In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,classification_report,confusion_mat
from matplotlib.colors import ListedColormap
from sklearn.svm import SVC
```

In [2]:

```
a=pd.read_csv('recipes_muffins_cupcakes.csv')
a
```

Out[2]:

	Type	Flour	Milk	Sugar	Butter	Egg	Baking Powder	Vanilla	Salt
0	Muffin	55	28	3	7	5	2	0	0
1	Muffin	47	24	12	6	9	1	0	0
2	Muffin	47	23	18	6	4	1	0	0
3	Muffin	45	11	17	17	8	1	0	0
4	Muffin	50	25	12	6	5	2	1	0
5	Muffin	55	27	3	7	5	2	1	0
6	Muffin	54	27	7	5	5	2	0	0
7	Muffin	47	26	10	10	4	1	0	0
8	Muffin	50	17	17	8	6	1	0	0
9	Muffin	50	17	17	11	4	1	0	0
10	Cupcake	39	0	26	19	14	1	1	0
11	Cupcake	42	21	16	10	8	3	0	0
12	Cupcake	34	17	20	20	5	2	1	0
13	Cupcake	39	13	17	19	10	1	1	0
14	Cupcake	38	15	23	15	8	0	1	0
15	Cupcake	42	18	25	9	5	1	0	0
16	Cupcake	36	14	21	14	11	2	1	0
17	Cupcake	38	15	31	8	6	1	1	0
18	Cupcake	36	16	24	12	9	1	1	0
19	Cupcake	34	17	23	11	13	0	1	0

In [3]:

```
a.head()
```

Out[3]:

	Type	Flour	Milk	Sugar	Butter	Egg	Baking Powder	Vanilla	Salt
0	Muffin	55	28	3	7	5	2	0	0
1	Muffin	47	24	12	6	9	1	0	0
2	Muffin	47	23	18	6	4	1	0	0
3	Muffin	45	11	17	17	8	1	0	0
4	Muffin	50	25	12	6	5	2	1	0

In [4]:

```
a.tail()
```

Out[4]:

	Type	Flour	Milk	Sugar	Butter	Egg	Baking Powder	Vanilla	Salt
15	Cupcake	42	18	25	9	5	1	0	0
16	Cupcake	36	14	21	14	11	2	1	0
17	Cupcake	38	15	31	8	6	1	1	0
18	Cupcake	36	16	24	12	9	1	1	0
19	Cupcake	34	17	23	11	13	0	1	0

In [5]:

```
X=a[['Sugar','Flour']].values
Y=a['Type'].values
```

In [13]:

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
model=SVC(kernel='linear',random_state=42)
model.fit(X_train,Y_train)
y_pred=model.predict(X_test)
```

