In [25]:

```
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.datasets import load_breast_cancer
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.svm import SVC
from time import time
import datetime

breast = load_breast_cancer()
breast_data = breast.data
breast_input = pd.DataFrame(breast_data)
breast_labels = breast.target
labels = np.reshape(breast_labels,(569,1))
final_breast_data = np.concatenate([breast_data,labels],axis=1)
breast_dataset = pd.DataFrame(final_breast_data)
features = breast.feature_names
features
```

Out[25]:

```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

In [26]:

```
features_labels = np.append(features,'label')
breast_dataset.columns = features_labels
breast_dataset['label'].replace('Benign',0,inplace=True)
breast_dataset['label'].replace('Malignant',1,inplace=True)
breast_dataset.tail()
```

Out[26]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mea symmetr
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.172
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.175
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.159
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.239
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.158

5 rows × 31 columns

In [27]:

```
from sklearn.preprocessing import StandardScaler
```

```
x = breast_dataset.loc[:, features].values
x = StandardScaler().fit_transform(x)
```

In [28]:

```
k=2     #设置降维的占比
pca= PCA(n_components=k)#调用PCA函数，先实例化
pcaCom = pca.fit_transform(x)
pcaCom = pd.DataFrame(pcaCom)
print("主成分的数量：",pca.n_components_)
X = pcaCom.iloc[:, [0, 1]].values
#X = pcaCom.iloc[:, [0, 1, 2, 3, 4, 5, 6, 7, 8]].values
Y = breast_dataset.iloc[:, 30].values
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, train_size=0.8, test_size=
```

```
主成分的数量： 2
```

In [29]:

```
Kernel = ["linear","poly","rbf","sigmoid"]
for kernel in Kernel:
    time0 = time()
    clf= SVC(kernel = kernel
             , gamma="auto"
             , degree = 1
             , cache_size=5000
           ).fit(X_train,Y_train)
    print("The accuracy under kernel %s is %f" % (kernel,clf.score(X_test,Y_test)))
    print(datetime.datetime.fromtimestamp(time()-time0).strftime("%M:%S:%f"))
Y_pred= clf.predict(X_test)
Y_pred[0:9]
```

```
The accuracy under kernel linear is 0.938596
00:00:007770
The accuracy under kernel poly is 0.938596
00:00:006672
The accuracy under kernel rbf is 0.938596
00:00:012376
The accuracy under kernel sigmoid is 0.921053
00:00:002083
```

Out[29]:

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

In [30]:

```
model = SVC(kernel='rbf')
model.fit(X_train, Y_train)
```

Out[30]:

▾ SVC

SVC()

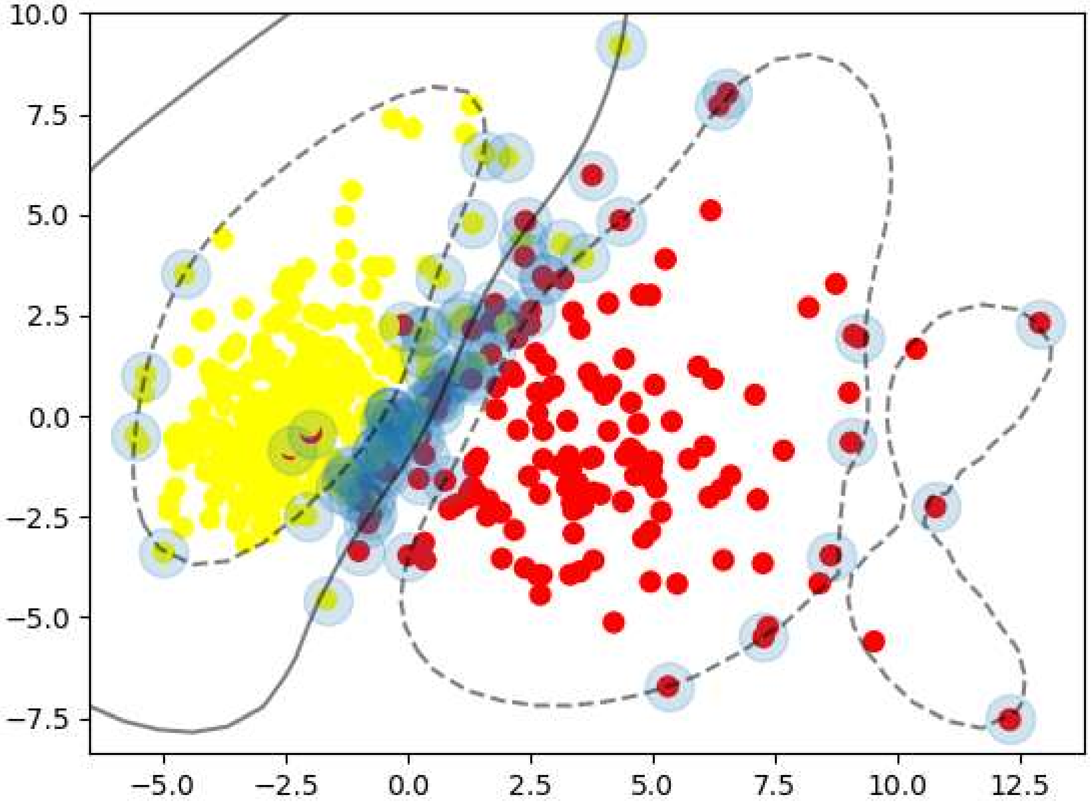
In [31]:

```
print("Accuracy:",metrics.accuracy_score(Y_test, Y_pred))
print("Precision:",metrics.precision_score(Y_test, Y_pred))
print("Recall:",metrics.recall_score(Y_test, Y_pred))
```

```
Accuracy: 0.9210526315789473
Precision: 0.9117647058823529
Recall: 0.9538461538461539
```

In [32]:

```
def plot_svc_decision_function(model, ax=None, plot_support=True):

    if ax is None:
        ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # 用SVM自带的decision_function函数来绘制
    x = np.linspace(xlim[0], xlim[1], 30)
    y = np.linspace(ylim[0], ylim[1], 30)
    Y, X = np.meshgrid(y, x)
    xy = np.vstack([X.ravel(), Y.ravel()]).T
    P = model.decision_function(xy).reshape(X.shape)

    # 绘制决策边界
    ax.contour(X, Y, P, colors='k',
               levels=[-1, 0, 1], alpha=0.5,
               linestyles=['--', '-', '--'])

    # 绘制支持向量
    if plot_support:
        ax.scatter(model.support_vectors_[:, 0],
                   model.support_vectors_[:, 1],
                   s=300, linewidth=1, alpha=0.2);
    ax.set_xlim(xlim)
    ax.set_ylim(ylim)

plt.scatter(X_train[:, 0], X_train[:, 1], c=Y_train, s=50, cmap='autumn')
plot_svc_decision_function (model);
```



In [150…

```
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.datasets import load_breast_cancer
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.svm import SVC
from time import time
import datetime
```

In [151…

```
housing = pd.DataFrame(pd.read_csv("Housing.csv"))
housing.head()
```

Out[151]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterhe
0	13300000	7420	4	2	3	yes	no	no	
1	12250000	8960	4	4	4	yes	no	no	
2	12250000	9960	3	2	2	yes	no	yes	
3	12215000	7500	4	2	2	yes	no	yes	
4	11410000	7420	4	1	2	yes	yes	yes	

In [152…

```
housing.shape
```

Out[152]:

```
(545, 13)
```

In [153…

```
varlist = ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning'

# Defining the map function
def binary_map(x):
    return x.map({'yes': 1, "no": 0})

# Applying the function to the housing list
housing[varlist] = housing[varlist].apply(binary_map)

# Check the housing dataframe now
housing.head()
```

Out[153]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterhe
0	13300000	7420	4	2	3	1	0	0	
1	12250000	8960	4	4	4	1	0	0	
2	12250000	9960	3	2	2	1	0	1	
3	12215000	7500	4	2	2	1	0	1	
4	11410000	7420	4	1	2	1	1	1	

In [154…

```
#Splitting the Data into Training and Testing Sets
```

```
from sklearn.model_selection import train_test_split

np.random.seed(0)
df_train, df_test = train_test_split(housing, train_size = 0.8, test_size = 0.2, rar
```

In [155…

```
num_vars = ['area', 'bedrooms', 'bathrooms', 'stories', 'mainroad', 'guestroom', 'ba
df = housing[num_vars]
df.head()
```

Out[155]:

	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airc
0	7420	4	2	3	1	0	0	0	
1	8960	4	4	4	1	0	0	0	
2	9960	3	2	2	1	0	1	0	
3	7500	4	2	2	1	0	1	0	
4	7420	4	1	2	1	1	1	0	

In [156…

```
df.shape
```

Out[156]:

```
(545, 12)
```

In [157…

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
x = df.loc[:, num_vars].values
x = StandardScaler().fit_transform(x)
df.head(5)
```

Out[157]:

	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airc
0	7420	4	2	3	1	0	0	0	
1	8960	4	4	4	1	0	0	0	
2	9960	3	2	2	1	0	1	0	
3	7500	4	2	2	1	0	1	0	
4	7420	4	1	2	1	1	1	0	

In [158…

```
k=1    #设置降维的占比
pca= PCA(n_components=k)#调用PCA函数，先实例化
pcaCom = pca.fit_transform(x)
pcaCom = pd.DataFrame(pcaCom)
print("主成分的数量：",pca.n_components_)
X = pcaCom.iloc[:, [0]].values
#X = pcaCom.iloc[:, [0, 1, 2, 3, 4, 5, 6, 7, 8]].values
Y = df.iloc[:, 11].values
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, train_size=0.8, test_size=
```

```
主成分的数量： 1
```

In [159…

```
from sklearn.svm import SVR

Kernel = ["linear","poly","rbf","sigmoid"]
for kernel in Kernel:
    time0 = time()
    clf= SVR(kernel = kernel
             , C=1e3
             , gamma=″auto″
           ).fit(X_train,Y_train)
    print(″The accuracy under kernel %s is %f″ % (kernel,clf.score(X_test,Y_test)))
    print(datetime.datetime.fromtimestamp(time()-time0).strftime(″%M:%S:%f″))
# Fit regression model
svr_rbf = SVR(kernel=’rbf’, C=1e3, gamma=0.1)
svr_lin = SVR(kernel=’linear’, C=1e3)
svr_poly = SVR (kernel=’poly’, C=1e3, degree=2)
y_rbf = svr_rbf.fit(X_train, Y_train).predict(X_test)
y_lin = svr_lin.fit(X_train, Y_train).predict(X_test)
y_poly = svr_poly.fit(X_train, Y_train).predict(X_test)
```

```
The accuracy under kernel linear is 0.586657
00:00:007734
The accuracy under kernel poly is 0.260309
00:00:007915
The accuracy under kernel rbf is 0.027422
00:00:009862
The accuracy under kernel sigmoid is 0.174441
00:00:009953
```

In [160…

```
print(’精确度：%.4f’%(svr_rbf.score(X_test,Y_test)))
```

```
精确度：0.0389
```

In [161…

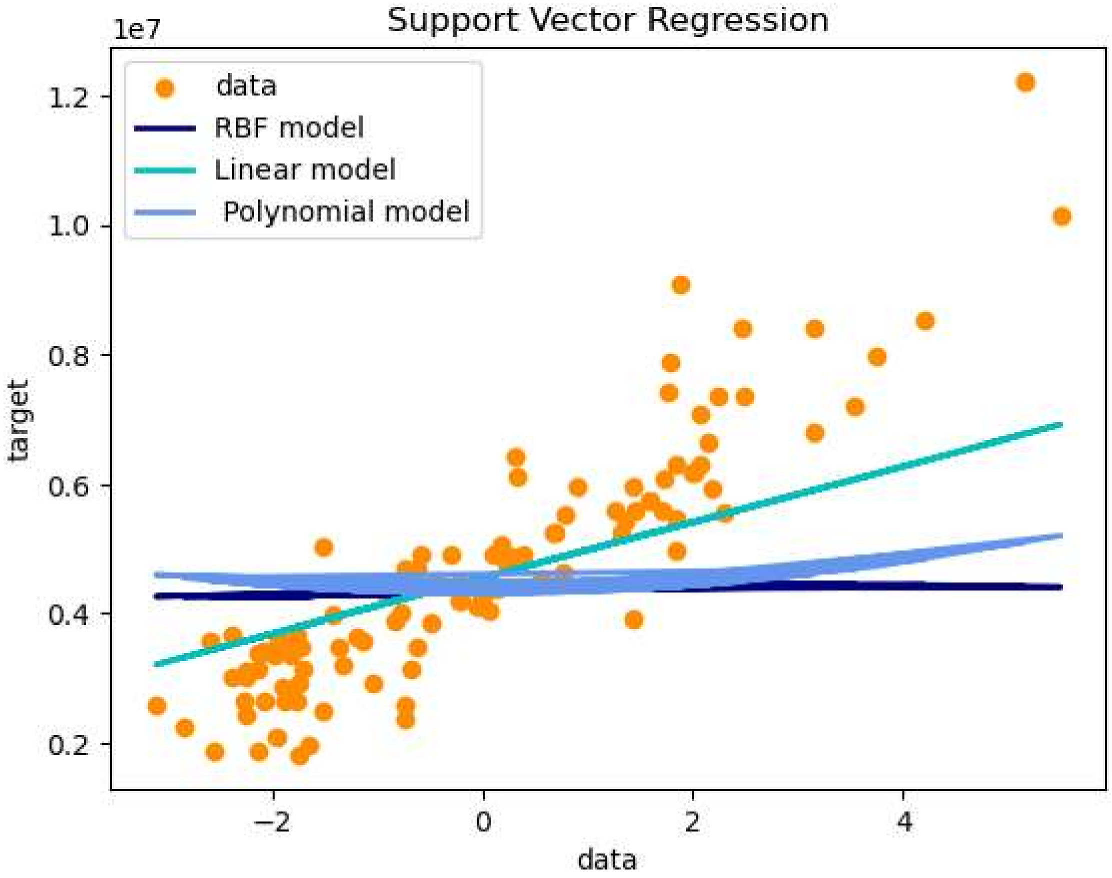
```
#X_train
#X_test
```

In [162…

```
#Y_test
```

In [163…

```
lw = 2
plt.scatter(X_test, Y_test, color=’darkorange’, label=’data’)
plt.plot(X_test, y_rbf, color=’navy’, lw=lw, label=’RBF model’)
plt.plot(X_test, y_lin, color=’c’, lw=lw, label=’Linear model’)
plt.plot(X_test, y_poly, color=’cornflowerblue’, lw=lw, label=’ Polynomial model’)
plt.xlabel(’data’)
plt.ylabel(’target’)
plt.title(’Support Vector Regression’)
plt.legend ()
plt.show()
```


Support Vector Regression
1e7
data
RBF model
Linear model
Polynomial model
1.2
1.0
0.8
0.6
0.4
0.2
target
−2
0
2
4
data