In [1]:
```
import pandas as pd
```

In [2]:
```
data=pd.read_csv("/home/placement/Downloads/Titanic Dataset.csv")
```

In [3]:
```
data.describe()
```

Out[3]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [4]:
```
data.head()
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [5]:
```
data.isna().sum()
```

Out[5]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [6]:
```
data.head(10)
```

Out[6]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C

In [7]:
```
data['PassengerId'].unique()
```

Out[7]:
```
array([  1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,  13,
        14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,  26,
        27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,  39,
        40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,  52,
        53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,  65,
        66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,  78,
        79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,  91,
        92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103, 104,
       105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117,
       118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130,
       131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143,
       144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156,
       157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169,
       170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182,
       183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195,
       196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208,
       209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221,
       222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234,
       235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247,
       248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260,
       261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273,
       274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286,
       287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299,
       300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312,
       313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325,
       326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338,
       339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351,
       352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364,
       365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377,
       378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390,
       391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403,
       404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416,
       417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429,
       430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442,
       443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455,
       456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468,
       469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481,
       482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494,
```

```
       495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507,
       508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520,
       521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533,
       534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546,
       547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559,
       560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572,
       573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585,
       586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598,
       599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611,
       612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624,
       625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637,
       638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650,
       651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663,
       664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676,
       677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689,
       690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702,
       703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715,
       716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728,
       729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741,
       742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754,
       755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767,
       768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780,
       781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793,
       794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806,
       807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819,
       820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832,
       833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845,
       846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858,
       859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871,
       872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884,
       885, 886, 887, 888, 889, 890, 891])
```

In [8]:
```
data['Survived'].unique()
```

Out[8]:
```
array([0, 1])
```

In [9]:
```
data['Pclass'].unique()
```

Out[9]:
```
array([3, 1, 2])
```

In [10]:
```
data['Name'].unique()
```

```
       'Hickman, Mr. Stanley George', 'Moore, Mr. Leonard Charles',
       'Nasser, Mr. Nicholas', 'Webber, Miss. Susan',
       'White, Mr. Percival Wayland', 'Nicola-Yarred, Master. Elias',
       'McMahon, Mr. Martin', 'Madsen, Mr. Fridtjof Arne',
       'Peter, Miss. Anna', 'Ekstrom, Mr. Johan', 'Drazenoic, Mr. Jozef',
       'Coelho, Mr. Domingos Fernandeo',
       'Robins, Mrs. Alexander A (Grace Charity Laury)',
       'Weisz, Mrs. Leopold (Mathilde Francoise Pede)',
       'Sobey, Mr. Samuel James Hayden', 'Richard, Mr. Emile',
       'Newsom, Miss. Helen Monypeny', 'Futrelle, Mr. Jacques Heath',
       'Osen, Mr. Olaf Elon', 'Giglio, Mr. Victor',
       'Boulos, Mrs. Joseph (Sultana)', 'Nysten, Miss. Anna Sofia',
       'Hakkarainen, Mrs. Pekka Pietari (Elin Matilda Dolck)',
       'Burke, Mr. Jeremiah', 'Andrew, Mr. Edgardo Samuel',
       'Nicholls, Mr. Joseph Charles',
       'Andersson, Mr. August Edvard ("Wennerstrom")',
       'Ford, Miss. Robina Maggie "Ruby"',
       'Navratil, Mr. Michel ("Louis M Hoffman")',
       'Byles, Rev. Thomas Roussel Davids', 'Bateman, Rev. Robert James',
```

In [11]:
```
data['Age'].unique()
```

Out[11]:
```
array([22.  , 38.  , 26.  , 35.  ,   nan, 54.  ,  2.  , 27.  , 14.  ,
        4.  , 58.  , 20.  , 39.  , 55.  , 31.  , 34.  , 15.  , 28.  ,
        8.  , 19.  , 40.  , 66.  , 42.  , 21.  , 18.  ,  3.  ,  7.  ,
       49.  , 29.  , 65.  , 28.5 ,  5.  , 11.  , 45.  , 17.  , 32.  ,
       16.  , 25.  ,  0.83, 30.  , 33.  , 23.  , 24.  , 46.  , 59.  ,
       71.  , 37.  , 47.  , 14.5 , 70.5 , 32.5 , 12.  ,  9.  , 36.5 ,
       51.  , 55.5 , 40.5 , 44.  ,  1.  , 61.  , 56.  , 50.  , 36.  ,
       45.5 , 20.5 , 62.  , 41.  , 52.  , 63.  , 23.5 ,  0.92, 43.  ,