In [7]:

```
accuracy=accuracy_score(Y_test,y_pred)
accuracy
```

Out[7]:

```
1.0
```

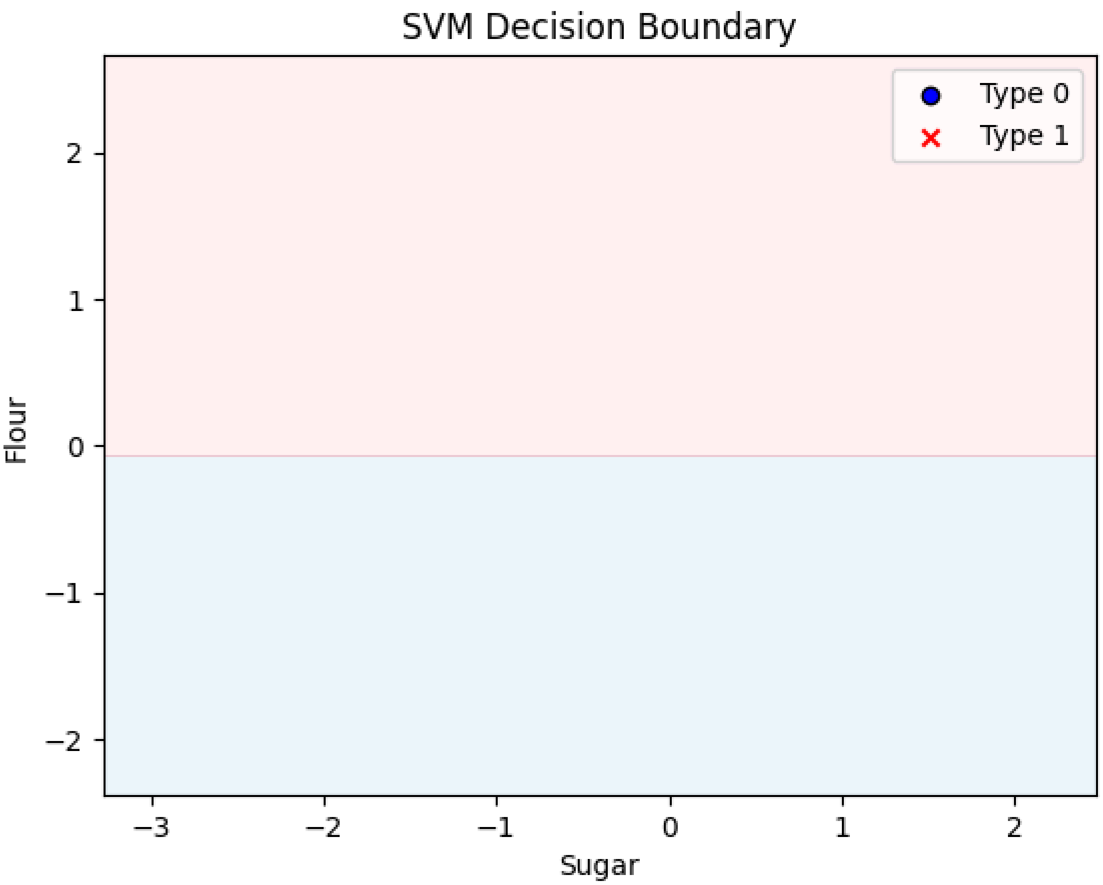
In [8]:

```
cr=classification_report(Y_test,y_pred)
print('\n Classification Report \n',cr)
```

```
Classification Report
              precision    recall  f1-score   support

     Cupcake       1.00      1.00      1.00         2
      Muffin       1.00      1.00      1.00         2

    accuracy                           1.00         4
   macro avg       1.00      1.00      1.00         4
weighted avg       1.00      1.00      1.00         4
```

In [14]:

