In [1]:

```
import pandas as pd
```

In [3]:

```
df = pd.read_csv('exp4.csv')
df.head()
```

Out[3]:

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

In [7]:

```
df.Sex = df.Sex.map({'male':1, 'female':2})
df.head()
```

Out[7]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cal
0	1	0	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.2500	N
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	2	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	2	26.0	0	0	STON/O2. 3101282	7.9250	N
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	2	35.0	1	0	113803	53.1000	C1
4	5	0	3	Allen, Mr. William Henry	1	35.0	0	0	373450	8.0500	N

In [8]:

```
df.corr()
```

Out[8]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	
PassengerId	1.000000	-0.005007	-0.035144	-0.042939	0.036847	-0.057527	-0.001652	0.0
Survived	-0.005007	1.000000	-0.338481	0.543351	-0.077221	-0.035322	0.081629	0.2
Pclass	-0.035144	-0.338481	1.000000	-0.131900	-0.369226	0.083081	0.018443	-0.5
Sex	-0.042939	0.543351	-0.131900	1.000000	-0.093254	0.114631	0.245489	0.1
Age	0.036847	-0.077221	-0.369226	-0.093254	1.000000	-0.308247	-0.189119	0.0
SibSp	-0.057527	-0.035322	0.083081	0.114631	-0.308247	1.000000	0.414838	0.1
Parch	-0.001652	0.081629	0.018443	0.245489	-0.189119	0.414838	1.000000	0.2
Fare	0.012658	0.257307	-0.549500	0.182333	0.096067	0.159651	0.216225	1.0

In [9]:

```
# Features with highest correlation - Sex, Pclass, Fare

df.drop(['PassengerId', 'Name', 'Age', 'SibSp', 'Parch', 'Ticket', 'Cabin', 'Embarked'], ax
df.head()
```

Out[9]:

	Survived	Pclass	Sex	Fare
0	0	3	1	7.2500
1	1	1	2	71.2833
2	1	3	2	7.9250
3	1	1	2	53.1000
4	0	3	1	8.0500

In [10]:

```
inputs = df.drop('Survived', axis='columns')
inputs.head()
```

Out[10]:

	Pclass	Sex	Fare
0	3	1	7.2500
1	1	2	71.2833
2	3	2	7.9250
3	1	2	53.1000
4	3	1	8.0500

In [11]:

```
target = df.Survived
target.head()
```

Out[11]:

```
0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
```

In [12]:

```
from sklearn.model_selection import train_test_split
```

In [13]:

```
x_train, x_test, y_train, y_test = train_test_split(inputs, target, test_size=0.3)
print(x_train)
```

```
     Pclass  Sex     Fare
384       3    1   7.8958
591       1    2  78.2667
6         1    1  51.8625
676       3    1   8.0500
773       3    1   7.2250
..      ...  ...      ...
236       2    1  26.0000
771       3    1   7.8542
83        1    1  47.1000
709       3    1  15.2458
383       1    2  52.0000

[623 rows x 3 columns]
```

In [14]:

```
from sklearn.svm import SVC
```

In [15]:

```
clf = SVC(kernel='linear', random_state=0)
clf.fit(x_train, y_train)
```

Out[15]:

```
SVC(kernel='linear', random_state=0)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [16]:

```
clf.support_vectors_
```

Out[16]:

```
array([[  1.    ,   1.    ,  51.8625],
       [  1.    ,   1.    ,  27.7208],
       [  3.    ,   2.    ,   7.8958],
       [  2.    ,   1.    ,   0.    ],
       [  2.    ,   1.    ,  27.75  ],
       [  3.    ,   2.    ,  31.275 ],
       [  3.    ,   2.    ,  69.55  ],
       [  1.    ,   1.    ,  82.1708],
       [  3.    ,   2.    ,   7.55  ],
       [  1.    ,   1.    ,  27.7208],
       [  3.    ,   2.    ,   7.75  ],
       [  2.    ,   2.    ,  13.    ],
       [  3.    ,   2.    ,  27.9   ],
       [  1.    ,   1.    , 153.4625],
       [  1.    ,   1.    ,  30.6958],
       [  3.    ,   2.    ,   7.6292],
       [  1.    ,   1.    ,  71.    ],
       [  1.    ,   1.    ,  35.    ],
```

In [17]:

```
clf.n_support_
```

Out[17]:

```
array([143, 140])
```

In [18]:

```
clf.support_
```

Out[18]:

```
array([  2,   6,  17,  19,  29,  30,  35,  42,  44,  51,  57,  58,  72,
        73,  79,  80,  84,  96,  99, 100, 105, 106, 108, 111, 114, 115,
       116, 118, 128, 132, 134, 136, 137, 141, 144, 145, 154, 156, 159,
       167, 168, 170, 175, 181, 182, 184, 185, 190, 197, 200, 203, 206,
       224, 225, 227, 229, 234, 236, 242, 243, 244, 249, 252, 260, 275,
       282, 286, 293, 301, 305, 314, 318, 320, 323, 326, 327, 331, 333,
       335, 340, 342, 347, 348, 350, 352, 358, 366, 367, 371, 380, 381,
       389, 391, 393, 394, 397, 398, 399, 400, 412, 415, 418, 419, 424,
       427, 437, 453, 459, 472, 476, 482, 485, 491, 492, 497, 500, 501,
       502, 504, 508, 510, 513, 519, 525, 530, 534, 538, 540, 544, 551,
       557, 571, 577, 580, 583, 585, 590, 597, 600, 603, 609, 612, 614,
         5,  11,  13,  15,  20,  22,  24,  27,  28,  36,  39,  41,  46,
        59,  61,  63,  75,  76,  82,  83,  85,  87,  89,  92,  95, 107,
       110, 112, 122, 126, 127, 138, 142, 148, 151, 152, 160, 163, 164,
       165, 176, 177, 178, 186, 189, 192, 195, 204, 205, 228, 230, 231,
       232, 239, 240, 248, 262, 263, 267, 271, 273, 274, 276, 283, 285,
       300, 308, 310, 317, 319, 330, 332, 338, 341, 343, 344, 349, 363,
       364, 369, 376, 378, 384, 386, 387, 401, 402, 405, 407, 411, 416,
       417, 422, 428, 432, 440, 444, 451, 462, 464, 465, 467, 469, 474,
       475, 479, 481, 484, 489, 493, 496, 505, 507, 511, 512, 521, 524,
       531, 542, 543, 545, 549, 552, 553, 564, 565, 567, 572, 575, 578,
       581, 582, 586, 587, 594, 604, 606, 607, 621, 622])
```

In [19]:

```
y_pred = clf.predict(x_test)
print(y_pred)
```

```
[1 0 0 0 0 1 0 1 1 1 1 0 0 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 1 1 0 0 1 1 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 1 0
 0 1 0 0 0 1 0 1 0 0 0 1 0 0 0 1 1 1 0 1 1 0 0 0 0 0 1 0 1 1 1 0 0 0 0 1 1
 1 1 1 0 1 1 0 1 0 0 0 0 1 0 0 1 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 0
 1 1 1 1 0 1 1 0 0 1 0 1 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 0 0 0
 0 1 1 1 1 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 1 1 0 0 1 0 0 1 0
 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 1 0 1 0 0 1 0 1 0 0 1
 1 0 0 1 0 0 0 0 0]
```

In [20]:

```
from sklearn.metrics import confusion_matrix
```

In [21]:

```
cm = confusion_matrix(y_test, y_pred)
cm
```

Out[21]:

```
array([[147,  27],
       [ 28,  66]], dtype=int64)
```

In [22]:

```
from sklearn.metrics import accuracy_score, precision_score, recall_score
```

In [23]:

```
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy = ', accuracy)

positive_precision = precision_score(y_test, y_pred, pos_label=1)
negative_precision = precision_score(y_test, y_pred, pos_label=0)
print('Positive Precision = ', positive_precision)
print('Negative Precision = ', negative_precision)

recall_sensitivity = recall_score(y_test, y_pred, pos_label=1)
recall_specificity = recall_score(y_test, y_pred, pos_label=0)
print('Recall sensitivity = ', recall_sensitivity)
print('Recall specificity = ', recall_specificity)
```

```
Accuracy =  0.7947761194029851
Positive Precision =  0.7096774193548387
Negative Precision =  0.84
Recall sensitivity =  0.7021276595744681
Recall specificity =  0.8448275862068966
```