       60.  , 10.  , 64.  , 13.  , 48.  ,  0.75, 53.  , 57.  , 80.  ,
       70.  , 24.5 ,  6.  ,  0.67, 30.5 ,  0.42, 34.5 , 74.  ])
```

In [12]:
```
data['SibSp'].unique()
```

Out[12]:
```
array([1, 0, 3, 4, 2, 5, 8])
```

In [13]:
```
data['Parch'].unique()
```

Out[13]:
```
array([0, 1, 2, 5, 3, 4, 6])
```

In [14]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [15]:
```
data1=data.drop(['PassengerId','Name','Ticket','Cabin','SibSp','Parch'],axis=1)
```

In [16]:
```
data1.isna().sum()
```

Out[16]:
```
Survived      0
Pclass        0
Sex           0
Age         177
Fare          0
Embarked      2
dtype: int64
```

In [17]:
```
data1['Sex']=data1['Sex'].map({'male':1,'female':0})
data['Pclass'].unique()
```

Out[17]:
```
array([3, 1, 2])
```

In [18]:
```
data1
```

Out[18]:

	Survived	Pclass	Sex	Age	Fare	Embarked
0	0	3	1	22.0	7.2500	S
1	1	1	0	38.0	71.2833	C
2	1	3	0	26.0	7.9250	S
3	1	1	0	35.0	53.1000	S
4	0	3	1	35.0	8.0500	S
...	...	...	...	...	...	...
886	0	2	1	27.0	13.0000	S
887	1	1	0	19.0	30.0000	S
888	0	3	0	NaN	23.4500	S
889	1	1	1	26.0	30.0000	C
890	0	3	1	32.0	7.7500	Q

891 rows × 6 columns

In [19]:
```
data2=data1.fillna(data1.median)
```

In [20]:
```
data2
```

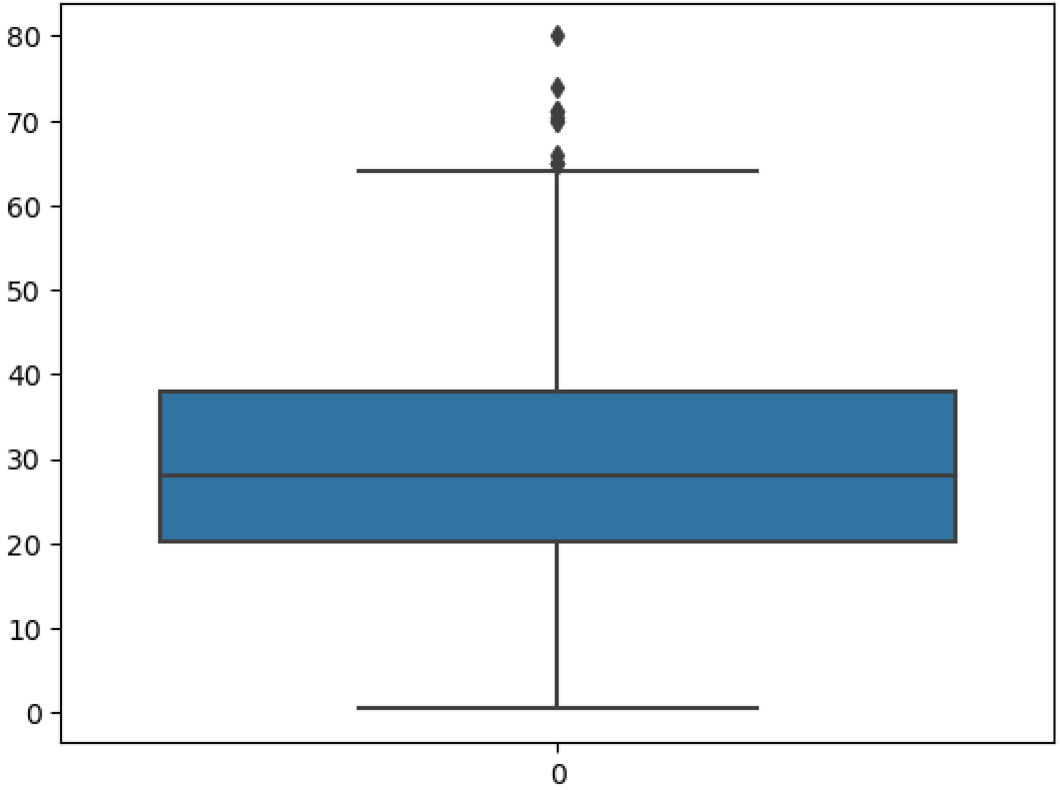
Out[20]:

	Survived	Pclass	Sex	Age	Fare	Embarked
0	0	3	1	22.0	7.2500	S
1	1	1	0	38.0	71.2833	C
2	1	3	0	26.0	7.9250	S
3	1	1	0	35.0	53.1000	S
4	0	3	1	35.0	8.0500	S
...	...	...	...	...	...	...
886	0	2	1	27.0	13.0000	S
887	1	1	0	19.0	30.0000	S
888	0	3	0	<bound method NDFrame._add_numeric_operations....	23.4500	S
889	1	1	1	26.0	30.0000	C
890	0	3	1	32.0	7.7500	Q

891 rows × 6 columns

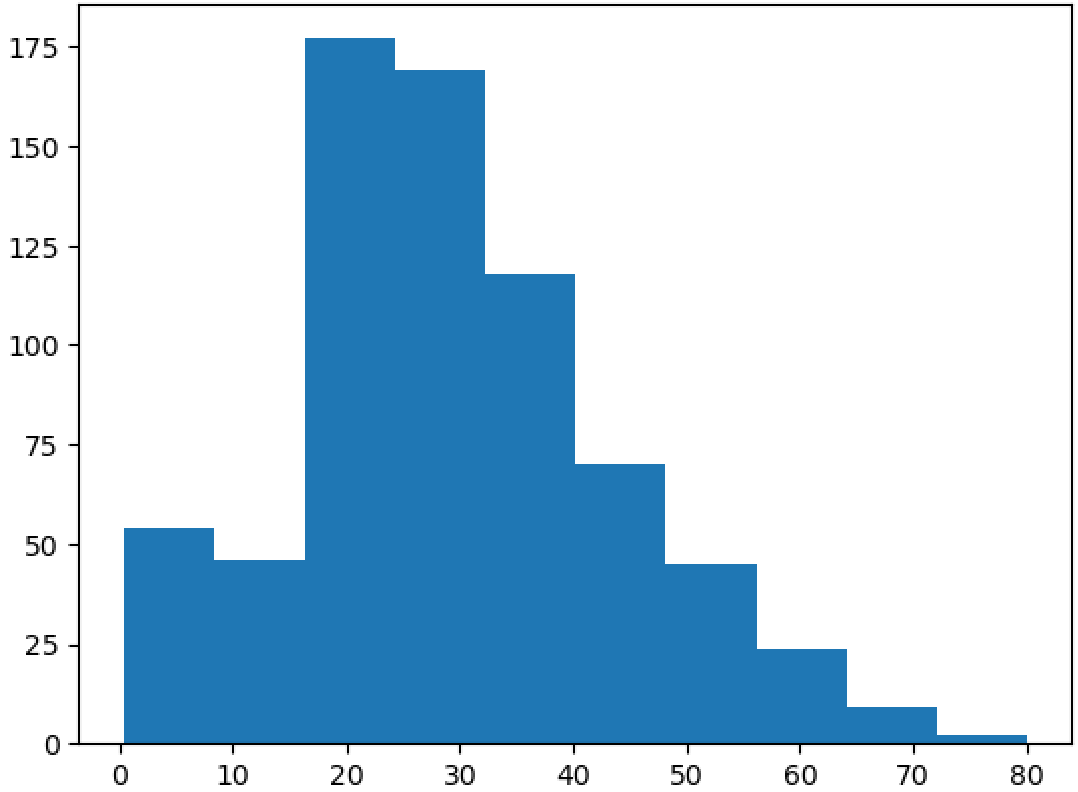
In [21]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(data.Age)
```

Out[21]: <Axes: >



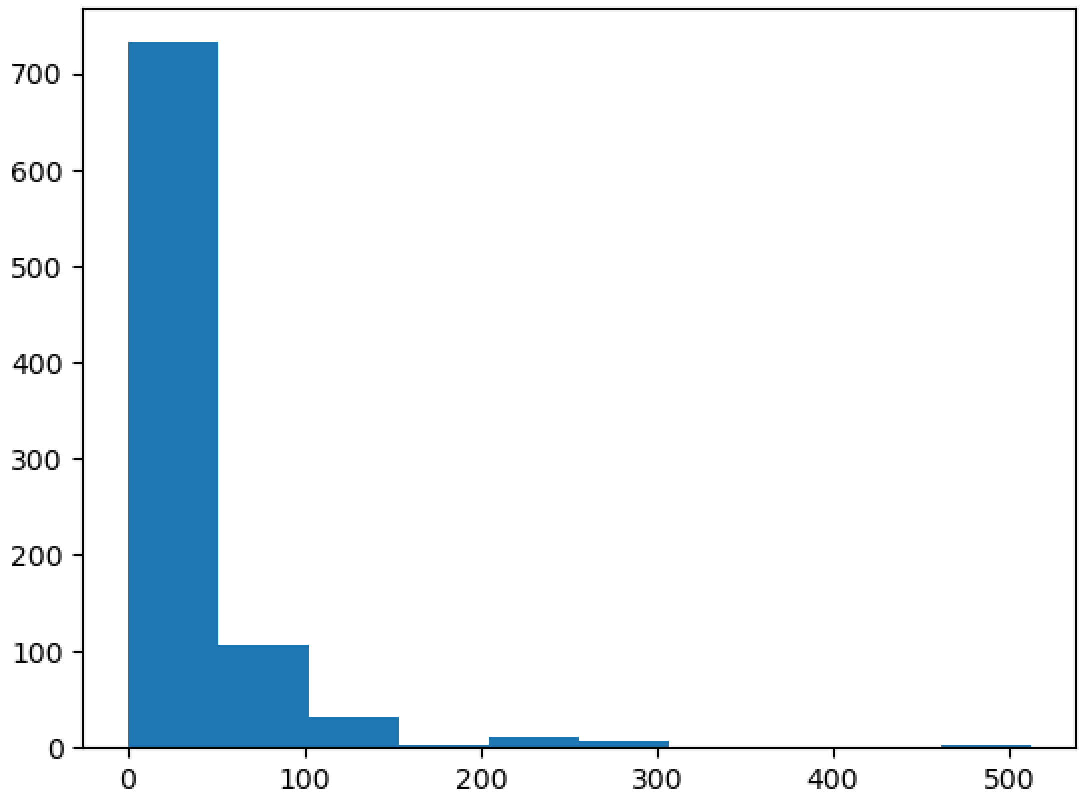
In [22]:
```
plt.hist(data1['Age'])
```

Out[22]:
```
(array([ 54.,  46., 177., 169., 118.,  70.,  45.,  24.,   9.,   2.]),
 array([ 0.42 ,  8.378, 16.336, 24.294, 32.252, 40.21 , 48.168, 56.126,
        64.084, 72.042, 80.   ]),
 <BarContainer object of 10 artists>)
```



In [23]:
```
plt.hist(data1['Fare'])
```

Out[23]:
```
(array([732., 106.,  31.,   2.,  11.,   6.,   0.,   0.,   0.,   3.]),
 array([  0.     ,  51.23292, 102.46584, 153.69876, 204.93168, 256.1646 ,
        307.39752, 358.63044, 409.86336, 461.09628, 512.3292 ]),
 <BarContainer object of 10 artists>)
```



In [24]:
```
data2.isna().sum()
```

Out[24]:
```
Survived    0
Pclass      0
Sex         0
Age         0
Fare        0
Embarked    0
dtype: int64
```

In [25]:
```
data1.fillna(35,inplace=True)
```

In [26]:
```
data1.describe()
```

Out[26]:

	Survived	Pclass	Sex	Age	Fare
count	891.000000	891.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	0.647587	30.752155	32.204208
std	0.486592	0.836071	0.477990	13.173100	49.693429
min	0.000000	1.000000	0.000000	0.420000	0.000000
25%	0.000000	2.000000	0.000000	22.000000	7.910400
50%	0.000000	3.000000	1.000000	32.000000	14.454200
75%	1.000000	3.000000	1.000000	35.000000	31.000000
max	1.000000	3.000000	1.000000	80.000000	512.329200

In [27]:
```
data1['Age'].unique()
```

Out[27]:
```
array([22.  , 38.  , 26.  , 35.  , 54.  ,  2.  , 27.  , 14.  ,  4.  ,
       58.  , 20.  , 39.  , 55.  , 31.  , 34.  , 15.  , 28.  ,  8.  ,
       19.  , 40.  , 66.  , 42.  , 21.  , 18.  ,  3.  ,  7.  , 49.  ,
       29.  , 65.  , 28.5 ,  5.  , 11.  , 45.  , 17.  , 32.  , 16.  ,
       25.  ,  0.83, 30.  , 33.  , 23.  , 24.  , 46.  , 59.  , 71.  ,
       37.  , 47.  , 14.5 , 70.5 , 32.5 , 12.  ,  9.  , 36.5 , 51.  ,
       55.5 , 40.5 , 44.  ,  1.  , 61.  , 56.  , 50.  , 36.  , 45.5 ,
       20.5 , 62.  , 41.  , 52.  , 63.  , 23.5 ,  0.92, 43.  , 60.  ,
       10.  , 64.  , 13.  , 48.  ,  0.75, 53.  , 57.  , 80.  , 70.  ,
       24.5 ,  6.  ,  0.67, 30.5 ,  0.42, 34.5 , 74.  ])
```

In [28]:
```
data1.groupby(['Age']).count()
```

Out[28]:

	Survived	Pclass	Sex	Fare	Embarked
Age					
0.42	1	1	1	1	1
0.67	1	1	1	1	1
0.75	2	2	2	2	2
0.83	2	2	2	2	2
0.92	1	1	1	1	1
...	...	...	...	...	...
70.00	2	2	2	2	2
70.50	1	1	1	1	1
71.00	2	2	2	2	2
74.00	1	1	1	1	1
80.00	1	1	1	1	1

88 rows × 5 columns

In [ ]:

In [29]:
```
data1['Pclass']=data1['Pclass'].map({1:'F',2:'S',3:'Third'})
```

In [30]:
```
data1.isna().sum()
```

Out[30]:
```
Survived    0
Pclass      0
Sex         0
Age         0
Fare        0
Embarked    0
dtype: int64
```

In [31]:
```
data1.head(5)
```

Out[31]:

	Survived	Pclass	Sex	Age	Fare	Embarked
0	0	Third	1	22.0	7.2500	S
1	1	F	0	38.0	71.2833	C
2	1	Third	0	26.0	7.9250	S
3	1	F	0	35.0	53.1000	S
4	0	Third	1	35.0	8.0500	S

In [32]:
```
data=pd.get_dummies(data1)
```

In [33]:
```
data.shape
```

Out[33]:
```
(891, 11)
```

In [34]:
```
data1.head(500)
```

Out[34]:

	Survived	Pclass	Sex	Age	Fare	Embarked
0	0	Third	1	22.0	7.2500	S
1	1	F	0	38.0	71.2833	C
2	1	Third	0	26.0	7.9250	S
3	1	F	0	35.0	53.1000	S
4	0	Third	1	35.0	8.0500	S
...	...	...	...	...	...	...
495	0	Third	1	35.0	14.4583	C
496	1	F	0	54.0	78.2667	C
497	0	Third	1	35.0	15.1000	S
498	0	F	0	25.0	151.5500	S
499	0	Third	1	24.0	7.7958	S

500 rows × 6 columns

In [35]:

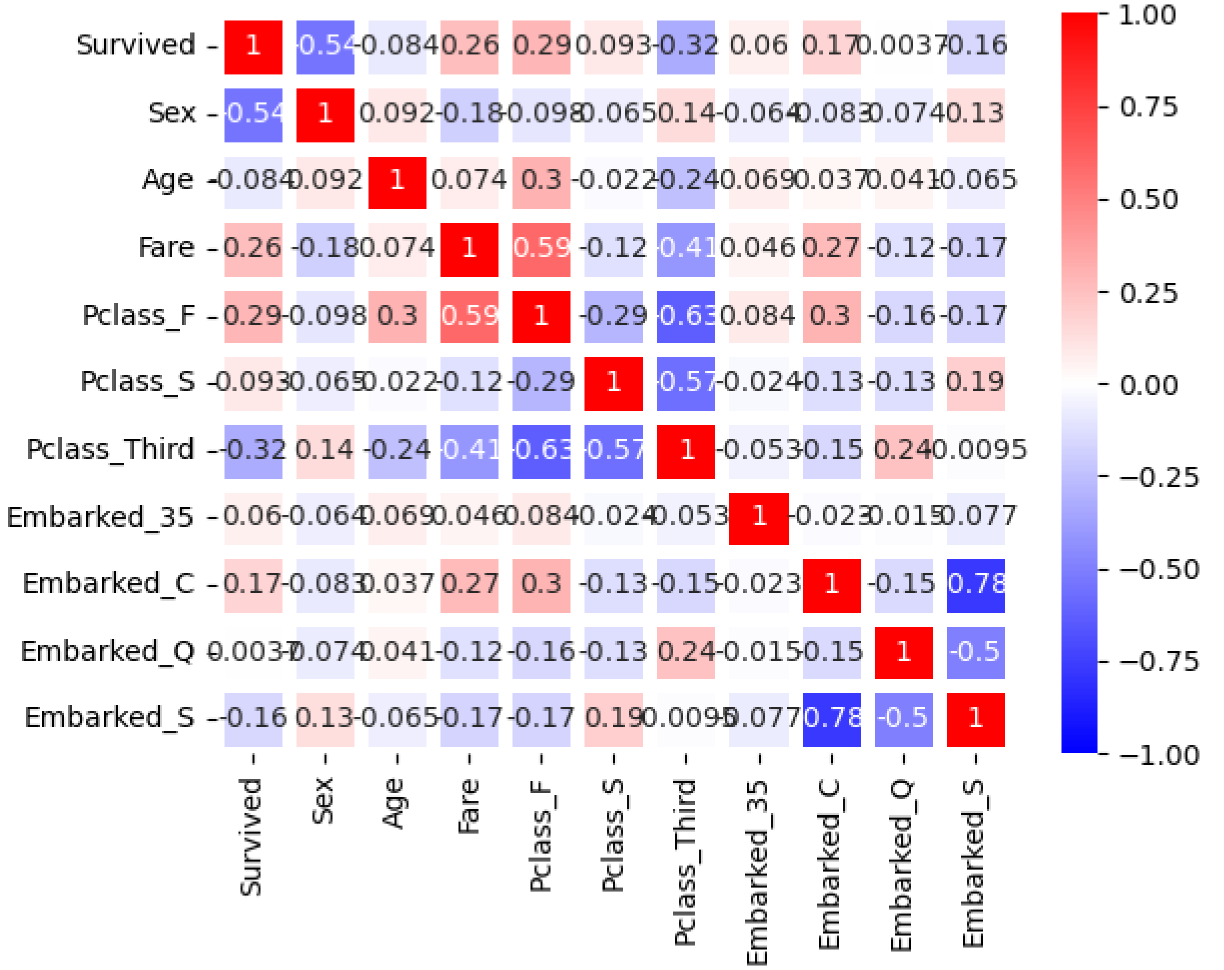
```
cor_mat=data.corr()
cor_mat
```

Out[35]:

	Survived	Sex	Age	Fare	Pclass_F	Pclass_S	Pclass_Third	Embarked_35	Embarked_C	Embarked_Q	Embarked_
Survived	1.000000	-0.543351	-0.083713	0.257307	0.285904	0.093349	-0.322308	0.060095	0.168240	0.003650	-0.1556
Sex	-0.543351	1.000000	0.091930	-0.182333	-0.098013	-0.064746	0.137143	-0.064296	-0.082853	-0.074115	0.1257
Age	-0.083713	0.091930	1.000000	0.074199	0.302149	-0.022021	-0.242412	0.069343	0.036953	0.040528	-0.0650
Fare	0.257307	-0.182333	0.074199	1.000000	0.591711	-0.118557	-0.413333	0.045646	0.269335	-0.117216	-0.1666
Pclass_F	0.285904	-0.098013	0.302149	0.591711	1.000000	-0.288585	-0.626738	0.083847	0.296423	-0.155342	-0.1703
Pclass_S	0.093349	-0.064746	-0.022021	-0.118557	-0.288585	1.000000	-0.565210	-0.024197	-0.125416	-0.127301	0.1920
Pclass_Third	-0.322308	0.137143	-0.242412	-0.413333	-0.626738	-0.565210	1.000000	-0.052550	-0.153329	0.237449	-0.0095
Embarked_35	0.060095	-0.064296	0.069343	0.045646	0.083847	-0.024197	-0.052550	1.000000	-0.022864	-0.014588	-0.0765
Embarked_C	0.168240	-0.082853	0.036953	0.269335	0.296423	-0.125416	-0.153329	-0.022864	1.000000	-0.148258	-0.7783