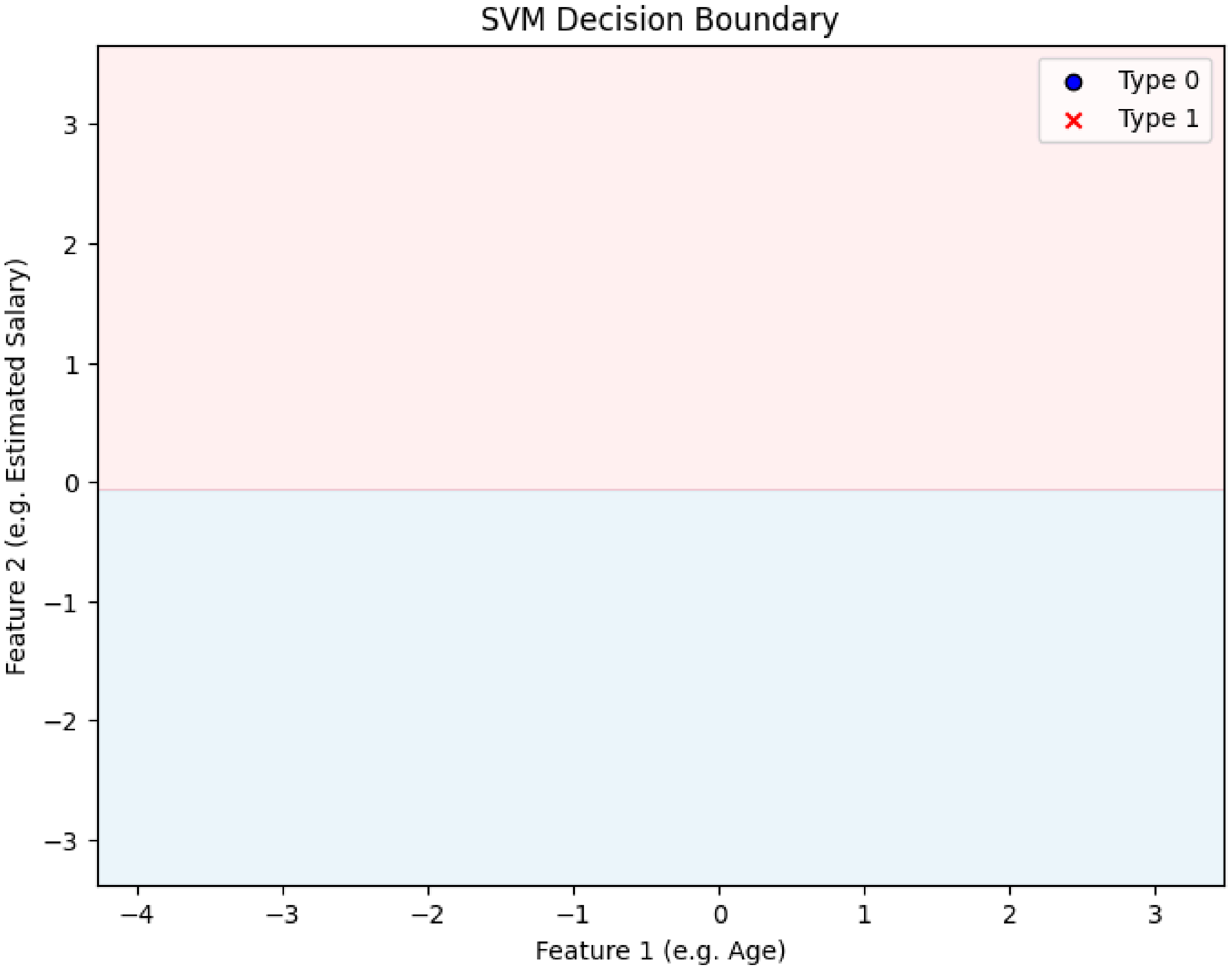
```
X1,X2=np.meshgrid(np.arange(start=X_train[:,0].min()-1,stop=X_train[:,0].max()
                  np.arange(start=X_train[:,1].min()-1,stop=X_train[:,1].max()
Z=model.predict(np.array([X1.ravel(),X2.ravel()]).T)
if not np.issubdtype(np.array(Z).dtype, np.number):
    le = LabelEncoder()
    Z = le.fit_transform(Z)

# Reshape back into grid shape
Z = Z.reshape(X1.shape).astype(float)
#Z=Z.reshape(X1.shape)
plt.contourf(X1,X2,Z,alpha=0.2,cmap=ListedColormap(('lightblue','lightpink')))
plt.scatter(X_train[Y_train == 0, 0], X_train[Y_train == 0, 1],
            marker='o', color='blue', label='Type 0', edgecolor='k')
plt.scatter(X_train[Y_train == 1, 0], X_train[Y_train == 1, 1],
            marker='x', color='red', label='Type 1')
plt.title('SVM Decision Boundary')
plt.xlabel('Sugar')
plt.ylabel('Flour')
plt.legend()
plt.show()
```



In [16]:

```
X1, X2 = np.meshgrid(
    np.arange(start=X_train[:, 0].min() - 2, stop=X_train[:, 0].max() + 2, ste
    np.arange(start=X_train[:, 1].min() - 2, stop=X_train[:, 1].max() + 2, ste
)

# Predict over grid points
Z = model.predict(np.array([X1.ravel(), X2.ravel()]).T)

# If predictions are non-numeric, convert them
if not np.issubdtype(np.array(Z).dtype, np.number):
    le = LabelEncoder()
    Z = le.fit_transform(Z)

# Reshape into grid shape for contour plotting
Z = Z.reshape(X1.shape).astype(float)

# Plot decision boundary
plt.figure(figsize=(8,6))
plt.contourf(X1, X2, Z, alpha=0.2, cmap=ListedColormap(('lightblue', 'lightpir

# Plot training points
plt.scatter(X_train[Y_train == 0, 0], X_train[Y_train == 0, 1],
            marker='o', color='blue', label='Type 0', edgecolor='k')
plt.scatter(X_train[Y_train == 1, 0], X_train[Y_train == 1, 1],
            marker='x', color='red', label='Type 1')
```

```
# Add labels and title
plt.title('SVM Decision Boundary')
plt.xlabel('Feature 1 (e.g. Age)')
plt.ylabel('Feature 2 (e.g. Estimated Salary)')
plt.legend()
plt.show()
```



In [ ]: