/05/13        Roma      Napoli     2     1   H   0.0   0.0   D
378  I1  19/05/13       Siena       Milan     1     2   A   1.0   0.0   H
379  I1  19/05/13      Torino     Catania     2     2   D   0.0   1.0   A

     …  BbAv<2.5  BbAH  BbAHh  BbMxAHH  BbAvAHH  BbMxAHA  BbAvAHA   PSCH  \
0    …       1.66    20  -0.25     1.85     1.76     2.19     2.11   1.97
1    …       2.04    23  -1.50     2.00     1.96     1.96     1.91   1.36
2    …       1.62    21  -0.25     2.12     2.05     1.86     1.81   2.68
3    …       1.57    24  -0.25     1.90     1.83     2.09     2.03   2.04
4    …       1.71    21  -0.50     2.06     2.03     1.88     1.84   2.64
..   …        …   …    …       …       …       …       …    …
375  …       2.14    23  -0.25     2.04     1.96     1.94     1.90   2.62
376  …       2.64    24   2.00     1.85     1.80     2.15     2.06  15.04
377  …       2.65    22  -0.25     2.27     2.18     1.75     1.71   2.52
378  …       2.52    25   2.00     1.81     1.73     2.21     2.14  13.50
379  …       1.98    22  -0.25     1.86     1.82     2.10     2.04   2.42

     PSCD   PSCA
0    3.31   4.74
1    5.03  11.65
2    3.20   2.98
3    3.25   4.45
4    3.14   3.10
..    …    …
375  3.45   2.85
376  7.26   1.23
377  3.86   2.72
378  6.80   1.25
379  3.20   3.36

[380 rows x 73 columns],     Div      Date   HomeTeam  AwayTeam  FTHG  FTAG FTR
HTHG  HTAG HTR  …  \
0    I1  24/08/13  Sampdoria  Juventus     0     1   A     0     0   D  …
1    I1  24/08/13     Verona     Milan     2     1   H     1     1   D  …
2    I1  25/08/13   Cagliari  Atalanta     2     1   H     1     1   D  …
3    I1  25/08/13      Inter     Genoa     2     0   H     0     0   D  …
4    I1  25/08/13      Lazio   Udinese     2     1   H     2     0   H  …
```

```
..   ..       …        …       …   …   …  ..   …   …  ..  …
375  I1  18/05/14   Juventus  Cagliari     3     0   H     3     0   H  …
376  I1  18/05/14      Lazio   Bologna     1     0   H     0     0   D  …
377  I1  18/05/14      Milan  Sassuolo     2     1   H     2     0   H  …
378  I1  18/05/14     Napoli    Verona     5     1   H     3     0   H  …
379  I1  18/05/14      Parma   Livorno     2     0   H     0     0   D  …

     BbAv<2.5  BbAH  BbAHh  BbMxAHH  BbAvAHH  BbMxAHA  BbAvAHA  PSCH  PSCD  \
0        1.80    19   1.00     2.09     2.04     1.88     1.83  8.38  4.53
1        1.82    18   1.00     1.69     1.58     2.51     2.42  5.75  3.59
2        1.60    16   0.00     1.78     1.72     2.21     2.14  2.54  3.22
3        1.87    19  -1.00     2.26     2.14     1.78     1.73  1.65  4.05
4        1.79    16  -0.50     2.00     1.96     1.94     1.91  1.93  3.48
..        …   …    …       …        …        …       …   …   …
375      2.50    22  -1.75     1.95     1.89     2.01     1.97  1.21  7.25
376      2.28    21  -1.00     1.83     1.78     2.15     2.08  1.34  5.49
377      2.76    22  -1.75     2.11     2.05     1.91     1.82  1.28  6.67
378      2.99    22  -1.00     1.71     1.68     2.33     2.24  1.61  4.50
379      2.64    20  -1.75     1.92     1.87     2.05     1.99  1.20  7.60

      PSCA
0     1.47
1     1.76
2     3.18
3     5.92
4     4.63
..     …
375  18.39
376  11.08
377  10.96
378   5.65
379  17.51

[380 rows x 67 columns],     Div      Date   HomeTeam    AwayTeam  FTHG  FTAG
FTR  HTHG  HTAG HTR  …  \
0    I1  30/08/14     Chievo    Juventus     0     1   A     0     1   A  …
1    I1  30/08/14       Roma  Fiorentina     2     0   H     1     0   H  …
2    I1  31/08/14   Atalanta      Verona     0     0   D     0     0   D  …
3    I1  31/08/14     Cesena       Parma     1     0   H     1     0   H  …
4    I1  31/08/14      Genoa      Napoli     1     2   A     1     1   D  …
..   ..       …        …          …   …   …  ..   …   …  ..  …
375  I1  31/05/15     Napoli       Lazio     2     4   A     0     2   A  …
376  I1  31/05/15       Roma     Palermo     1     2   A     0     1   A  …
377  I1  31/05/15  Sampdoria       Parma     2     2   D     0     0   D  …
378  I1  31/05/15   Sassuolo       Genoa     3     1   H     3     1   H  …
379  I1  31/05/15     Torino      Cesena     5     0   H     3     0   H  …

     BbAv<2.5  BbAH  BbAHh  BbMxAHH  BbAvAHH  BbMxAHA  BbAvAHA  PSCH  PSCD  \
```

```
0        1.80    27   1.00     2.00     1.94     1.98     1.92  8.35  3.84
1        1.94    21  -0.75     2.00     1.95     1.98     1.92  1.73  3.80
2        1.70    21  -0.50     2.15     2.10     1.84     1.80  2.07  3.47
3        1.59    21   0.25     1.91     1.86     2.08     2.01  5.44  3.65
4        1.82    21   0.50     1.91     1.87     2.04     2.01  3.92  3.54
..        …   …    …       …        …        …        …   …   …
375      2.03    25  -0.50     2.15     2.06     1.85     1.78  1.92  3.75
376      2.54    27  -1.50     2.20     2.11     1.85     1.74  1.66  4.35
377      2.62    28  -1.50     2.11     2.03     1.92     1.83  1.52  4.64
378      2.62    25   0.25     1.83     1.75     2.17     2.10  3.15  3.80
379      2.46    28  -1.50     2.10     2.02     1.87     1.83  1.39  5.43

     PSCA
0    1.56
1    5.56
2    3.99
3    1.76
4    2.07
..    …
375  4.30
376  5.32
377  6.85
378  2.28
379  8.52

[380 rows x 67 columns],     Div      Date    HomeTeam    AwayTeam  FTHG  FTAG
FTR  HTHG  HTAG HTR  \
0    I1  22/08/15       Lazio     Bologna     2     1   H     2     1   H
1    I1  22/08/15      Verona        Roma     1     1   D     0     0   D
2    I1  23/08/15      Empoli      Chievo     1     3   A     1     0   H
3    I1  23/08/15  Fiorentina       Milan     2     0   H     1     0   H
4    I1  23/08/15   Frosinone      Torino     1     2   A     1     0   H
..   ..       …         …          …   …   …  ..   …   …  ..
375  I1  15/05/16      Empoli      Torino     2     1   H     1     0   H
376  I1  15/05/16       Genoa    Atalanta     1     2   A     0     0   D
377  I1  15/05/16       Lazio  Fiorentina     2     4   A     1     3   A
378  I1  15/05/16     Palermo      Verona     3     2   H     1     0   H
379  I1  15/05/16     Udinese       Carpi     1     2   A     0     2   A

     …  BbAv<2.5  BbAH  BbAHh  BbMxAHH  BbAvAHH  BbMxAHA  BbAvAHA  PSCH  \
0    …       1.94    25  -1.25     2.12     2.04     1.86     1.80  1.48
1    …       1.90    26   0.75     2.01     1.95     1.94     1.88  5.68
2    …       1.57    26  -0.25     1.91     1.85     2.07     1.99  2.34
3    …       1.88    26  -0.25     1.88     1.83     2.08     2.01  2.31
4    …       1.57    26   0.25     2.07     1.98     1.90     1.86  3.77
..   …        …   …    …       …        …        …      …   …
375  …       2.40    27  -0.25     2.31     2.20     1.74     1.70  2.71
376  …       1.57    29  -0.50     1.95     1.89     2.03     1.97  1.59
```

```
377  …      2.23    30  -0.25     1.92     1.87     2.05     1.99  1.92
378  …      2.84    28  -1.75     1.94     1.87     2.05     1.99  1.28
379  …      2.00    30  -0.25     2.02     1.97     1.96     1.88  2.08

     PSCD   PSCA
0    4.32   8.99
1    3.85   1.71
2    3.31   3.45
3    3.35   3.47
4    3.44   2.16
..    …     …
375  3.67   2.64
376  4.58   5.80
377  4.17   3.85
378  6.56  11.89
379  3.79   3.61

[380 rows x 64 columns],     Div      Date   HomeTeam    AwayTeam  FTHG  FTAG
FTR  HTHG  HTAG HTR  …  \
0    I1  20/08/16   Juventus  Fiorentina     2     1   H   1.0   0.0   H  …
1    I1  20/08/16       Roma     Udinese     4     0   H   0.0   0.0   D  …
2    I1  21/08/16   Atalanta       Lazio     3     4   A   0.0   3.0   A  …
3    I1  21/08/16    Bologna     Crotone     1     0   H   0.0   0.0   D  …
4    I1  21/08/16     Chievo       Inter     2     0   H   0.0   0.0   D  …
..   ..       …         …          …   …   …  ..   …   …  ..  …
375  I1  28/05/17      Inter     Udinese     5     2   H   3.0   0.0   H  …
376  I1  28/05/17    Palermo      Empoli     2     1   H   0.0   0.0   D  …
377  I1  28/05/17       Roma       Genoa     3     2   H   1.0   1.0   D  …
378  I1  28/05/17  Sampdoria      Napoli     2     4   A   0.0   2.0   A  …
379  I1  28/05/17     Torino    Sassuolo     5     3   H   3.0   2.0   H  …

     BbAv<2.5  BbAH  BbAHh  BbMxAHH  BbAvAHH  BbMxAHA  BbAvAHA   PSCH   PSCD  \
0        1.78    36  -1.00     1.85     1.82     2.11     2.04   1.51   4.15
1        2.04    32  -1.50     2.45     2.31     1.72     1.63   1.46   4.72
2        1.63    31   0.25     1.85     1.80     2.15     2.07   2.96   3.27
3        1.53    31  -0.50     1.95     1.87     2.06     1.98   1.78   3.50
4        1.60    31   0.25     2.16     2.09     1.84     1.77   4.14   3.36
..        …   …    …       …        …        …       …    …     …
375      3.39    19  -1.50     1.94     1.91     1.99     1.95   1.42   5.40
376      2.17    19   1.00     1.90     1.85     2.07     2.02   6.00   4.50
377      4.67    15  -3.00     1.90     1.84     2.10     2.02   1.06  16.40
378      4.32    15   2.25     1.99     1.95     1.94     1.91  10.15   7.10
379      3.62    19  -1.00     2.03     1.97     1.93     1.88   1.85   4.19

      PSCA
0     8.61
1     8.12
2     2.66
```

```
3     5.79
4     2.08
..     …
375   7.75
376   1.58
377  42.00
378   1.28
379   4.13

[380 rows x 64 columns]]
```

5 IL CONTEGGIO DEI DATI DI UN DATAFRAME

Il comando qui sotto permette di sapere il numero di DataFrame presenti nella lista "listadataframes". Len è l'acronimo di "length" cioè lunghezza.

[15]:
```
len(listadataframes)
```

[15]: 24

Il comando "len(listadataframes[23])" permette al programma di contare il numero di istanze (righe) all'interno del DataFrame n°23.

[16]:
```
len(listadataframes[23])
```

[16]: 380

Invece il comando "len(listadataframes[23].columns)" permette al programma di contare il numero di colonne (Feature) all'interno del DataFrame n°23. Quindi riassumendo: se si desidera sapere il numero istanze basta indicare il nome della lista e il numero del singolo DataFrame, invece se si desidera sapere il numero di Feature bisogna aggiungere l'attributo ".columns".

[17]:
```
len(listadataframes[23].columns)
```

[17]: 64

Il comando sottostante legge il DataFrame scelto dall'utente via input. Bisogna sottolineare il fatto che su Python l'indice (cioè la numerazione) parte dal numero 0 infatti sono 23 gli effettivi DataFrame e non 24 come stampava l'output di un codice precedente, in ogni caso il numero 24 è giusto se si inizia a contare dal numero 1 e non dallo 0.

[23]:
```
DataFramescelto=int(input("Quale DataFrame si desidera visualizzare? "))
print("Eccolo qua!!!")
listadataframes[DataFramescelto]
```

```
Quale DataFrame si desidera visualizzare? 12
Eccolo qua!!!
```

[23]:
```
    Div      Date    HomeTeam   AwayTeam  FTHG  FTAG FTR  HTHG  HTAG HTR  …  \
0    I1  27/08/05  Fiorentina  Sampdoria     2     1   H     2     0   H  …
```

```
1    I1  27/08/05     Livorno      Lecce     2     1   H     1     1   D  …
2    I1  28/08/05      Ascoli      Milan     1     1   D     0     0   D  …
3    I1  28/08/05       Inter    Treviso     3     0   H     1     0   H  …
4    I1  28/08/05    Juventus     Chievo     1     0   H     1     0   H  …
..   ..       …          …        …   …   …  ..   …   …  ..  …
375  I1  14/05/06     Palermo    Messina     1     0   H     1     0   H  …
376  I1  14/05/06     Reggina   Juventus     0     2   A     0     1   A  …
377  I1  14/05/06   Sampdoria      Lecce     1     3   A     0     1   A  …
378  I1  14/05/06       Siena    Livorno     0     0   D     0     0   D  …
379  I1  14/05/06     Treviso    Udinese     2     1   H     1     1   D  …

     BbMx>2.5  BbAv>2.5  BbMx<2.5  BbAv<2.5  BbAH  BbAHh  BbMxAHH  BbAvAHH  \
0        2.30      2.03      1.75      1.66  16.0  -0.25     2.00     1.95
1        2.00      1.85      2.00      1.84  17.0  -0.50     2.00     1.96
2        1.90      1.78      2.05      1.89  15.0   1.25     2.10     2.05
3        1.65      1.55      2.40      2.23  15.0  -1.75     1.95     1.90
4        1.85      1.71      2.10      1.96  15.0  -1.50     1.90     1.87
..        …         …         …         …    …     …        …        …
375      1.67      1.61      2.35      2.16  17.0  -1.25     2.05     1.99
376      1.71      1.63      2.25      2.12  19.0   1.50     1.99     1.85
377      1.75      1.65      2.25      2.07  15.0  -0.75     1.80     1.77
378      1.80      1.70      2.16      2.03  17.0  -0.25     2.11     2.07
379      1.95      1.81      2.00      1.93  16.0   0.25     1.86     1.83

     BbMxAHA  BbAvAHA
0       1.96     1.92
1       1.95     1.89
2       1.86     1.82
3       2.01     1.97
4       2.07     1.98
..       …        …
375     1.93     1.90
376     2.07     1.98
377     2.16     2.13
378     1.87     1.83
379     2.09     2.07

[380 rows x 68 columns]
```

Infatti se si prova ad inserire un numero pari a 24 o superiore il programma darà errore durante la ricerca del DataFrame. L'errore in questione è: "IndexError: list index out of range" cioè che per l'appunto l'indice è fuori portata della lista (in questo caso si riferisce alla lista "listadataframes").

[24]:
```
DataFramescelto=int(input("Quale DataFrame si desidera visualizzare? "))
print("Eccolo qua!!!")
listadataframes[DataFramescelto]
```

```
Quale DataFrame si desidera visualizzare? 24
```

```
Eccolo qua!!!
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[24], line 3
      1 DataFramescelto=int(input("Quale DataFrame si desidera visualizzare? ")
      2 print("Eccolo qua!!!")
----> 3 listadataframes[DataFramescelto]

IndexError: list index out of range
```

Per poter continuare questa esercitazione, è necessario installare la libreria "sklearn" attraverso il comando qui sotto incollandolo sul terminale di Windows (CMD) per Python "classico" o su una cella di Notebook Jupyter se si sta programmando Python lì, questa libreria permette di gestire diverse operazioni di Machine Learning.

[ ]:
```
pip install sklearn
```

6 LO SPLITTING DATASET E LE VISUALIZZAZIONI DEI DATI IN GRAFICI

Questo codice è un vero e proprio esempio di "Dataset Splitting". Quest'ultimo incomincia con l'importazione delle librerie "numpy" e "sklearn", nel dettaglio quando si importa la libreria "sklearn" viene usata una formattazione particolare: from viene utilizzato per indicare il nome della libreria mentre con il comando import viene importata una sola parte della libreria. Nella vera parte di codice invece viene prima definito il random seed, che crea dei dati randomici che non cambiano ad ogni esecuzione del codice poichè il seme (per l'appunto il seed) è impostato a 0 altrimenti se non fosse specificato cambierebbe di volta in volta e si genererebbero così numeri casuali ogni volta diversi. Poi nella riga sotto vengono creati i veri e propri dati randomici (usando sempre np.random.) delle altezze usando una distribuzione Gaussiana (o normale) con il valore medio di 160 cm (picco della Gaussiana) e con la deviazione standard di 10 cm (cioè che permette di definire il range ad alta probabilità del valore delle altezze, quindi nel range ci sono solo i valori più comuni che sono: da 150 cm a 170 cm). Infine viene anche indicato il numero 100 che definisce il numero di dati totali che dovranno essere generati casualmente dal programma. Nella riga successiva viene indicata la formula con cui vengono calcolati i dati dei pesi, in quest'ultima viene sempre usata la distribuzione normale per generare dei dati randomici nello stesso modo di prima ma con valori diversi. È importante sottolineare il fatto che i valori sia delle altezze che dei pesi fanno parte adesso di un unico DataSet creato. A questo punto non rimane altro che iniziare a splittare (dividere in due parti non uguali) sia i valori delle altezze che quelli dei pesi, questo perchè è necessario avere un dataset di Training (70%) e uno di Test (30%). La differenza tra i dati Training e i dati di Test è che i primi vengono esclusivamente utilizzati per allenare il modello finale, invece quelli di Test svolgono il compito di testare il modello (usando per l'appunto valori diversi dalla fase di Training). Alla fine per riassumere un po' il tutto vengono stampate le dimensioni dei dati di Training e di Test.

[2]:
```
import numpy as np
from sklearn.model_selection import train_test_split # in questo caso viene␣
 ↪solo importata una parte di libreria poichè è strettamente necessaria quella␣
 ↪determinata funzione

# Creare dati casuali per altezze: variabile indipendente = input (cioè quello␣
 ↪che serve per fare delle previsioni) e sono le Feature nel DataSet
# Pesi sono la variabile dipendente = output o target (cioè ciò che bisogna␣
 ↪prevedere) del DataSet
np.random.seed(0)
altezze = np.random.normal(160, 10, 100) # la variabile "altezze" è␣
 ↪indipendente, cioè non ha una formula in cui viene denominata un'altra␣
 ↪variabile
# Formula di esempio:
pesi = 0.5 * altezze + np.random.normal(0, 5, 100) # la variabile "pesi" è␣
 ↪dipendente, cioè ha una formula in cui viene denominata un'altra variabile

# Previsione del modello: dal valore delle altezze prevedere il peso

# Suddividere il dataset in training set (70%) e test set (30%) formando due␣
 ↪DataSet
X_train, X_test, y_train, y_test = train_test_split(altezze, pesi, test_size=0.
 ↪3, random_state=42) # riprendendo la formula di prima: le X sono i valori␣
 ↪delle altezze perchè sono le Feature del DataSet, cioè l'input. Invece le Y␣
 ↪sono gli output o target del DataSet, cioè i valori dei pesi. "test_size=0.
 ↪3" vuol dire che il DataSet di Test è il 30% di quello totale mentre␣
 ↪random_state sceglie in modo randomico i valori del DataSet per il Training␣
 ↪e il Test
# Stampare le dimensioni dei training set e test set
print("Dimensioni del Training Set (altezze e pesi):", X_train.shape, y_train.
 ↪shape) # shape = dimensione dei DataSet di Training
print("Dimensioni del Test Set (altezze e pesi):", X_test.shape, y_test.shape)␣
 ↪# shape = dimensione dei DataSet di Test
```

```
Dimensioni del Training Set (altezze e pesi): (70,) (70,)
Dimensioni del Test Set (altezze e pesi): (30,) (30,)
```

Questo output particolare qui sotto non è altro che l'array di tutti i 100 valori che rappresentano altezze generate casualmente secondo la distribuzione Gaussiana nel codice di prima, con una media di 160 cm e una deviazione standard di 10 cm. Ogni valore dell'array corrisponde all'altezza di una persona esempio nel dataset. Inoltre nel codice per fare la "prova del 9" in fondo alla stampa dell'array è stato fatto stampare il numero totale di elementi dell'array tramite il comando "shape" che infatti è pari a 100, come descritto all'inizio.

CURIOSITÀ: il comando "shape" stampa sempre un valore all'interno di una parentesi tonda insieme ad una virgola posta sempre alla fine del numero prima della chiusura della parentesi finale. Questo perchè a dir la verità il comando "shape" è stato progettato per le matrici e non per gli array. Le matrici sono degli array ma che si sviluppano non solo in larghezza (asse x) ma

anche in altezza (asse y), definendo così più valori contemporaneamente. Si può e solitamente si usa questo comando anche per gli array perchè funziona correttamente, con il fatto però che quando Python non trova i dati dell'asse y lascierà uno spazio vuoto mantenendo così la virgola. Gli array si possono quindi definire unidimensionali mentre le matrici bidimensionali.

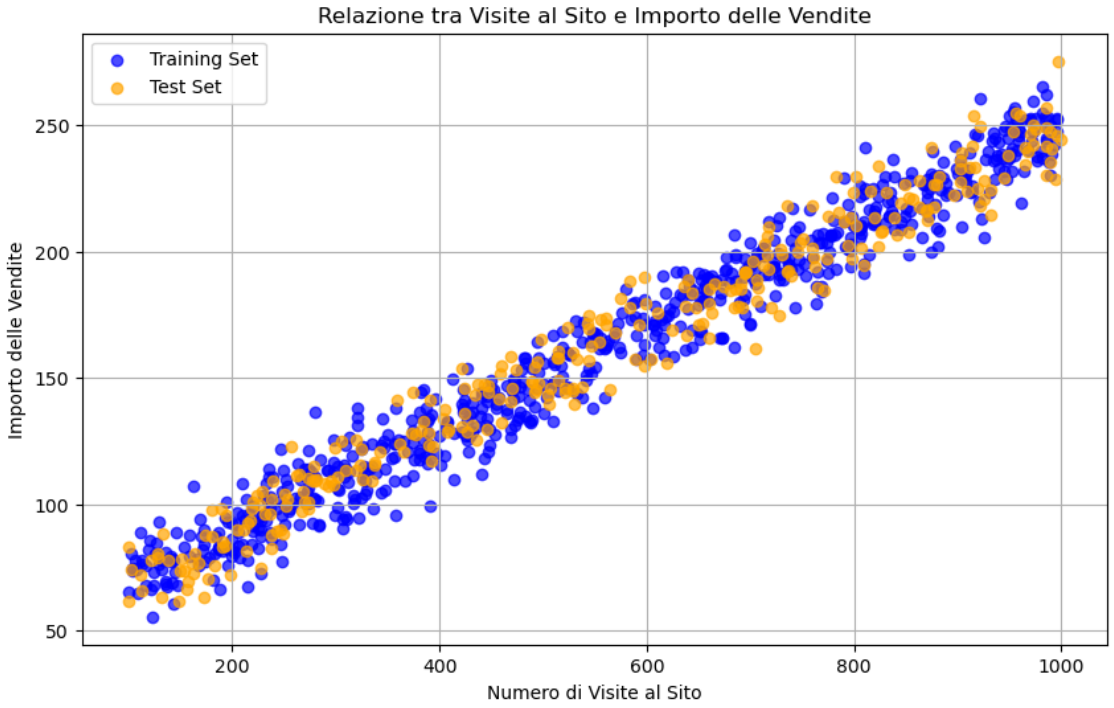
[22]:
```
# Stampare l'array delle altezze
print("Array delle altezze:")
print(altezze)
# Dimensione dell'array
dimensione = altezze.shape
print("La grandezza dell'array delle altezze è di:", dimensione)
```

```
Array delle altezze:
[177.64052346 164.00157208 169.78737984 182.40893199 178.6755799
 150.2272212  169.50088418 158.48642792 158.96781148 164.10598502
 161.44043571 174.54273507 167.61037725 161.21675016 164.43863233
 163.33674327 174.94079073 157.94841736 163.13067702 151.45904261
 134.47010184 166.53618595 168.64436199 152.5783498  182.69754624
 145.45634325 160.45758517 158.1281615  175.32779214 174.6935877
 161.54947426 163.7816252  151.12214252 140.19203532 156.52087851
 161.56348969 172.30290681 172.02379849 156.12673183 156.97697249
 149.51447035 145.79982063 142.93729809 179.50775395 154.90347818
 155.61925698 147.4720464  167.77490356 143.86102152 157.8725972
 151.04533439 163.86902498 154.89194862 148.19367816 159.71817772
 164.28331871 160.66517222 163.02471898 153.65677906 156.37258834
 153.27539552 156.40446838 151.86853718 142.73717398 161.77426142
 155.98219064 143.69801653 164.62782256 150.92701636 160.51945396
 167.29090562 161.28982911 171.39400685 147.6517418  164.02341641
 153.15189909 151.29202851 154.21150335 156.88447468 160.56165342
 148.34850159 169.00826487 164.6566244  144.63756314 174.88252194
 178.95889176 171.78779571 158.20075164 149.29247378 170.54451727
 155.96823053 172.2244507  162.08274978 169.76639036 163.56366397
 167.06573168 160.10500021 177.85870494 161.26912093 164.01989363]
La grandezza dell'array delle altezze è di: (100,)
```

Il codice sottostante è un'altro esempio di Dataset Splitting. La struttura è perlopiù identica al codice precedente riguardo la creazione dei dati e il Dataset Splitting. L'unica differenza con il codice precedente è il grafico finale sulla relazione tra le visite del sito e l'importo delle vendite. Da come si può notare, c'è una relazione di tipo lineare. Ovviamente si può subito notare come ci sia più o meno una diretta proporzionalità tra la variabile dipendente (visite al sito) e quella indipendente (importo vendite), infatti all'aumentare delle visite al sito aumenta anche l'importo delle vendite. Il fatto che non sia perfettamente lineare è dovuto al rumore generato dalla gaussiana nella formula finale (cioè quella dell'importo vendite). Il rumore è quel "disturbo" o variazione imprevista nei dati sull'importo delle vendite (in questo caso), è come un'imprevista interferenza che può rendere l'andamento dei dati meno lineare, in questo caso è proprio "np.random.normal()" a creare questa interferenza creando dei dati randomici attraverso una distribuzione Gaussiana

[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
# Creare dati casuali per le visite al sito: variabile indipendente = input␣
 ↪(cioè quello che serve per fare delle previsioni) e sono le Feature nel␣
 ↪DataSet
# L'importo delle vendite sono la variabile dipendente = output o target (cioè␣
 ↪ciò che bisogna prevedere) del DataSet
np.random.seed(0)
visite_al_sito = np.random.randint(100, 1000, 1000) # la variabile "visite al␣
 ↪sito" è indipendente, cioè non ha una formula in cui viene denominata␣
 ↪un'altra variabile. "np.random.randint(100, 1000, 1000)" vuol dire che␣
 ↪vengono creati dei valori randomici, sempre attraverso la libreria numpy. Si␣
 ↪legge: il primo parametro (100) indica il valore minimo che può assumere il␣
 ↪numero mentre il secondo parametro (1000) indica il valore massimo, infine␣
 ↪il terzo parametro (1000) indica il numero di valori da generare
importo_vendite = 50 + 0.2 * visite_al_sito + np.random.normal(0, 10, 1000) #␣
 ↪la variabile "importo delle vendite" è dipendente, cioè ha una formula in␣
 ↪cui viene denominata un'altra variabile
# 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) # riprendendo la formula di␣
 ↪prima: le X sono i valori delle visite al sito perchè sono le Feature del␣
 ↪DataSet, cioè l'input. Invece le Y sono gli output o target del DataSet,␣
 ↪cioè i valori degli importi vendite. "test_size=0.3" vuol dire che il␣
 ↪DataSet di Test è il 30% di quello totale mentre random_state sceglie in␣
 ↪modo randomico i valori del DataSet per il Training e il Test

# Previsione del modello: dal valore delle visite al sito prevedere l'importo␣
 ↪vendite

# Creazione di un grafico a dispersione
plt.figure(figsize=(10, 6)) # dimensione del grafico
plt.scatter(X_train, y_train, label='Training Set', color='blue', alpha=0.7) #␣
 ↪label è il nome della legenda, alpha è il valore della trasparenza: più è␣
 ↪vicino ad 0 come valore i pallini del grafico saranno più trasparenti
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)
```

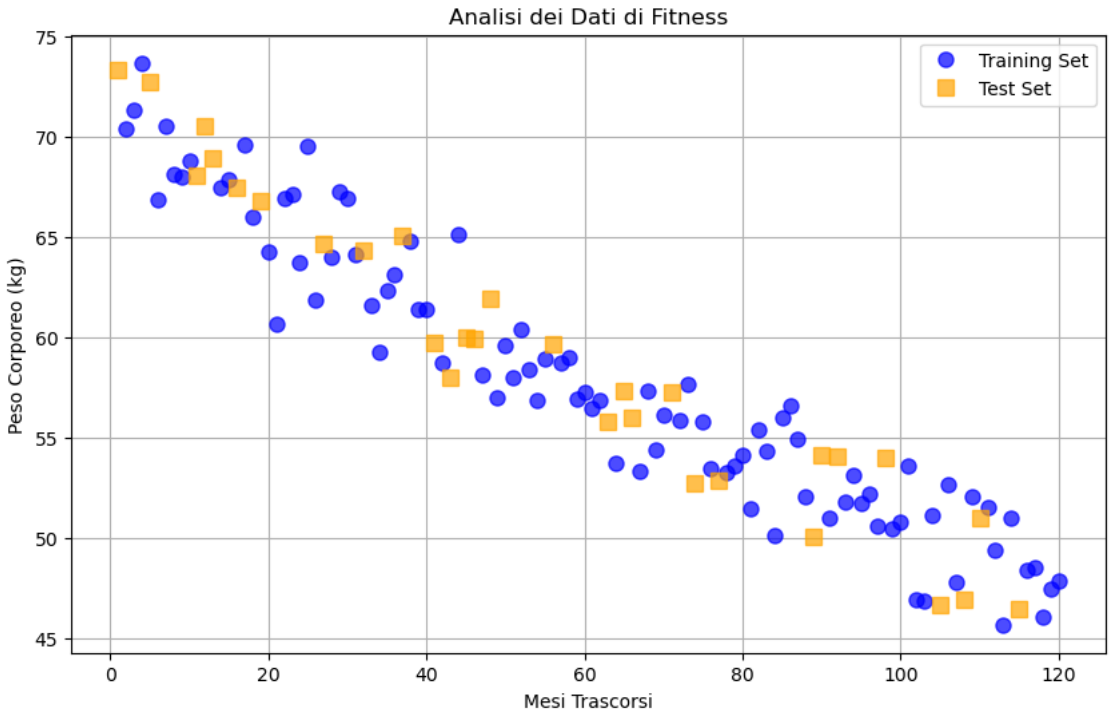



```
Dimensioni del Training Set (visite al sito e importo delle vendite): (700,)
(700,)
Dimensioni del Test Set (visite al sito e importo delle vendite): (300,) (300,)
```

Il codice sottostante è un'altro esempio di Dataset Splitting. La struttura è perlopiù identica al codice precedente riguardo la creazione dei dati e il Dataset Splitting. L'unica differenza con il codice precedente è il grafico finale sulla relazione tra i mesi trascorsi e peso corporeo. Da come si può notare, c'è una relazione sempre di tipo lineare ma ben diversa dal codice precedente. Infatti si può subito notare come ci sia più o meno una proporzionalità diretta ma negativa, quindi non inversa come si può pensare. Rispetto al codice di prima l'output viene stampato in maniera decrescente ma è pur sempre lineare, infatti non viene un'iperbole. Quindi il grafico va letto nel modo in cui all'aumentare della variabile indipendente (mesi trascorsi) diminuisce la variabile dipendente (peso corporeo). Il fatto che non sia perfettamente lineare è dovuto al rumore, generato dalla gaussiana nella formula finale (cioè quella del peso corporeo).

[9]:
```
import numpy as np #da sistemare
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

# Creazione di dati casuali per mesi trascorsi e peso corporeo
```

```
# Creare dati casuali per i mesi trascorsi: variabile indipendente = input␣
 ↪(cioè quello che serve per fare delle previsioni) e sono le Feature nel␣
 ↪DataSet
# Il peso corporeo è la variabile dipendente = output o target (cioè ciò che␣
 ↪bisogna prevedere) del DataSet
np.random.seed(0)
n=120 # è un parametro, non è una variabile ed è molto comodo in quanto se si␣
 ↪cambia quello si cambia tutto ciò "collegato" ad esso
mesi_trascorsi = np.arange(1, n+1) # la variabile "visite al sito" è␣
 ↪indipendente, cioè non ha una formula in cui viene denominata un'altra␣
 ↪variabile. "np.arange(1, n+1)" serve per generare un array di numeri interi␣
 ↪da 1 a n, inclusi, che rappresentano i singoli mesi nel periodo di␣
 ↪osservazione.
peso_corporeo = 70 - 0.2 * mesi_trascorsi + np.random.normal(0, 2, n)

# Suddivisione del dataset in training set (75%) e test set (25%)
X_train, X_test, y_train, y_test = train_test_split(mesi_trascorsi,␣
 ↪peso_corporeo, test_size=0.25, random_state=42) # riprendendo la formula di␣
 ↪prima: le X sono i valori dei mesi trascorsi perchè sono le Feature del␣
 ↪DataSet, cioè l'input. Invece le Y sono gli output o target del DataSet,␣
 ↪cioè i valori del peso corporeo. "test_size=0.25" vuol dire che il DataSet␣
 ↪di Test è il 25% di quello totale mentre random_state sceglie in modo␣
 ↪randomico i valori del DataSet per il Training e il Test

# Creazione di un grafico a linee
plt.figure(figsize=(10, 6)) #dimesioni del grafico
plt.plot(X_train, y_train, label='Training Set', marker='o', color='blue',␣
 ↪linestyle='', markersize=8,alpha=0.7) # label è il nome della legenda, alpha␣
 ↪è il valore della trasparenza: più è vicino ad 0 come valore i pallini del␣
 ↪grafico saranno più trasparenti
plt.plot(X_test, y_test, label='Test Set', marker='s', color='orange',␣
 ↪linestyle='', markersize=8,alpha=0.7)
plt.xlabel('Mesi Trascorsi')
plt.ylabel('Peso Corporeo (kg)')
plt.title('Analisi dei Dati di Fitness')
plt.legend()
plt.grid(True)
plt.show()

# Stampare le dimensioni dei training set e test set
print("Dimensioni del Training Set (mesi trascorsi e peso corporeo):", X_train.
 ↪shape, y_train.shape)
print("Dimensioni del Test Set (mesi trascorsi e peso corporeo):", X_test.
 ↪shape, y_test.shape)
```

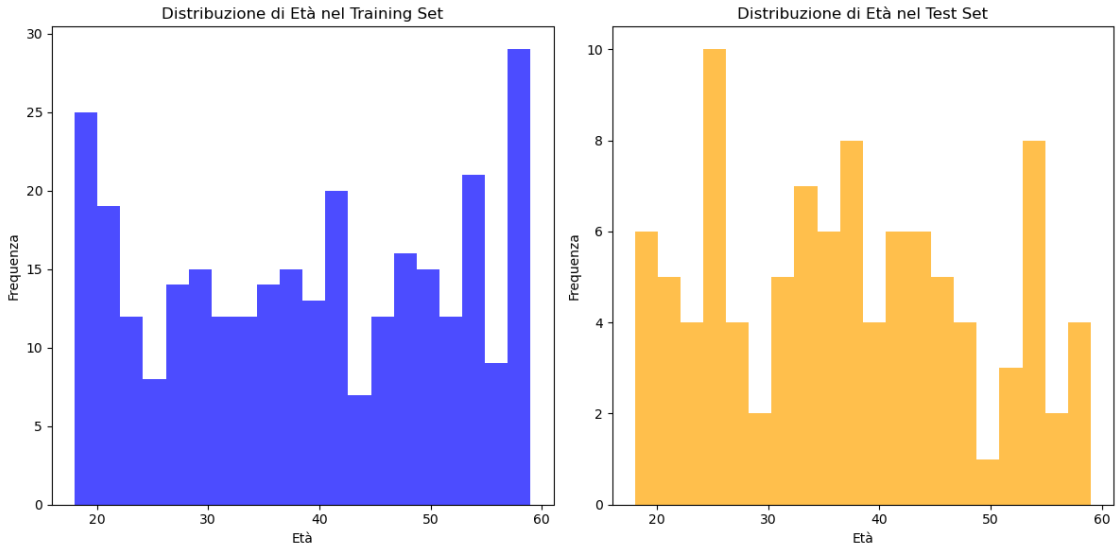



```
Dimensioni del Training Set (mesi trascorsi e peso corporeo): (90,) (90,)
Dimensioni del Test Set (mesi trascorsi e peso corporeo): (30,) (30,)
```

Nel codice sottostante vengono creati direttamente due DataSet: uno di Training e uno di Test. Entrambi i dataset vengono creati con valori randomici nella prima parte del codice. Poi nella seconda parte vengono creati i due grafici, quello blu rappresenta la distribuzione di età nel Training Set e quello arancione rappresenta la distrubuzione di età nel Test Set. Per creare grafici viene usata la libreria "matplotlib".

[10]:
```
import numpy as np
import matplotlib.pyplot as plt

# Creazione di dati casuali per età
np.random.seed(0)
eta_training_set = np.random.randint(18, 60, 300) # "np.random.randint(100,␣
 ↪1000, 1000)" vuol dire che vengono creati dei valori randomici, sempre␣
 ↪attraverso la libreria numpy. Si legge: il primo parametro (100) indica il␣
 ↪valore minimo che può assumere il numero mentre il secondo parametro (1000)␣
 ↪indica il valore massimo, infine il terzo parametro (1000) indica il numero␣
 ↪di valori da generare
eta_test_set = np.random.randint(18, 60, 100)
# Confronto delle distribuzioni di età
# Primo grafico
plt.figure(figsize=(12, 6)) # dimesioni del grafico
```

```
plt.subplot(1, 2, 1) # permette di creare all'interno di una figura più plot.␣
 ↪Il primo valore indica il numero di righe, il secondo indica il numero di␣
 ↪colonne ed invece il terzo indica in quale colonna stampare
plt.hist(eta_training_set, bins=20, color='blue', alpha=0.7) # crea un␣
 ↪istogramma, bins vuole indicare la barre verticali cioè i cosidetti "bins"
plt.title('Distribuzione di Età nel Training Set')
plt.xlabel('Età')
plt.ylabel('Frequenza')
# Secondo grafico
plt.subplot(1, 2, 2)
plt.hist(eta_test_set, bins=20, color='orange', alpha=0.7)
plt.title('Distribuzione di Età nel Test Set')
plt.xlabel('Età')
plt.ylabel('Frequenza')
plt.tight_layout()
plt.show()
```



7 LO SPLITTING DATASET CON LE CLASSI

Questo codice è molto simile a quelli mostrati sopra, infatti vengono inserite le librerie numpy e sklearn e poi vengono generati in maniera randomica i valori di x e y. Questo codice introduce il concetto di classi, infatti l'array y ha presente ben due classi. Le classi sono le categorie a cui appartengono i dati come "cane" o "gatto"; in questo codice le classi sono A e B. Il concetto di "split stratificato" è come dividere i dati in modo da mantenere la stessa proporzione di categorie in entrambe le parti. Questo è importante quando ci sono più di una categoria come in questo caso, ci si può assicurare che il modello le impari da entrambe le classi, quindi sia dalla classe A che quella B. Uno split normale non tiene conto di questo, quindi potrebbe dare al modello un'idea sbagliata delle categorie. Quindi, lo split stratificato aiuta il modello a imparare meglio perchè

assicura che ciascuna categoria sia rappresentata in modo equo sia nel Training che in quello di Test. Ciò significa che il modello avrà l'opportunità di imparare da esempi di entrambe le categorie, migliorando così la sua capacità di generalizzazione e di prendere decisioni accurate su nuovi dati. Infatti nei dati stampati alla fine i valori sono simili perchè lo split stratificato ha "equiparato" all'incirca le proporzioni tra il Test e il Training

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

np.random.seed(3)
# Supponiamo di avere un dataset con feature (input) X e target (output) y
X = np.random.rand(100, 2) # crea dati del dataset (100 campioni, 2 feature)
y = np.random.choice(['A', 'B'], size=100) # etichette (stringhe A e B) di␣
 ↪classi casuali (categorie), size indica la grandezza di y (cioè ha 100␣
 ↪valori)
# Per mostrare che i valori siano casuali, essi vengono stampati
print("I valori di x sono:")
print(X)
print("I valori di y sono:")
print(y)

# Calcolare le percentuali delle classi nell'intero dataset originale
percentuale_classe_A = sum(y == 'A') / len(y)*100 # nel primo caso la␣
 ↪percentuale viene calcolata dividendo il numero dei valori di A per la␣
 ↪lunghezza di y
percentuale_classe_B = 100 - percentuale_classe_A # nel secondo per calcolare␣
 ↪la percentuale mancante si sottrae a 100 il valore della percentuale classe A

# Eseguire uno split stratificato con una proporzione specificata
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42) # "test_size=0.3" vuol dire che il DataSet di Test è il 30%␣
 ↪di quello totale mentre random_state sceglie in modo randomico i valori del␣
 ↪DataSet per il Training e il Test

# Calcolare le percentuali delle classi nel training set e nel test set
percentuale_classe_A_train = sum(y_train == 'A') / len(y_train)*100
percentuale_classe_B_train = 100 - percentuale_classe_A_train

percentuale_classe_A_test = sum(y_test == 'A') / len(y_test)*100
percentuale_classe_B_test = 100 - percentuale_classe_A_test

# Stampare delle proporzioni
print("Percentuale Classe A nel Data Set:", percentuale_classe_A)
print("Percentuale Classe B nel Data Set:", percentuale_classe_B)
print("Percentuale Classe A nel Training Set:", percentuale_classe_A_train)
print("Percentuale Classe B nel Training Set:", percentuale_classe_B_train)
print("Percentuale Classe A nel Test Set:", percentuale_classe_A_test)
```

```
print("Percentuale Classe B nel Test Set:", percentuale_classe_B_test)
```

```
I valori di x sono:
[[0.5507979  0.70814782]
 [0.29090474 0.51082761]
 [0.89294695 0.89629309]
 [0.12558531 0.20724288]
 [0.0514672  0.44080984]
 [0.02987621 0.45683322]
 [0.64914405 0.27848728]
 [0.6762549  0.59086282]
 [0.02398188 0.55885409]
 [0.25925245 0.4151012 ]
 [0.28352508 0.69313792]
 [0.44045372 0.15686774]
 [0.54464902 0.78031476]
 [0.30636353 0.22195788]
 [0.38797126 0.93638365]
 [0.97599542 0.67238368]
 [0.90283411 0.84575087]
 [0.37799404 0.09221701]
 [0.6534109  0.55784076]
 [0.36156476 0.2250545 ]
 [0.40651992 0.46894025]
 [0.26923558 0.29179277]
 [0.4576864  0.86053391]
 [0.5862529  0.28348786]
 [0.27797751 0.45462208]
 [0.20541034 0.20137871]
 [0.51403506 0.08722937]
 [0.48358553 0.36217621]
 [0.70768662 0.74674622]
 [0.69109292 0.68918041]
 [0.37360012 0.6681348 ]
 [0.33984866 0.57279387]
 [0.32580716 0.44514505]
 [0.06152893 0.24267542]
 [0.97160261 0.2305842 ]
 [0.69147751 0.65047686]
 [0.72393914 0.47508861]
 [0.59666377 0.06696942]
 [0.07256214 0.19897603]
 [0.151861   0.10010434]
 [0.12929386 0.55327773]
 [0.18781482 0.95210124]
 [0.68161178 0.54101967]
 [0.7071806  0.26388667]
 [0.92672568 0.83919306]
```

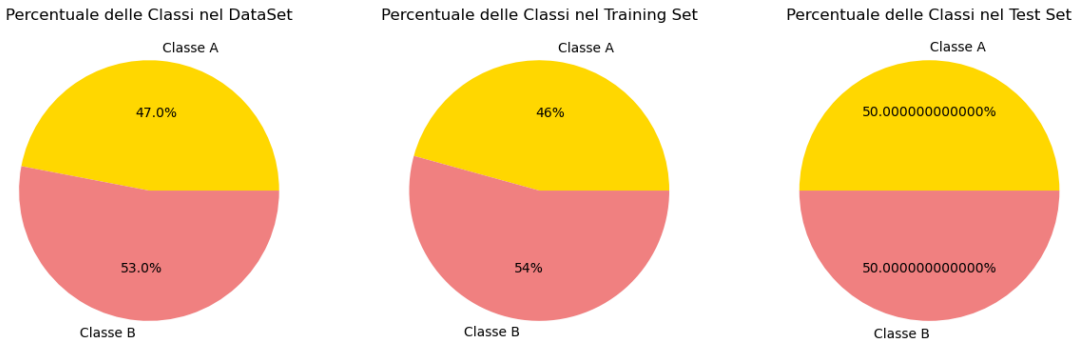
```
[0.7263195  0.48023996]
[0.84210319 0.74475232]
[0.66032591 0.91397527]
[0.63366556 0.36594058]
[0.55284457 0.19638058]
[0.1920723  0.72566962]
[0.7849367  0.97209836]
[0.85097142 0.54359433]
[0.08979087 0.48887324]
[0.92793635 0.7876182 ]
[0.48509423 0.45527936]
[0.21798577 0.17721338]
[0.07362367 0.89239319]
[0.64017662 0.14333232]
[0.41412692 0.04910892]
[0.20937335 0.73070812]
[0.65112277 0.4789783 ]
[0.27478051 0.65222313]
[0.95644951 0.43552056]
[0.07013251 0.05773149]
[0.08287102 0.95970719]
[0.54076084 0.83746243]
[0.17003354 0.26034507]
[0.69197751 0.89557033]
[0.34068848 0.0646732 ]
[0.86411967 0.29087245]
[0.74108241 0.15803365]
[0.69496344 0.84141962]
[0.72715208 0.35910752]
[0.72668975 0.13946712]
[0.31381912 0.41958276]
[0.87721204 0.15374021]
[0.88012479 0.79896432]
[0.9716243  0.36770298]
[0.20493977 0.24057032]
[0.8278628  0.96522815]
[0.69881    0.48249704]
[0.28704976 0.83368788]
[0.87217951 0.09213159]
[0.21594947 0.83176109]
[0.8483039  0.314653  ]
[0.2792946  0.43081502]
[0.5394465  0.09556682]
[0.83691214 0.53473487]
[0.77496782 0.23083627]
[0.96529335 0.75102731]
[0.34309386 0.94852765]
[0.70051178 0.84056109]
```

```
 [0.04549731 0.05564154]
 [0.74273727 0.30468643]
 [0.51678437 0.15626242]
 [0.97795241 0.50275105]
 [0.82900108 0.0740378 ]
 [0.47891545 0.06227948]
 [0.88424143 0.44581018]]
I valori di y sono:
['B' 'B' 'A' 'B' 'B' 'B' 'A' 'B' 'B' 'B' 'B' 'B' 'A' 'A' 'B' 'A' 'A' 'B'
 'A' 'B' 'B' 'B' 'B' 'A' 'A' 'B' 'A' 'A' 'B' 'B' 'A' 'B' 'B' 'A' 'A' 'A'
 'B' 'A' 'A' 'B' 'A' 'B' 'B' 'B' 'A' 'A' 'B' 'B' 'A' 'A' 'B' 'A' 'B' 'B'
 'B' 'A' 'A' 'B' 'B' 'B' 'A' 'A' 'B' 'A' 'A' 'A' 'A' 'A' 'B' 'B' 'A' 'A'
 'A' 'B' 'A' 'A' 'A' 'B' 'A' 'B' 'B' 'A' 'A' 'B' 'B' 'B' 'B' 'B' 'B' 'B'
 'A' 'B' 'B' 'B' 'A' 'B' 'A' 'A' 'A' 'A']
Percentuale Classe A nel Data Set: 47.0
Percentuale Classe B nel Data Set: 53.0
Percentuale Classe A nel Training Set: 45.714285714285715
Percentuale Classe B nel Training Set: 54.285714285714285
Percentuale Classe A nel Test Set: 50.0
Percentuale Classe B nel Test Set: 50.0
```

Nel grafico sottostante vengono stampati la percentuale di classi del DataSet, la percentuale di classi del Training Set, la percentuale di classi del Test Set. Anche qua infatti si può notare un equiparazione dovuta allo split stratificato:

[20]:
```
import matplotlib.pyplot as plt
# Etichette delle classi
labels=["Classe A", "Classe B"]
# Colori delle fette del grafico
colors = ['gold', 'lightcoral']
# Crea un grafico a torta con etichette
plt.figure(figsize=(15,10)) # dimesioni del grafico
plt.subplot(1,3,1) # permette di creare all'interno di una figura più plot. Il␣
 ↪primo valore indica il numero di righe, il secondo indica il numero di␣
 ↪colonne ed invece il terzo indica in quale colonna stampare
plt.pie([percentuale_classe_A, percentuale_classe_B], labels=labels,␣
 ↪colors=colors, autopct='%1.1f%%') # si usa pie perchè in questo caso è un␣
 ↪grafico a torta, autopct è l'etichetta con il valore: più numeri si mettono␣
 ↪dopo il punto e più valori decimali vengono mostrati
plt.title('Percentuale delle Classi nel DataSet')

# Crea un grafico a torta con etichette
plt.subplot(1,3,2)
plt.pie([percentuale_classe_A_train, percentuale_classe_B_train],␣
 ↪labels=labels, colors=colors, autopct='%1.0f%%')#si usa pie perchè in questo␣
 ↪caso è un grafico a torta, autopct è l'etichetta con il valore: più numeri␣
 ↪si mettono dopo il punto e più valori decimali vengono mostrati
plt.title('Percentuale delle Classi nel Training Set')
```

```
# Crea un grafico a torta con etichette
plt.subplot(1,3,3)
plt.pie([percentuale_classe_A_test, percentuale_classe_B_test], labels=labels,␣
 ↪colors=colors, autopct='%1.12f%%')#si usa pie perchè in questo caso è un␣
 ↪grafico a torta, autopct è l'etichetta con il valore: più numeri si mettono␣
 ↪dopo il punto e più valori decimali vengono mostrati
plt.title('Percentuale delle Classi nel Test Set')
plt.show()
```



Il codice sottostante genera un grande DataSet di dati contenente 24 milioni di numeri casuali compresi tra 0 e 100. Dopodiché, viene estratto un campione casuale di dimensione circa il 30% del totale. Successivamente, calcola la media e la deviazione standard di questo campione casuale. La media è il valore medio dei numeri nel campione, mentre la deviazione standard indica quanto i numeri nel campione si discostano dalla media. Infine, calcola la media e la deviazione standard per l'intero dataset. La media è il valore medio di tutti i numeri nel dataset, mentre la deviazione standard indica quanto i numeri nel dataset si discostano dalla media. Alla fine il codice stampa i seguenti risultati: la media e la deviazione standard del campione casuale, e la media e la deviazione standard del dataset completo.

[18]:
```
import random
import numpy as np

dataset=[]
# Creazione di un dataset di 24 milioni elementi (ad esempio, dati casuali)
popolazione=24000000
for i in range(popolazione):
    dataset.append(random.randint(0, 100000)) # il Dataset prima del ciclo for␣
 ↪è vuoto ma viene definito il numero della popolazione, dopo bisogna però␣
 ↪aggiungere ogni singolo elemento di 24000000 alla lista creandolo in maniera␣
 ↪randomica e per far questo utiizza un ciclo for

# Estrazione di un campione casuale e non: cioè una parte del Dataset creando␣
 ↪così un sub DataSet
```

```
campione = int(round(0.3 * popolazione)) # per creare un sub DataSet in maniera␣
 ↪del tutto causale, viene estratta casualmente una parte del Dataset␣
 ↪utilizzando questa formula. Lo 0.3 = 3%. Round = approssima all'intero e poi␣
 ↪Int trasforma il risultato in intero
campione_casuale = random.sample(dataset, campione) # il comando random.
 ↪sample() campiona (estrae) il DataSet, si usa "sample" quando bisogna␣
 ↪estrarre casualmente un valore. "dataset" rappresenta la sequenza dalla␣
 ↪quale estrarre un campione casuale invece "campione" rappresenta il numero␣
 ↪di elementi da estrarre casualmente dal dataset (il 30% perchè è collegato␣
 ↪alla formula di sopra).

# Calcolo della media e della deviazione standard del campione
media_campione = np.mean(campione_casuale) # il comando np.mean() calcola la␣
 ↪media del campione casuale
deviazione_standard_campione = np.std(campione_casuale) # calcola la deviazione␣
 ↪standard (std = standard deviation). La deviazione standard è una misura di␣
 ↪dispersione o variabilità dei dati in un insieme. In sostanza, dice quanto i␣
 ↪dati sono distanziati dalla media.

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

print(f"Media del campione casuale: {media_campione: .2f}") # il ".2f" indica␣
 ↪quanti valori dopo la virgola mostrare, più è alto più vengono mostrati
print(f"Deviazione standard del campione casuale: {deviazione_standard_campione:
 ↪ .2f}")
print(f"Media del dataset completo: {media_dataset: .10f}")
print(f"Deviazione standard del dataset completo: {deviazione_standard_dataset:␣
 ↪.10f}")
```

```
Media del campione casuale:  49974.55
Deviazione standard del campione casuale:  28867.11
Media del dataset completo:  49990.8340742917
Deviazione standard del dataset completo:  28866.2865067231
```

Questo codice utilizza la libreria pandas per creare un DataFrame contenente una colonna chiamata "ColonnaAB" con una distribuzione specificata tra le categorie 'A' e 'B'. Per prima cosa, il codice imposta un seed per la riproducibilità dei risultati. Poi, definisce il numero totale di elementi nel DataFrame (100.000) e la percentuale di elementi della categoria 'A' (0.7). Successivamente, utilizza la funzione np.random.choice per generare una colonna di lunghezza num_elementi con distribuzione desiderata tra le categorie 'A' e 'B', specificata attraverso il parametro p, dove p è una lista di probabilità associate a ciascuna categoria. Infine, crea il DataFrame utilizzando la libreria pandas, assegnando alla colonna il nome 'ColonnaAB' e la visualizza.

[34]:
```
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 = 100000
# 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
```

[34]:

	ColonnaAB
0	A
1	B
2	B
3	A
4	A
…	…
99995	B
99996	B
99997	A
99998	A
99999	A

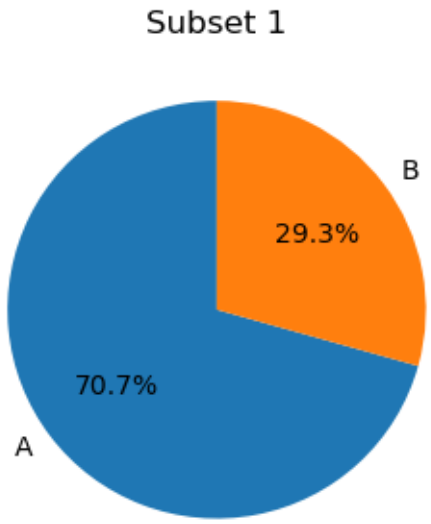
[100000 rows x 1 columns]

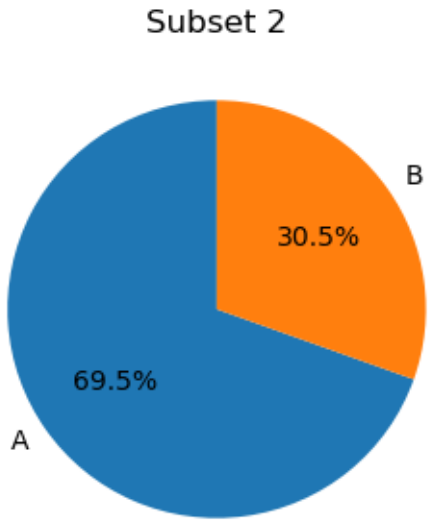
Questo codice crea tre sottoinsiemi (subset) del DataFrame originale, ciascuno con dimensioni simili. Per prima cosa, crea subset1 e lo inizializza estraendo casualmente un terzo dei dati dal DataFrame originale utilizzando il metodo sample(frac=1/3). Successivamente, rimuove gli indici dei dati presenti in subset1 dal DataFrame originale utilizzando il metodo drop(). Poi, crea subset2 e ripete il processo, estraendo casualmente la metà dei dati rimasti nel DataFrame originale e rimuovendoli dal DataFrame. Infine, subset3 è costituito dai dati rimanenti nel DataFrame originale dopo aver creato subset1 e subset2.

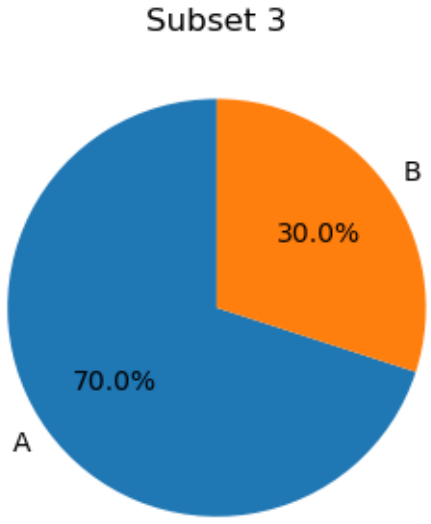
[37]:
```
# Creare tre subset di dimensioni simili
subset1 = df.sample(frac=1/3)
df = df.drop(subset1.index)

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

subset3 = df  # L'ultimo subset con il rimanente
```

Qua sotto vengono stampati i grafici esplicativi:

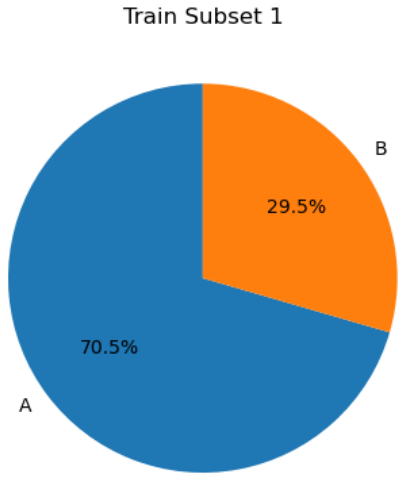
[38]:
```
# Calcolare le percentuali di "A" e "B" per ogni subset
percentuali_subset1 = subset1['ColonnaAB'].value_counts(normalize=True)
percentuali_subset2 = subset2['ColonnaAB'].value_counts(normalize=True)
percentuali_subset3 = subset3['ColonnaAB'].value_counts(normalize=True)

# 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()
```

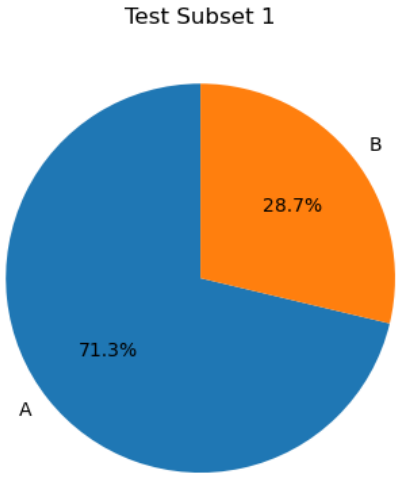

Subset 1
B
29.3%
70.7%
A

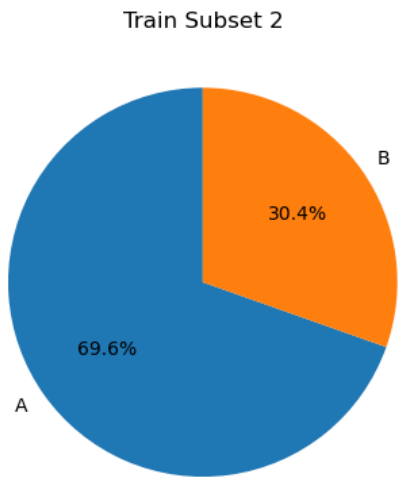

Subset 2
B
30.5%
69.5%
A

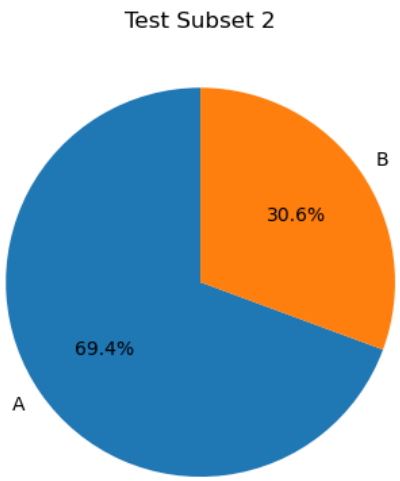

Subset 3
B
30.0%
70.0%
A

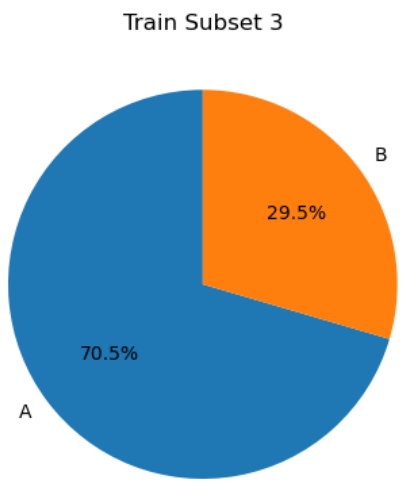
[39]:
```
# Dividere ciascun subset in training set e test set
train_subset1, test_subset1 = train_test_split(subset1, test_size=0.2,␣
 ↪random_state=42)
train_subset2, test_subset2 = train_test_split(subset2, test_size=0.2,␣
 ↪random_state=42)
train_subset3, test_subset3 = train_test_split(subset3, 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)

# 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()
```

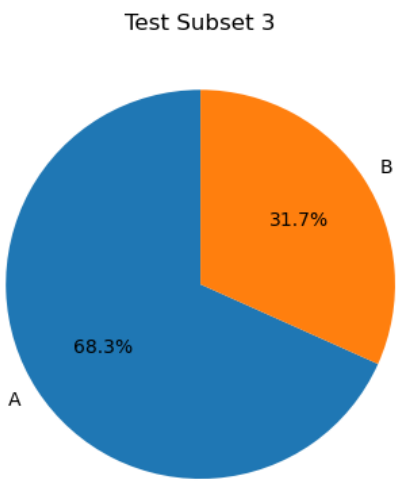

Train Subset 1
B
29.5%
70.5%
A


Test Subset 1
B
28.7%
71.3%
A


Train Subset 2
B
30.4%
69.6%
A

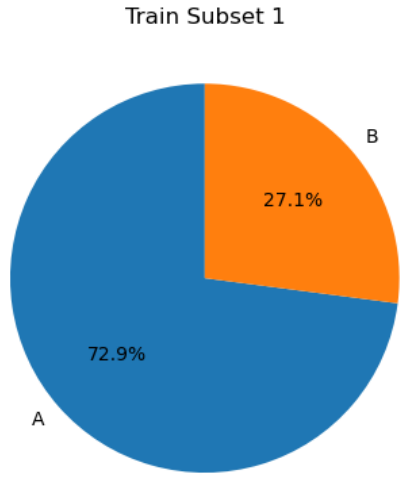

Test Subset 2
B
30.6%
69.4%
A

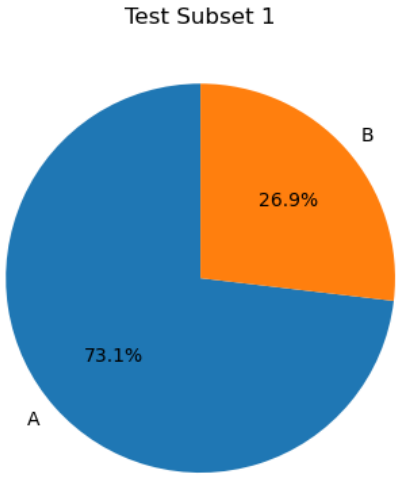

Train Subset 3
B
29.5%
70.5%
A

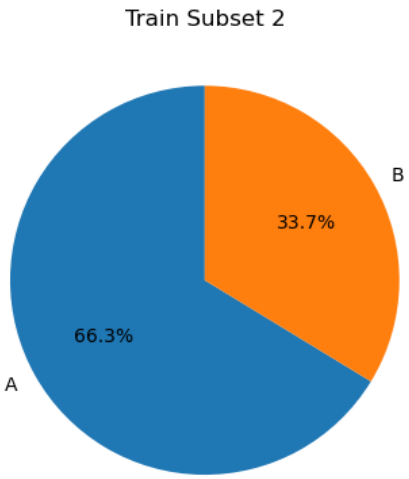

Test Subset 3
B
31.7%
68.3%
A

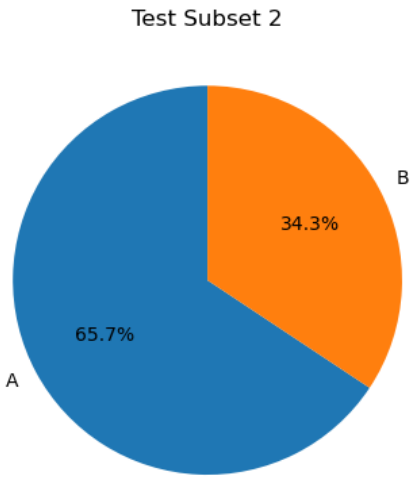
[40]:
```
np.random.seed(41)

# Creare il DataFrame originale
num_elementi = 1000
percentuale_A = 0.7
colonna = np.random.choice(['A', 'B'], size=num_elementi, p=[percentuale_A, 1 -␣
 ↪percentuale_A])
df = pd.DataFrame({'ColonnaAB': colonna})

# 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 con il rimanente

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

# 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')
```

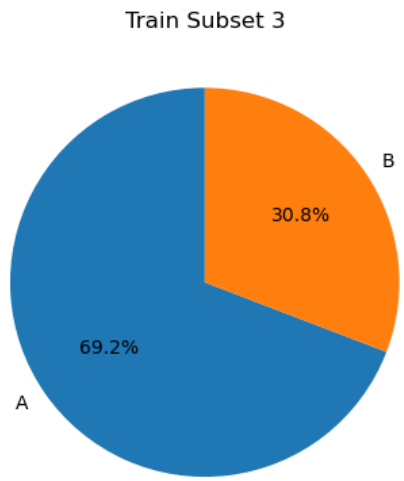
```
# Terza riga di torte (Subset 3)
draw_pie(axs[2, 0], train_subset3['ColonnaAB'].value_counts(normalize=True),␣
 ↪'Train Subset 3')
draw_pie(axs[2, 1], test_subset3['ColonnaAB'].value_counts(normalize=True),␣
 ↪'Test Subset 3')

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

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

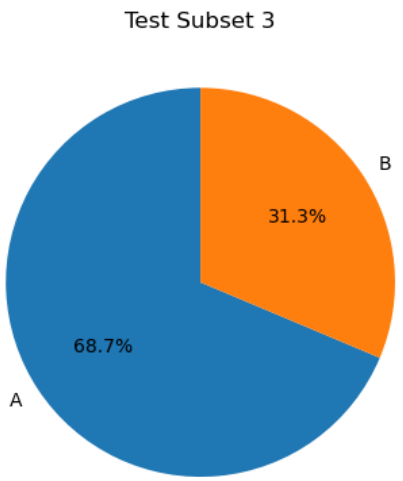

Train Subset 1
B
27.1%
72.9%
A


Test Subset 1
B
26.9%
73.1%
A


Train Subset 2
B
33.7%
66.3%
A

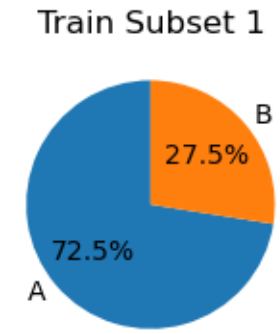

Test Subset 2
B
34.3%
65.7%
A

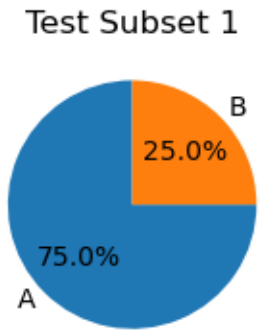

Train Subset 3
B
30.8%
69.2%
A

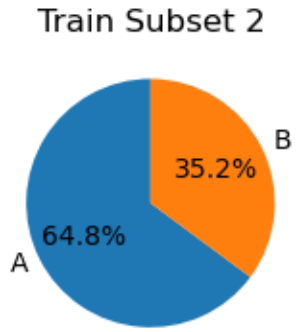

Test Subset 3
B
31.3%
68.7%
A

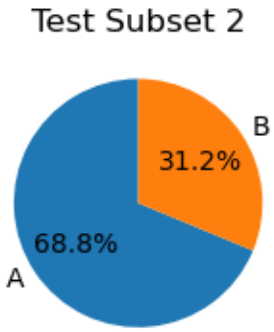
[41]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns

# Impostare il seed per la riproducibilità
np.random.seed(41)

# Creare il DataFrame originale
num_elementi = 1000
percentuale_A = 0.7
colonna = np.random.choice(['A', 'B'], size=num_elementi, p=[percentuale_A, 1 -␣
 ↪percentuale_A])
df = pd.DataFrame({'ColonnaAB': colonna})

# Numero di subset desiderato
num_subset = 5

# Creare i subset di dimensioni simili
subset_list = []
for i in range(num_subset):
    subset = df.sample(frac=1/num_subset)
    df = df.drop(subset.index)
    subset_list.append(subset)

# Creare il grafico con 2 torte per ognuno dei N subset
fig, axs = plt.subplots(num_subset, 2, figsize=(10, 2*num_subset))

# Iterare attraverso i subset e disegnare le torte
for i, subset in enumerate(subset_list):
    # Dividere ciascun subset in training set e test set
    train_set, test_set = train_test_split(subset, test_size=0.2,␣
 ↪random_state=42) # posso aggiungere stratify=subset['ColonnaAB']

    # Prima colonna: Training Set
    draw_pie(axs[i, 0], train_set['ColonnaAB'].value_counts(normalize=True),␣
 ↪f'Train Subset {i + 1}')

    # Seconda colonna: Test Set
    draw_pie(axs[i, 1], test_set['ColonnaAB'].value_counts(normalize=True),␣
 ↪f'Test Subset {i + 1}')

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

# Mostrare il grafico
```

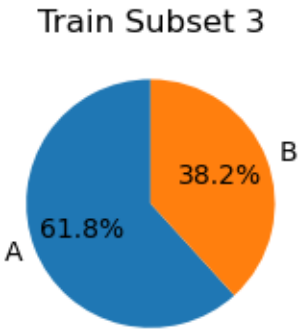
```
plt.show()
```

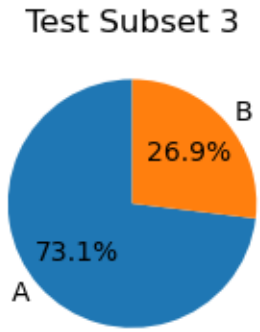

Train Subset 1
B
27.5%
72.5%
A

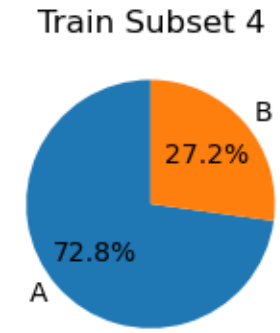

Test Subset 1
B
25.0%
75.0%
A

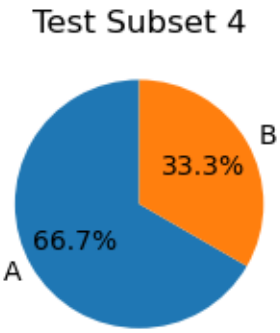

Train Subset 2
B
35.2%
64.8%
A

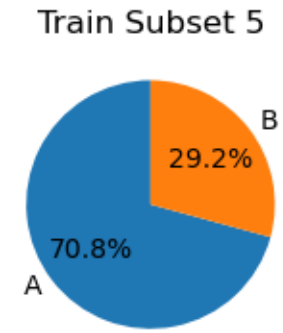

Test Subset 2
B
31.2%
68.8%
A

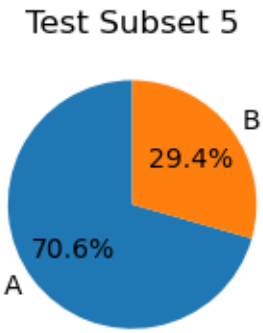

Train Subset 3
B
38.2%
61.8%
A


Test Subset 3
B
26.9%
73.1%
A


Train Subset 4
B
27.2%
72.8%
A


Test Subset 4
B
33.3%
66.7%
A


Train Subset 5
B
29.2%
70.8%
A


Test Subset 5
B
29.4%
70.6%
A

8 OUTLIERS

9 GLI OUTLIER CHE COSA SONO?

Gli outlier sono valori anomali che si distaccano notevolmente dalla normale distribuzione dei dati. Questi valori "fuori scala" possono influenzare drasticamente le previsioni di un modello statistico e vengono gestiti proprio per questo.

10 LA DEVIAZIONE STANDARD

La funzione che viene creata nel codice calcola la deviazione standard di una lista di numeri utilizzando la formula matematica corrispondente. Innanzitutto, la funzione calcola la media dei numeri nella lista. Successivamente, calcola la somma dei quadrati delle differenze tra ciascun numero nella lista e la media calcolata in precedenza. Infine, divide questa somma per il numero totale di elementi nella lista e ne calcola la radice quadrata per ottenere la deviazione standard. La deviazione standard è una misura della dispersione dei dati intorno alla media: maggiore è la deviazione standard, maggiore è la dispersione dei dati. Alla fine in fondo al codice, viene mostrato un esempio di utilizzo in cui viene creata una lista di numeri [1, 2, 3, 4, 5] che la funzione "calcola_deviazione_standard" calcolerà la sua deviazione standard, e il risultato viene stampato a schermo.

Formula deviazione standard: $\sigma = \sqrt{(\Sigma(xi - x)^2/n)}$

$\sqrt{}$ = radice quadrata

Σ = sommatoria di tutti gli elementi dentro la parentesi quadra

xi = sono i singoli valori dei dati

$\bar{x}$ = è la media dei dati

n = è il numero totale di dati

[1]:
```python
def calcola_deviazione_standard(lista):
    n = len(lista)

    # Calcola la media
    media = sum(lista) / n

    # Calcola la somma dei quadrati delle differenze dalla media
    somma_quadrati_diff = sum((x - media) ** 2 for x in lista)

    # Calcola la deviazione standard
    deviazione_standard = (somma_quadrati_diff / n) ** 0.5

    return deviazione_standard
```

```
# Esempio di utilizzo
numero_lista = [1, 2, 3, 4, 5]
deviazione_standard = calcola_deviazione_standard(numero_lista)

# Stampa il risultato
print(f"La deviazione standard della lista è: {deviazione_standard}")
```

```
La deviazione standard della lista è: 1.4142135623730951
```

Questo codice utilizza la libreria pandas per creare un DataFrame di esempio chiamato df contenente una colonna chiamata 'Valori' con una serie di valori. Successivamente, calcola la media e la deviazione standard dei valori nella colonna 'Valori' utilizzando i metodi mean() e std() rispettivamente. Infine, stampa la deviazione standard calcolata.

[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import pandas as pd
import matplotlib.pyplot as plt

# Crea un DataFrame di esempio
data = {'Valori': [1, 2, 3, 4, 5, 10, 15, 20, 25, 300, 1000, 100000000,␣
 ↪-50000000, -50]}
df = pd.DataFrame(data)
# Lista con outliers da entrambi i lati

# Calcola la media e la deviazione standard
mean_value = df['Valori'].mean()
std_dev = df['Valori'].std()
std_dev
```

[2]:
```
30786384.39895254
```

Questo codice identifica gli outliers nel DataFrame originale considerando i valori che si trovano oltre 3 deviazioni standard dalla media. Per prima cosa, crea un DataFrame chiamato "outliers" che contiene solo le righe del DataFrame originale (df) in cui il valore della colonna "Valori" è superiore a 3 deviazioni standard sopra la media (mean_value + 3 * std_dev) o inferiore a 3 deviazioni standard sotto la media (mean_value - 3 * std_dev). Infine, visualizza il DataFrame "outliers" che contiene solo i valori considerati outliers. Questo metodo è utile per l'appunto per identificare e rimuovere dati anomali o non rappresentativi che potrebbero influenzare negativamente l'analisi o i modelli di machine learning.

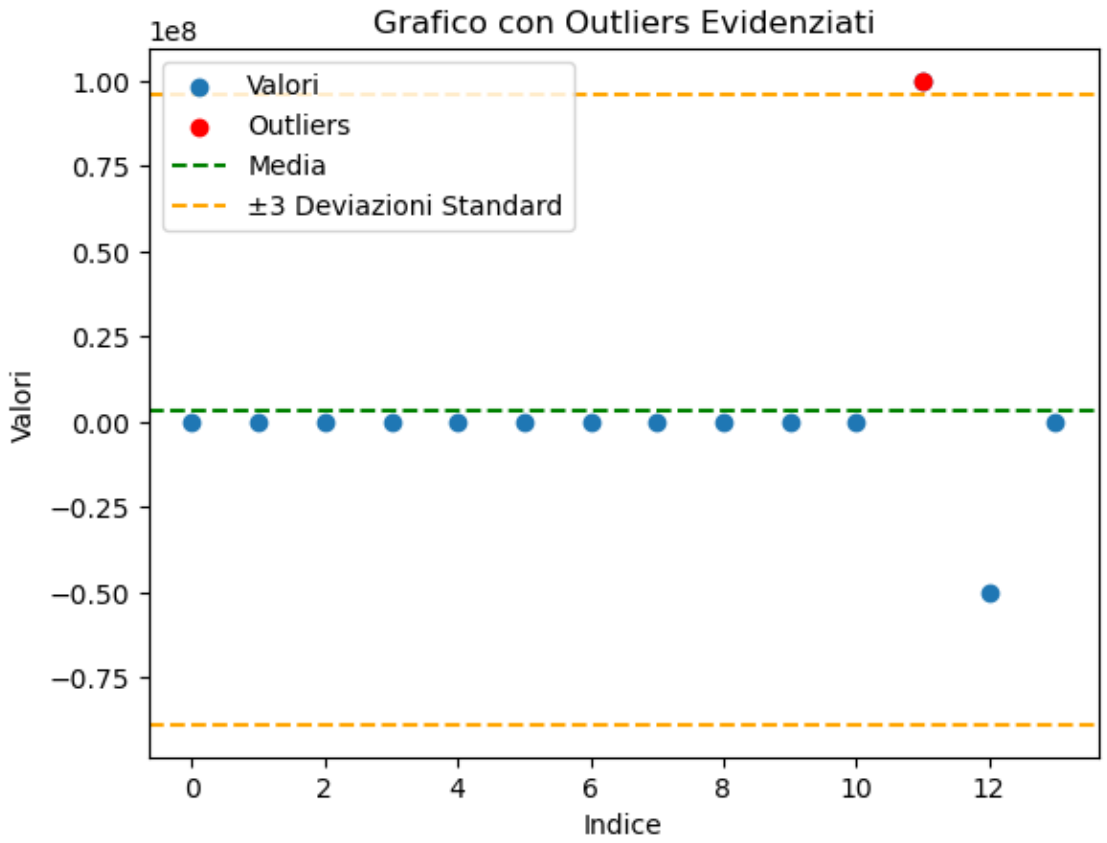
[3]:
```
#Identifica gli outliers consiederando +3 sigma dalla media
outliers=df[(df["Valori"]>mean_value+3*std_dev) |␣
 ↪(df["Valori"]<mean_value-3*std_dev)]
outliers
```

[3]:

	Valori
11	100000000

Questo codice crea un grafico a dispersione (scatter plot) con i valori presenti nel DataFrame df. Successivamente, evidenzia gli outliers nel grafico utilizzando un colore rosso per i punti corrispondenti agli outliers, i quali sono presumibilmente individuati in un DataFrame chiamato outliers. Inoltre, aggiunge al grafico la linea della media dei valori (mean_value) e le linee corrispondenti a più o meno 3 deviazioni standard dalla media, utilizzando linee tratteggiate verdi e arancioni rispettivamente. Infine, vengono aggiunte etichette agli assi, un titolo al grafico e una legenda che spiega i diversi elementi presenti nel grafico. Il grafico risultante mostra la distribuzione dei valori nel DataFrame df, evidenziando gli outliers e indicando la media e la deviazione standard dei valori.

[4]:
```
# Crea un grafico a dispersione
plt.scatter(df.index, df['Valori'], label='Valori')

# Evidenzia gli outliers nel grafico con un colore diverso
plt.scatter(outliers.index, outliers['Valori'], color='red', label='Outliers')

# Aggiungi la media e la deviazione standard al grafico
plt.axhline(y=mean_value, color='green', linestyle='--', label='Media')
plt.axhline(y=mean_value + 3 * std_dev, color='orange', linestyle='--',␣
 ↪label='±3 Deviazioni Standard')
plt.axhline(y=mean_value - 3 * std_dev, color='orange', linestyle='--')

# Aggiungi etichette e legenda al grafico
plt.xlabel('Indice')
plt.ylabel('Valori')
plt.title('Grafico con Outliers Evidenziati')
plt.legend()

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



[5]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Crea un DataFrame di esempio con 4 features
data = {'Feature1': [1, 200, 3, 4, 50000, 10, 15, 20, 2500000, 300000000,␣
 ↪100000000],
        'Feature2': [2, 4, 6, 8, 10, 20, 30, 40, 500, 60, 200],
        'Feature3': [5, 10, 15, 20000, 25, 50, 75, 100, 125, 150, 500000],
        'Feature4': [1, -200000, 3, 4000000000, 5, 10, 15, 20, 200, 30, 10000]}

df = pd.DataFrame(data)

# Definisci il numero minimo di features che devono superare la soglia per␣
 ↪considerare un dato un outlier
min_features_threshold = 1
k=3 #intervallo di confidenza

# Lista per salvare gli indici degli outliers
outlier_indices = []
```

```
# Itera su ogni feature
for feature in df.columns:
    mean_value = df[feature].mean()
    std_dev = df[feature].std()
    # Identifica gli outliers per ciascuna feature
    df['Outlier_' + feature] = (df[feature] > mean_value + k * std_dev) |␣
 ↪(df[feature] < mean_value - k * std_dev)
df
```

[5]:

	Feature1	Feature2	Feature3	Feature4	Outlier_Feature1	\
0	1	2	5	1	False	
1	200	4	10	-200000	False	
2	3	6	15	3	False	
3	4	8	20000	4000000000	False	
4	50000	10	25	5	False	
5	10	20	50	10	False	
6	15	30	75	15	False	
7	20	40	100	20	False	
8	2500000	500	125	200	False	
9	300000000	60	150	30	False	
10	100000000	200	500000	10000	False	

	Outlier_Feature2	Outlier_Feature3	Outlier_Feature4
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	True
4	False	False	False
5	False	False	False
6	False	False	False
7	False	False	False
8	False	False	False
9	False	False	False
10	False	True	False

[6]:
```
#Elimina le righe corrispondenti agli outliers quelli che hanno una features␣
 ↪fuoriscala
outliers = df['Num_Outliers'] = df.filter(like='Outlier_').sum(axis=1)
df
```

[6]:

	Feature1	Feature2	Feature3	Feature4	Outlier_Feature1	\
0	1	2	5	1	False	
1	200	4	10	-200000	False	
2	3	6	15	3	False	
3	4	8	20000	4000000000	False	
4	50000	10	25	5	False	
5	10	20	50	10	False	

```
6          15        30        75          15             False
7          20        40       100          20             False
8     2500000       500       125         200             False
9   300000000        60       150          30             False
10  100000000       200    500000       10000             False

    Outlier_Feature2  Outlier_Feature3  Outlier_Feature4  Num_Outliers
0              False             False             False             0
1              False             False             False             0
2              False             False             False             0
3              False             False              True             1
4              False             False             False             0
5              False             False             False             0
6              False             False             False             0
7              False             False             False             0
8              False             False             False             0
9              False             False             False             0
10             False              True             False             1
```

[7]:
```
# Filtra i dati per mantenere solo le righe con almeno il numero minimo di␣
 ↪features superanti la soglia
outliers = df[df['Num_Outliers'] >= min_features_threshold]
outliers
```

[7]:
```
     Feature1  Feature2  Feature3    Feature4  Outlier_Feature1  \
3           4         8     20000  4000000000             False
10  100000000       200    500000       10000             False

    Outlier_Feature2  Outlier_Feature3  Outlier_Feature4  Num_Outliers
3              False             False              True             1
10             False              True             False             1
```

[8]:
```
# Aggiungi una colonna che indica se il record è un outlier o meno
df['Is_Outlier'] = df.index.isin(outliers.index)
df
```

[8]:
```
     Feature1  Feature2  Feature3    Feature4  Outlier_Feature1  \
0           1         2         5           1             False
1         200         4        10     -200000             False
2           3         6        15           3             False
3           4         8     20000  4000000000             False
4       50000        10        25           5             False
5          10        20        50          10             False
6          15        30        75          15             False
7          20        40       100          20             False
8     2500000       500       125         200             False
9   300000000        60       150          30             False
```

```
10  100000000       200    500000       10000             False

    Outlier_Feature2  Outlier_Feature3  Outlier_Feature4  Num_Outliers  \
0              False             False             False             0
1              False             False             False             0
2              False             False             False             0
3              False             False              True             1
4              False             False             False             0
5              False             False             False             0
6              False             False             False             0
7              False             False             False             0
8              False             False             False             0
9              False             False             False             0
10             False              True             False             1

    Is_Outlier
0        False
1        False
2        False
3         True
4        False
5        False
6        False
7        False
8        False
9        False
10        True
```

[9]:
```
# Rimuovi colonne ausiliarie
df.drop(df.filter(like='Outlier_').columns, axis=1, inplace=True)
df.drop('Num_Outliers', axis=1, inplace=True)
df
```

[9]:
```
      Feature1  Feature2  Feature3    Feature4  Is_Outlier
0            1         2         5           1       False
1          200         4        10     -200000       False
2            3         6        15           3       False
3            4         8     20000  4000000000        True
4        50000        10        25           5       False
5           10        20        50          10       False
6           15        30        75          15       False
7           20        40       100          20       False
8      2500000       500       125         200       False
9    300000000        60       150          30       False
10   100000000       200    500000       10000        True
```

[10]:
```
df_filtered = df[df['Is_Outlier'] == False ]
df_filtered
```

[10]:

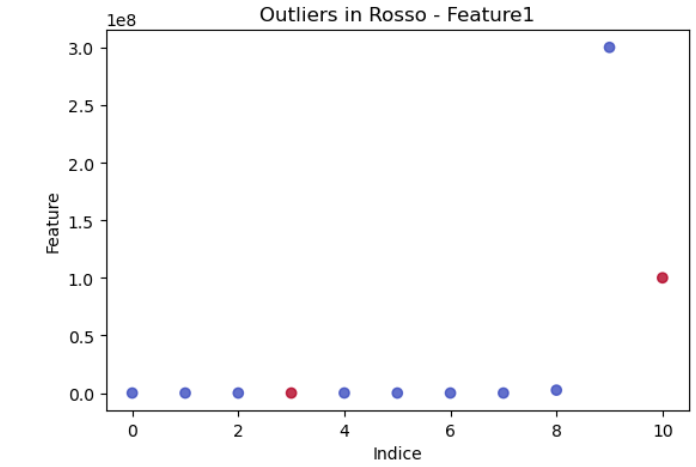
	Feature1	Feature2	Feature3	Feature4	Is_Outlier
0	1	2	5	1	False
1	200	4	10	-200000	False
2	3	6	15	3	False
4	50000	10	25	5	False
5	10	20	50	10	False
6	15	30	75	15	False
7	20	40	100	20	False
8	2500000	500	125	200	False
9	300000000	60	150	30	False

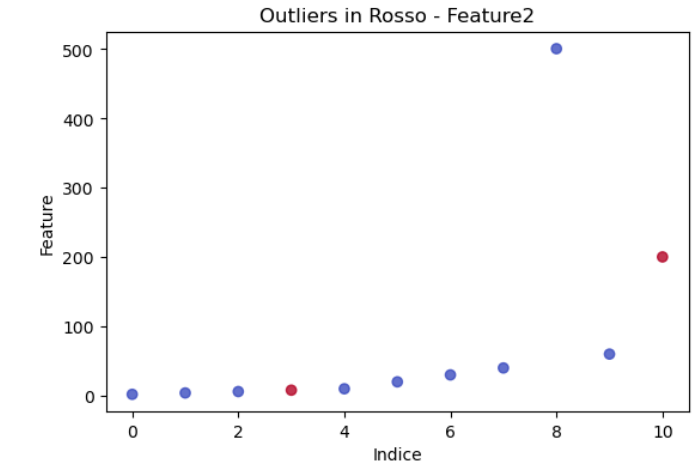
[13]:
```
df_filtered = df[df['Is_Outlier'] == False ]
df_filtered
```

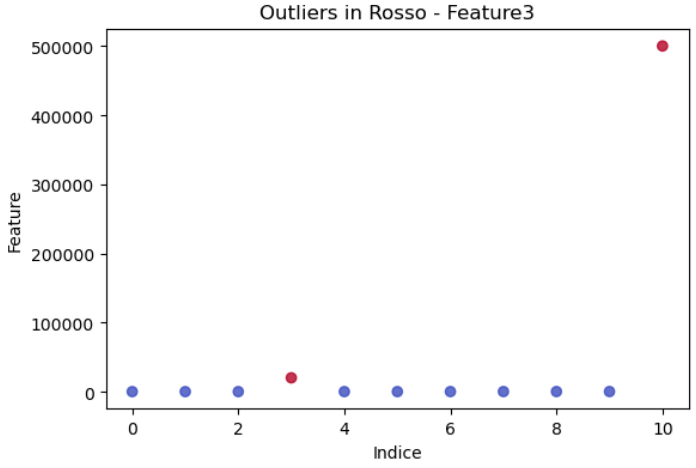
[13]:

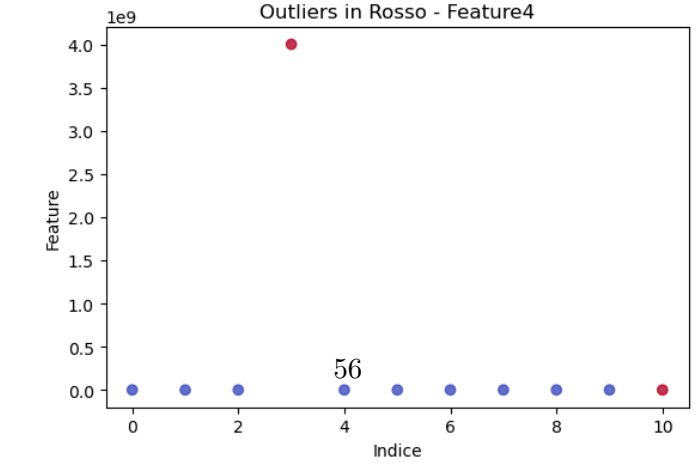
	Feature1	Feature2	Feature3	Feature4	Is_Outlier
0	1	2	5	1	False
1	200	4	10	-200000	False
2	3	6	15	3	False
4	50000	10	25	5	False
5	10	20	50	10	False
6	15	30	75	15	False
7	20	40	100	20	False
8	2500000	500	125	200	False
9	300000000	60	150	30	False

[12]:
```
# Organizza i grafici in una matrice, con una colonna e 4 righe
num_features = len(df.columns) - 1  # Escludi la colonna 'Is_Outlier'
num_features
num_rows = num_features
num_cols = 1  # Una colonna

plt.figure(figsize=(6, 4 * num_rows))
for i, feature in enumerate(df.columns[:-1]):  # Escludi la colonna 'Is_Outlier'
    plt.subplot(num_rows, num_cols, i + 1)
    plt.scatter(df.index, df[feature], c=df['Is_Outlier'], cmap='coolwarm',␣
 ↪alpha=0.8)
    plt.title(f'Outliers in Rosso - {feature}')
    plt.xlabel('Indice')
    plt.ylabel('Feature')

plt.tight_layout()
plt.show()
```


Outliers in Rosso - Feature1
1e8
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Feature
0
2
4
6
8
10
Indice


Outliers in Rosso - Feature2
500
400
300
200
100
0
Feature
0
2
4
6
8
10
Indice


Outliers in Rosso - Feature3
500000
400000
300000
200000
100000
0
Feature
0
2
4
6
8
10
Indice


Outliers in Rosso - Feature4
1e9
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Feature
0
2
4
6
8
10
Indice