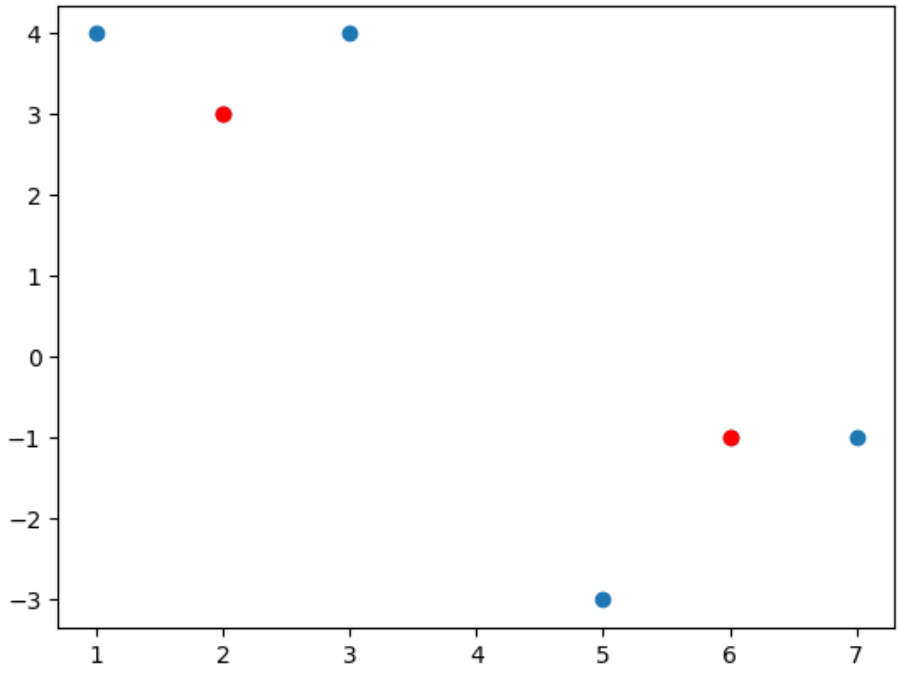
In [2]:
```
import numpy as np
from sklearn.svm import SVC

X = np.array([[3,4],[1,4],[2,3],[6,-1],[7,-1],[5,-3]] )
y = np.array([-1,-1, -1, 1, 1 , 1 ])

clt=SVC(C=1e4 , kernel='linear')
clt.fit(X,y)

print("Coeff: ",clt.coef_)
print("Intercepts: ",clt.intercept_)
print("Support: ",clt.support_)
print("Supp Vectors: ",clt.support_vectors_)
print("N supp: ",clt.n_support_)
```

```
Coeff:  [[ 0.25 -0.25]]
Intercepts:  [-0.75]
Support:  [2 3]
Supp Vectors:  [[ 2.  3.]
 [ 6. -1.]]
N supp:  [1 1]
```

In [4]:
```
x1=clt.support_vectors_[0]
x2=clt.support_vectors_[1]
```

In [5]:
```
import matplotlib.pyplot as plt
plt.scatter(X[:,0],X[:,1])
plt.scatter(x1[0],x1[1],color='red')
plt.scatter(x2[0],x2[1],color='red')
plt.show()
```



In [7]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.svm import SVC
```

In [8]:
```
data=pd.read_csv('glass.csv')
data
```

Out[8]:

	Id	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.00	0.0	1
1	2	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.00	0.0	1
2	3	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.00	0.0	1
3	4	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.00	0.0	1
4	5	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.00	0.0	1
...	...	...	...	...	...	...	...	...	...	...	...
209	210	1.51623	14.14	0.00	2.88	72.61	0.08	9.18	1.06	0.0	7
210	211	1.51685	14.92	0.00	1.99	73.06	0.00	8.40	1.59	0.0	7
211	212	1.52065	14.36	0.00	2.02	73.42	0.00	8.44	1.64	0.0	7
212	213	1.51651	14.38	0.00	1.94	73.61	0.00	8.48	1.57	0.0	7
213	214	1.51711	14.23	0.00	2.08	73.36	0.00	8.62	1.67	0.0	7

214 rows × 11 columns

In [14]:
```
X=data.drop('Type',axis=1)
Y=data.Type
```

In [15]:
```
X.head()
```

Out[15]:

	Id	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe
0	1	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.0
1	2	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.0
2	3	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.0
3	4	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.0
4	5	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.0

In [16]:
```
x_train, x_test, y_train,y_test=train_test_split(X,Y)
```

In [17]:
```
lin_model=SVC(kernel='linear')
lin_model.fit(x_train,y_train)
y_pred=lin_model.predict(x_test)
```

In [18]:
```
print("Accuracy ",accuracy_score(y_test,y_pred))
print(" Confusion matrix \n",confusion_matrix(y_test,y_pred))
print("Classification report ",classification_report(y_test,y_pred))
```

```
Accuracy  0.9629629629629629
 Confusion matrix
 [[16  0  0  0  0  0]
 [ 2 18  0  0  0  0]
 [ 0  0  3  0  0  0]
 [ 0  0  0  3  0  0]
 [ 0  0  0  0  5  0]
 [ 0  0  0  0  0  7]]
Classification report                precision    recall  f1-score   support

           1       0.89      1.00      0.94        16
           2       1.00      0.90      0.95        20
           3       1.00      1.00      1.00         3
           5       1.00      1.00      1.00         3
           6       1.00      1.00      1.00         5
           7       1.00      1.00      1.00         7

    accuracy                           0.96        54
   macro avg       0.98      0.98      0.98        54
weighted avg       0.97      0.96      0.96        54
```

In [19]:
```
model1=SVC(kernel='sigmoid',gamma=0.0001)
model2=SVC(kernel='poly',degree=3)
model3=SVC(kernel='rbf')
```

In [20]:
```
model1.fit(x_train,y_train)
model2.fit(x_train,y_train)
model3.fit(x_train,y_train)
```

Out[20]:
```
SVC
SVC()
```

In [21]:
```
ypred1=model1.predict(x_test)
ypred2=model2.predict(x_test)
ypred3=model3.predict(x_test)

print(accuracy_score(y_test,ypred1))
print(accuracy_score(y_test,ypred2))
print(accuracy_score(y_test,ypred3))
```

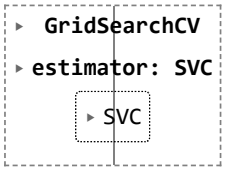
```
0.5925925925925926
0.8148148148148148
0.7592592592592593
```

In [22]:
```
params_grid = [{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4],
                     'C': [1, 10, 100, 1000]},
                    {'kernel': ['linear'], 'C': [1, 10, 100, 1000]},
                    {'kernel': ['poly'], 'gamma': [1e-3, 1e-4],'degree':[2,3,4]},
                    {'kernel': ['sigmoid'], 'C': [1, 10, 100, 1000],'gamma': [1e-3, 1e-4,1e-5]}
```

In [23]:
```
svm_model = GridSearchCV(SVC(), params_grid, cv=5)
svm_model.fit(x_train,y_train)
```

```
C:\Users\Admin\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\model_selecti
on\_split.py:700: UserWarning: The least populated class in y has only 4 members, which is les
s than n_splits=5.
  warnings.warn(
```

Out[23]:



In [24]:
```
print('Best score for training data:', svm_model.best_score_,"\n")

# View the best parameters for the model found using grid search
print('Best C:',svm_model.best_estimator_.C,"\n")
print('Best Kernel:',svm_model.best_estimator_.kernel,"\n")
print('Best Gamma:',svm_model.best_estimator_.gamma,"\n")

final_model = svm_model.best_estimator_
Y_pred = final_model.predict(x_test)
```

```
Best score for training data: 0.9875

Best C: 1.0

Best Kernel: poly

Best Gamma: 0.001
```

In [25]:
```
print(accuracy_score(y_test,Y_pred))
```

```
0.9444444444444444
```

In [ ]:
```

```