Embarked_Q	0.003650	-0.074115	0.040528	-0.117216	-0.155342	-0.127301	0.237449	-0.014588	-0.148258	1.000000	-0.4966
Embarked_S	-0.155660	0.125722	-0.065062	-0.166603	-0.170379	0.192061	-0.009511	-0.076588	-0.778359	-0.496624	1.0000

In [36]:
```
sns.heatmap(cor_mat,vmax=1,vmin=-1,annot=True,linewidth=-5,cmap='bwr')
```

Out[36]: <Axes: >



In [37]:
```
data.groupby('Survived').count()
```

Out[37]:

	Sex	Age	Fare	Pclass_F	Pclass_S	Pclass_Third	Embarked_35	Embarked_C	Embarked_Q	Embarked_S
Survived										
0	549	549	549	549	549	549	549	549	549	549
1	342	342	342	342	342	342	342	342	342	342

In [38]:
```
y=data['Survived']
x=data.drop('Survived',axis=1)
```

In [39]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.33,random_state=42)
```

In [40]:

```
from sklearn.linear_model import LogisticRegression
classifier= LogisticRegression()
classifier.fit(x_train,y_train)
```

```
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWa
rning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/pre
processing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.or
g/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[40]:

```
▾ LogisticRegression
LogisticRegression()
```

In [41]:

```
y_pred=classifier.predict(x_test)
```

In [42]:

```
y_pred
```

Out[42]:

```
array([0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0,
       0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 1, 0])
```

In [43]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
```

Out[43]:
```
array([[155,  20],
       [ 37,  83]])
```

In [44]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[44]:
```
0.8067796610169492
```

In [ ]: