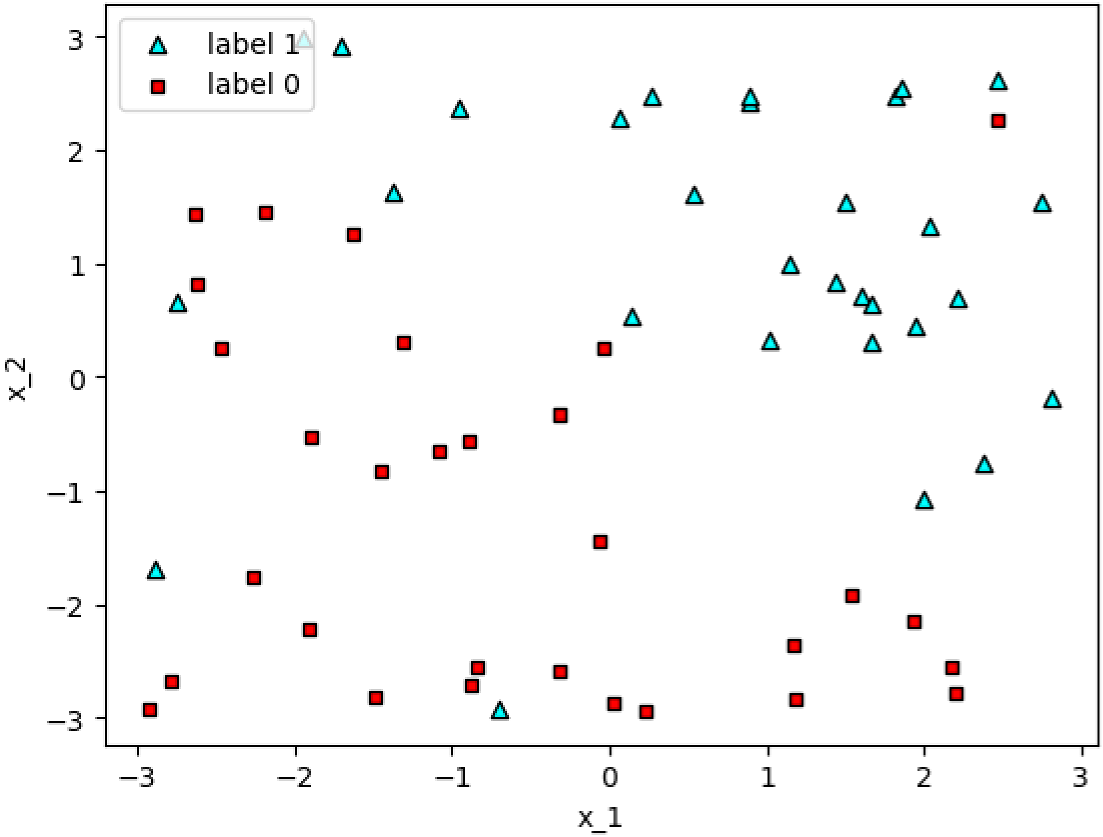
In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import utils
from sklearn.svm import SVC
```

Building an SVM to separate a linear dataset

In [ ]:

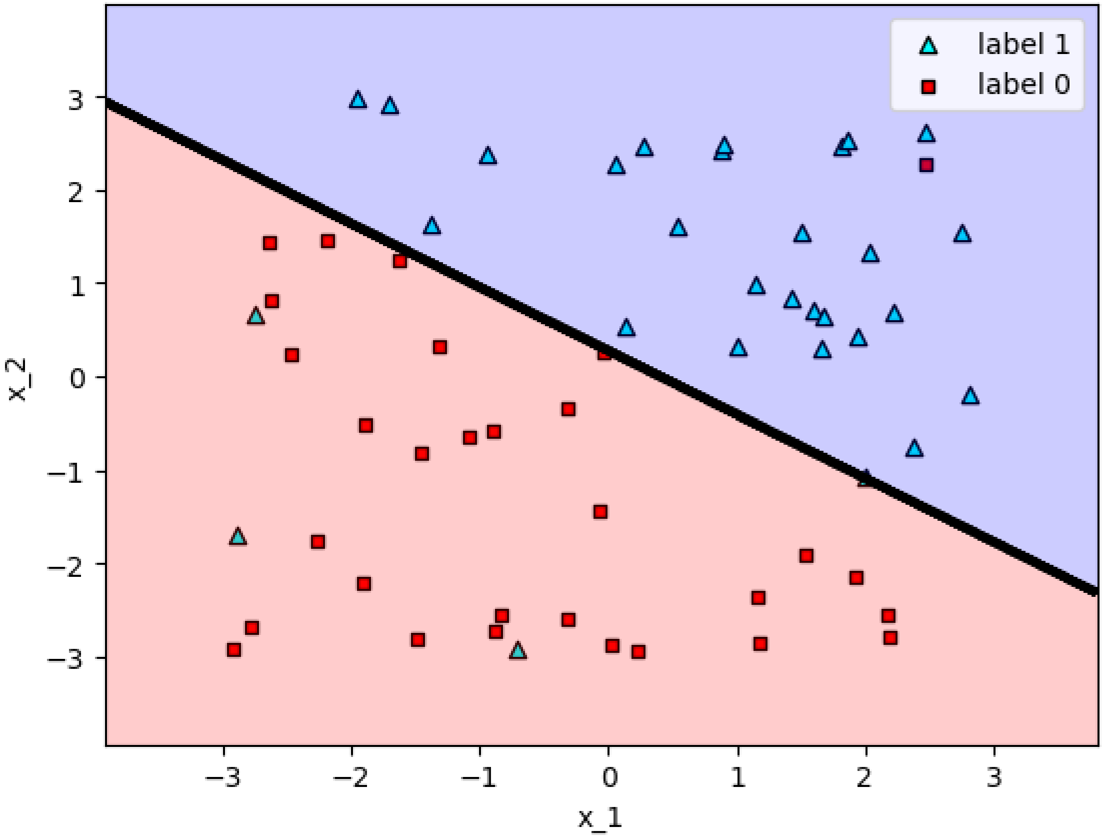
```
# Loading the linear dataset

linear_data = pd.read_csv('linear.csv')
features = np.array(linear_data[['x_1', 'x_2']])
labels = np.array(linear_data['y'])
utils.plot_points(features, labels)
```



In [ ]:

```
svm_linear = SVC(kernel='linear')
svm_linear.fit(features, labels)
print("Accuracy:", svm_linear.score(features, labels))
utils.plot_model(features, labels, svm_linear)
```

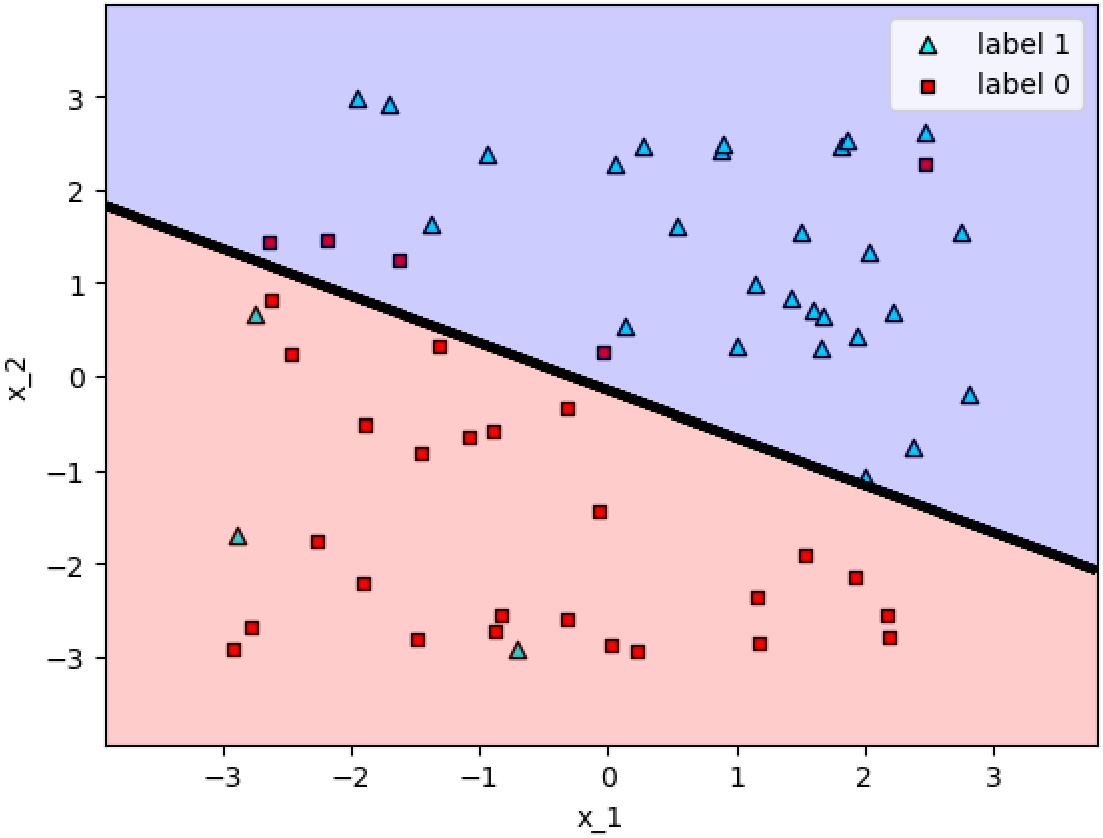
```
Accuracy: 0.9333333333333333
```



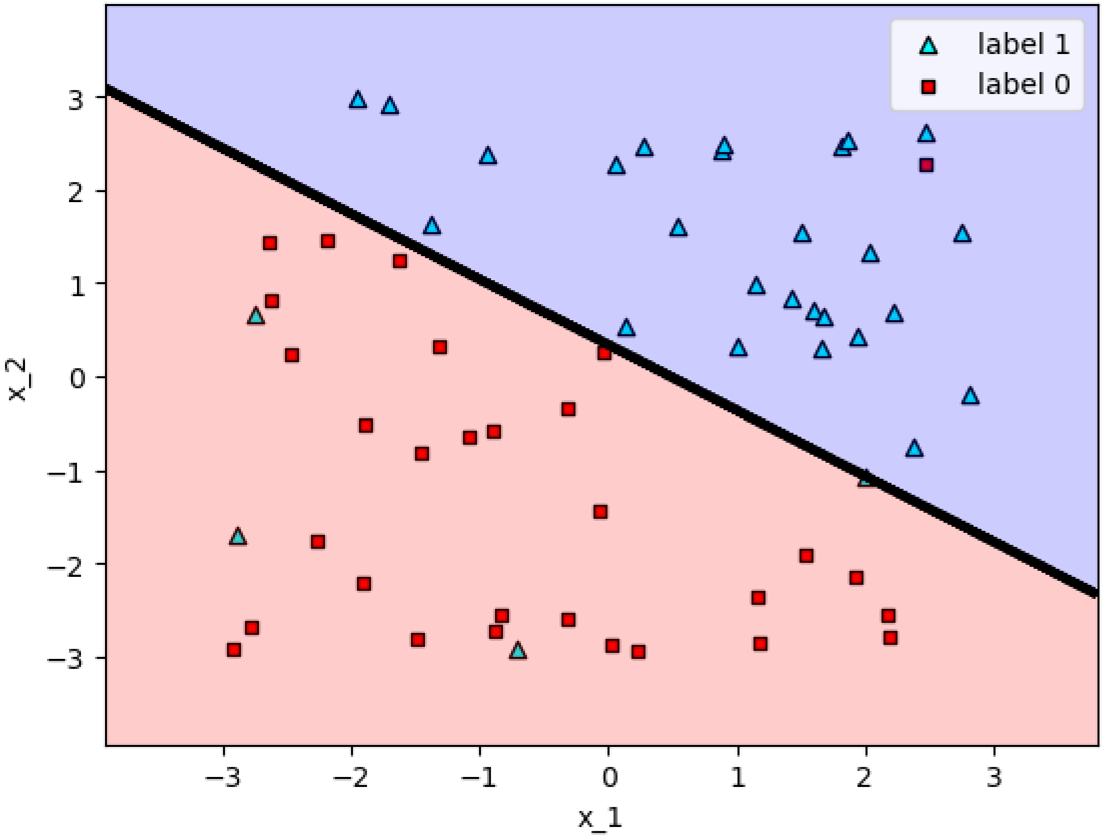
In [ ]:

```
# C = 0.01
svm_c_001 = SVC(kernel='linear', C=0.01)
svm_c_001.fit(features, labels)
print("C = 0.01")
print("Accuracy:", svm_c_001.score(features, labels))
utils.plot_model(features, labels, svm_c_001)

# C = 100
svm_c_100 = SVC(kernel='linear', C=100)
svm_c_100.fit(features, labels)
print("C = 100")
print("Accuracy:", svm_c_100.score(features, labels))
utils.plot_model(features, labels, svm_c_100)
```

```
C = 0.01
Accuracy: 0.8666666666666667
```



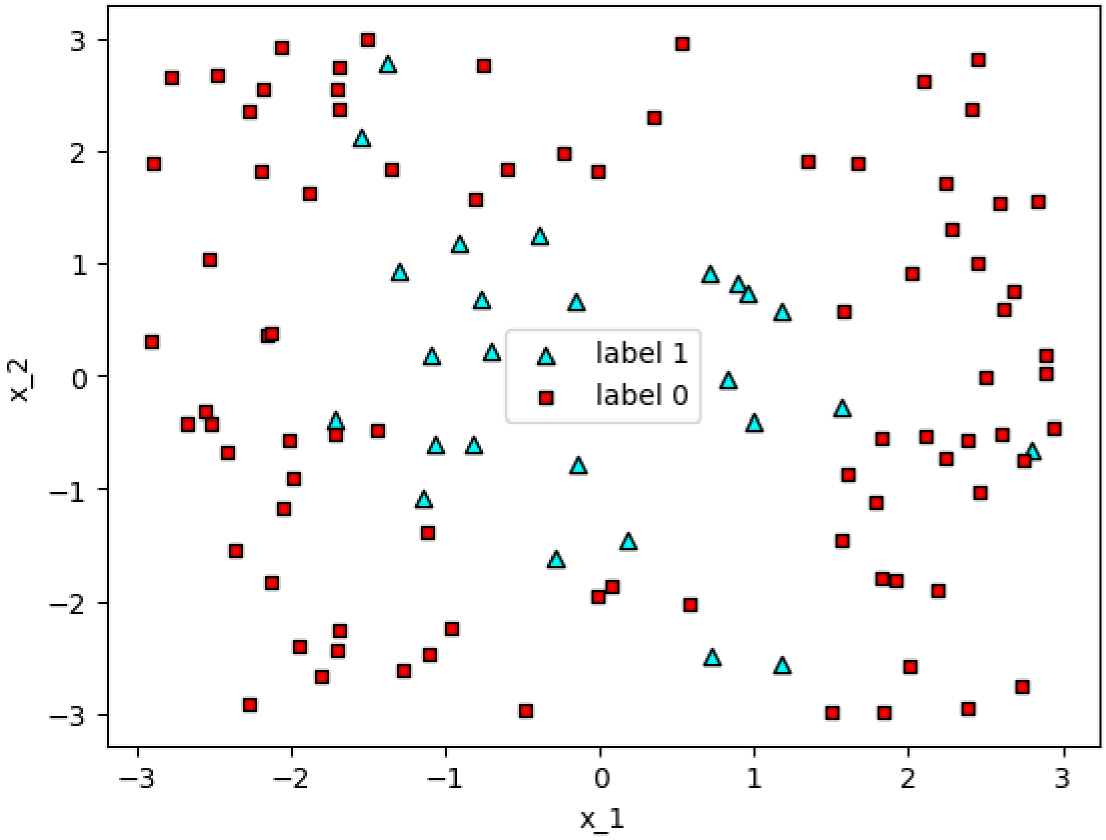
```
C = 100
Accuracy: 0.9166666666666666
```



Building polynomial kernels for a circular dataset

In [ ]:

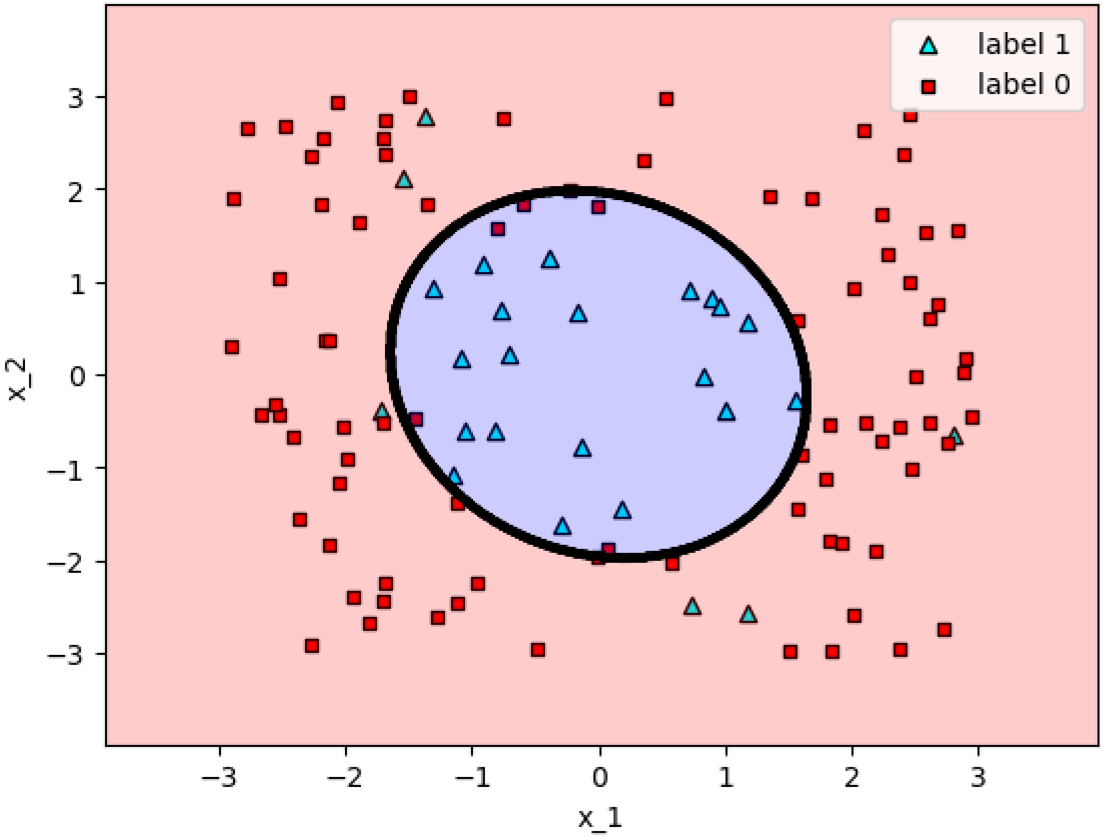
```
# Loading the one_circle dataset

circular_data = pd.read_csv('one_circle.csv')
features = np.array(circular_data[['x_1', 'x_2']])
labels = np.array(circular_data['y'])
utils.plot_points(features, labels)
```



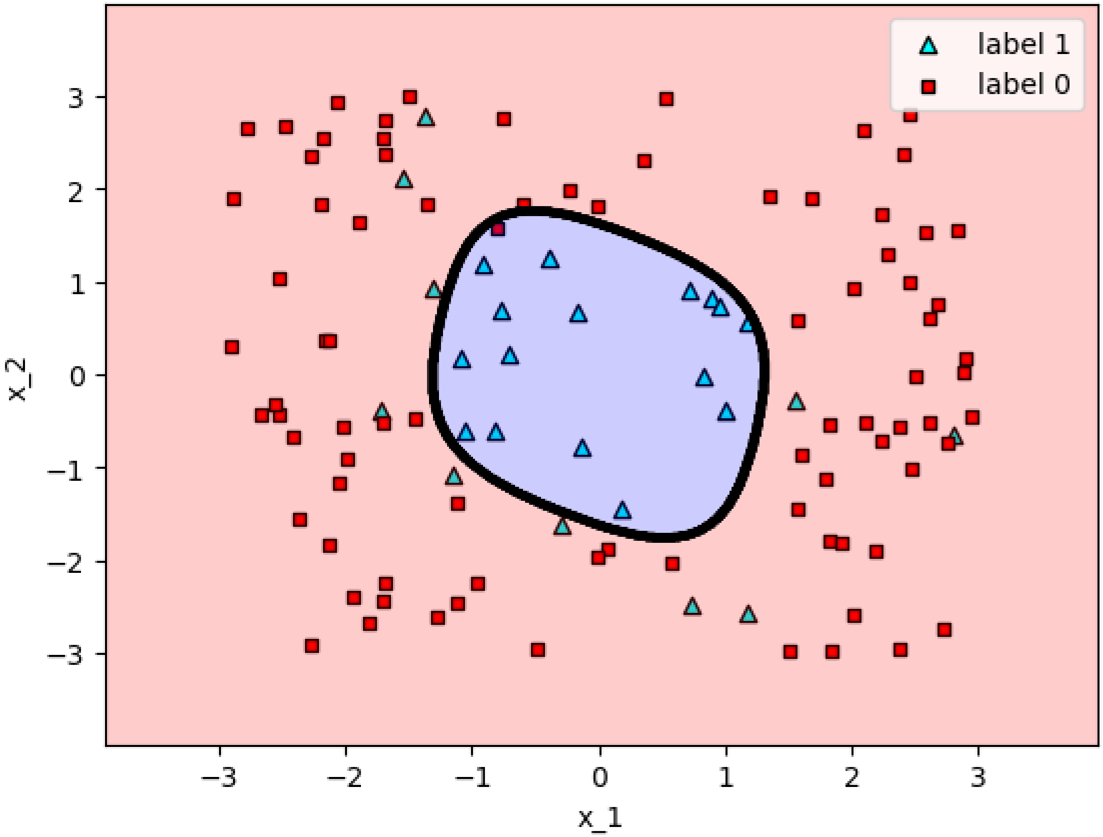
In [ ]:

```
# Degree = 2
svm_degree_2 = SVC(kernel='poly', degree=2)
svm_degree_2.fit(features, labels)
print("Polynomial kernel of degree = 2")
print("Accuracy:", svm_degree_2.score(features, labels))
utils.plot_model(features, labels, svm_degree_2)

# Degree = 4
svm_degree_4 = SVC(kernel='poly', degree=4)
svm_degree_4.fit(features, labels)
print("Polynomial kernel of degree = 4")
print("Accuracy:", svm_degree_4.score(features, labels))
utils.plot_model(features, labels, svm_degree_4)
```

```
Polynomial kernel of degree = 2
Accuracy: 0.8909090909090909
```



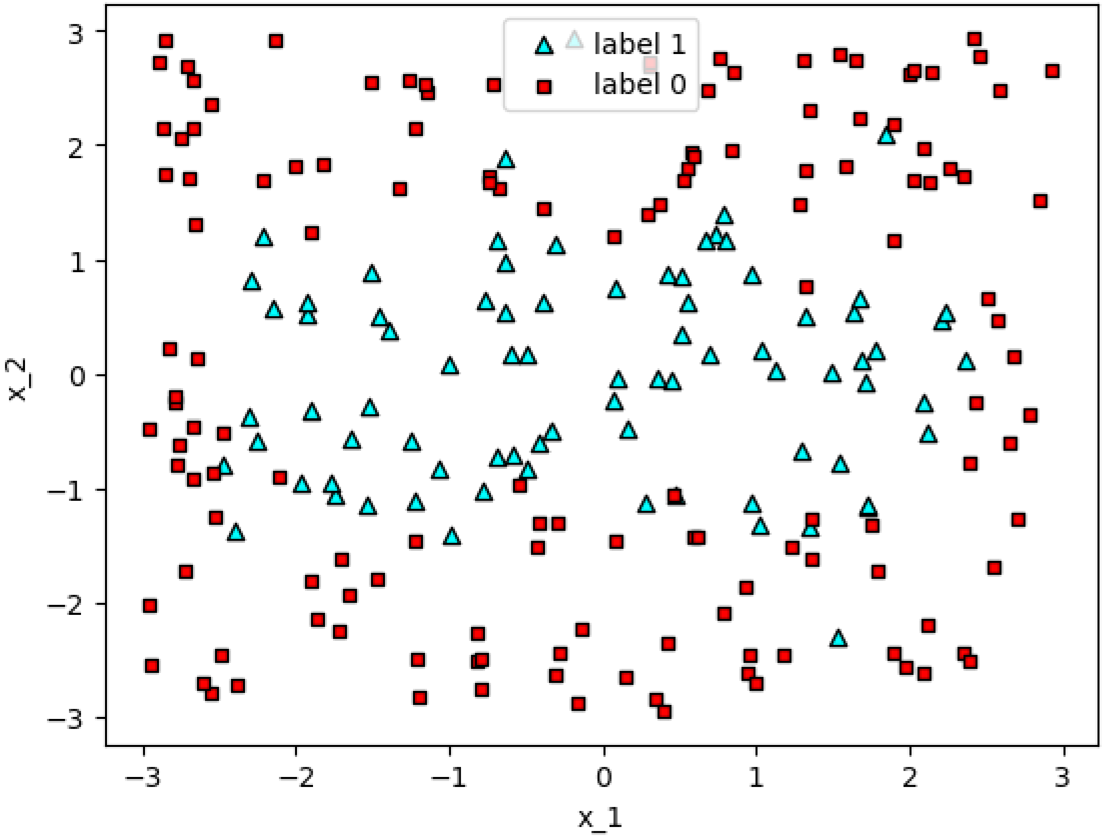
```
Polynomial kernel of degree = 4
Accuracy: 0.9
```



Experimenting with gammas in the rbf kernel

In [ ]:

```
# Loading the two_circles dataset

two_circles_data = pd.read_csv('two_circles.csv')
features = np.array(two_circles_data[['x_1', 'x_2']])
labels = np.array(two_circles_data['y'])
utils.plot_points(features, labels)
```

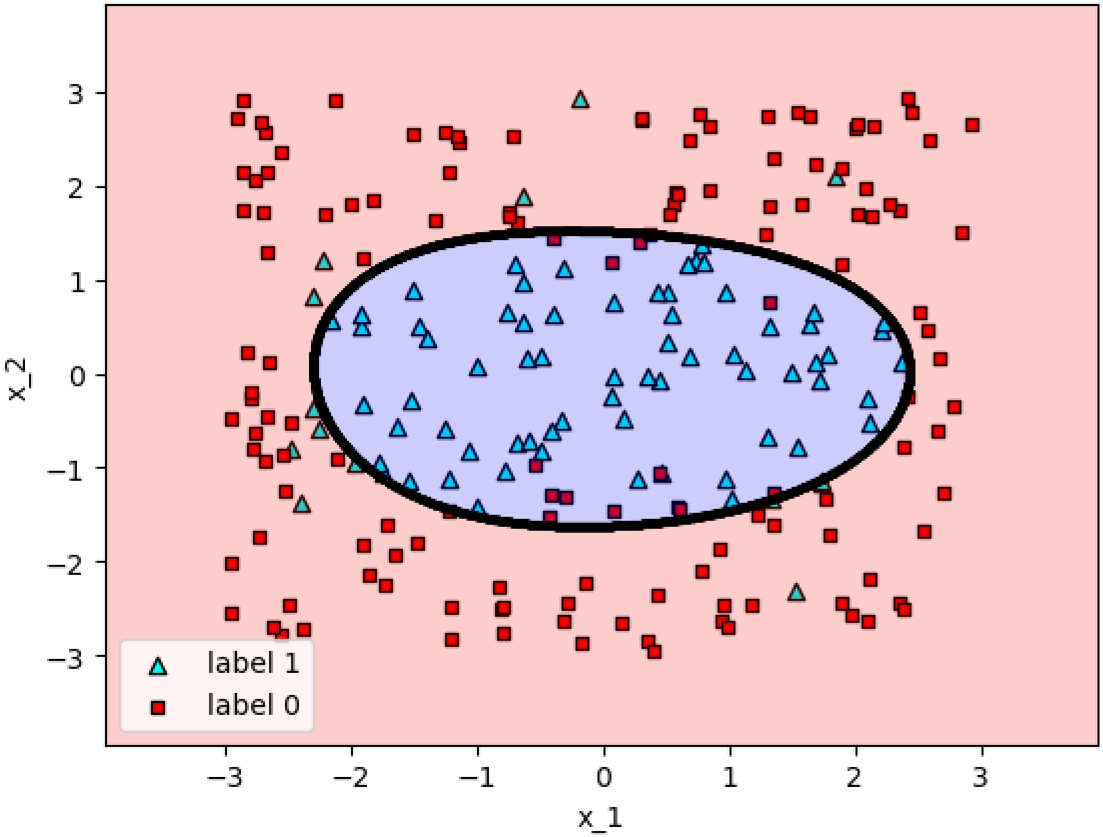


In [ ]:

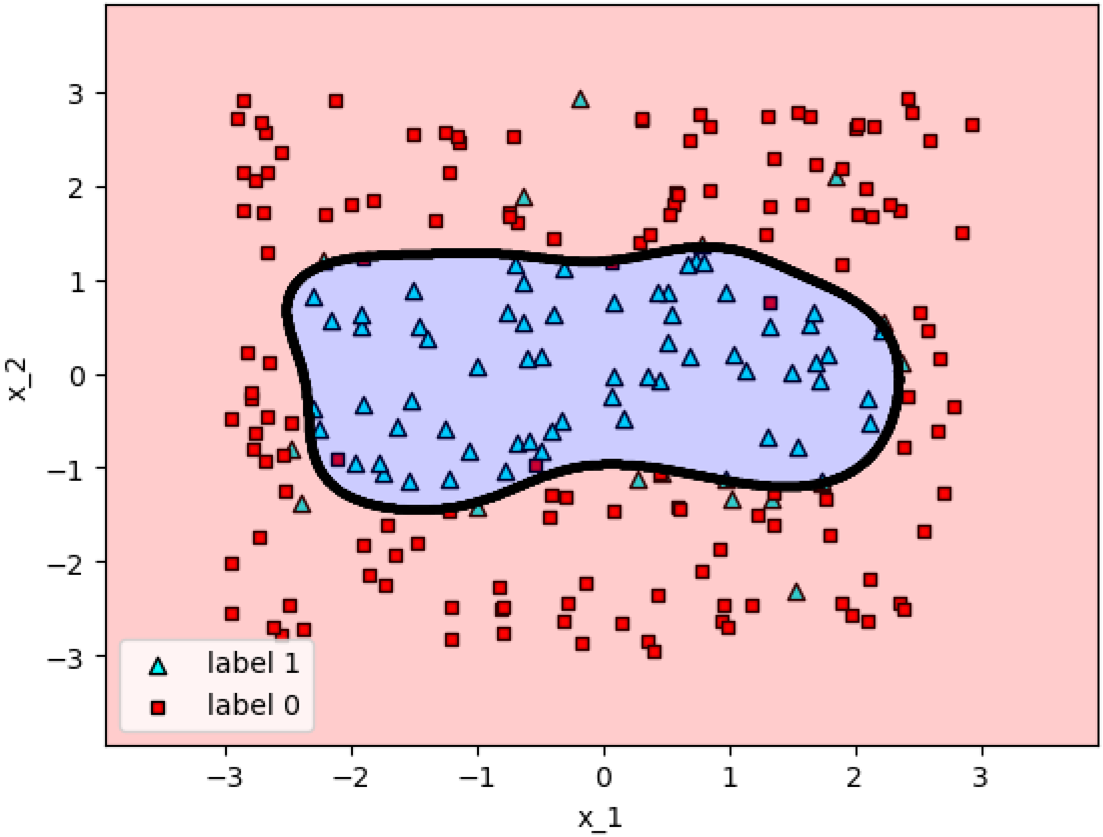
```
# gamma = 0.1
svm_gamma_01 = SVC(kernel='rbf', gamma=0.1)
svm_gamma_01.fit(features, labels)
print("Gamma = 0.1")
print("Accuracy:", svm_gamma_01.score(features, labels))
utils.plot_model(features, labels, svm_gamma_01)

# gamma = 1
svm_gamma_1 = SVC(kernel='rbf', gamma=1)
svm_gamma_1.fit(features, labels)
print("Gamma = 1")
print("Accuracy:", svm_gamma_1.score(features, labels))
utils.plot_model(features, labels, svm_gamma_1)

# gamma = 10
svm_gamma_10 = SVC(kernel='rbf', gamma=10)
svm_gamma_10.fit(features, labels)
print("Gamma = 10")
print("Accuracy:", svm_gamma_10.score(features, labels))
```

```
utils.plot_model(features, labels, svm_gamma_10)

# gamma = 100
svm_gamma_100 = SVC(kernel='rbf', gamma=100)
svm_gamma_100.fit(features, labels)
print("Gamma = 100")
print("Accuracy:", svm_gamma_100.score(features, labels))
utils.plot_model(features, labels, svm_gamma_100)
```

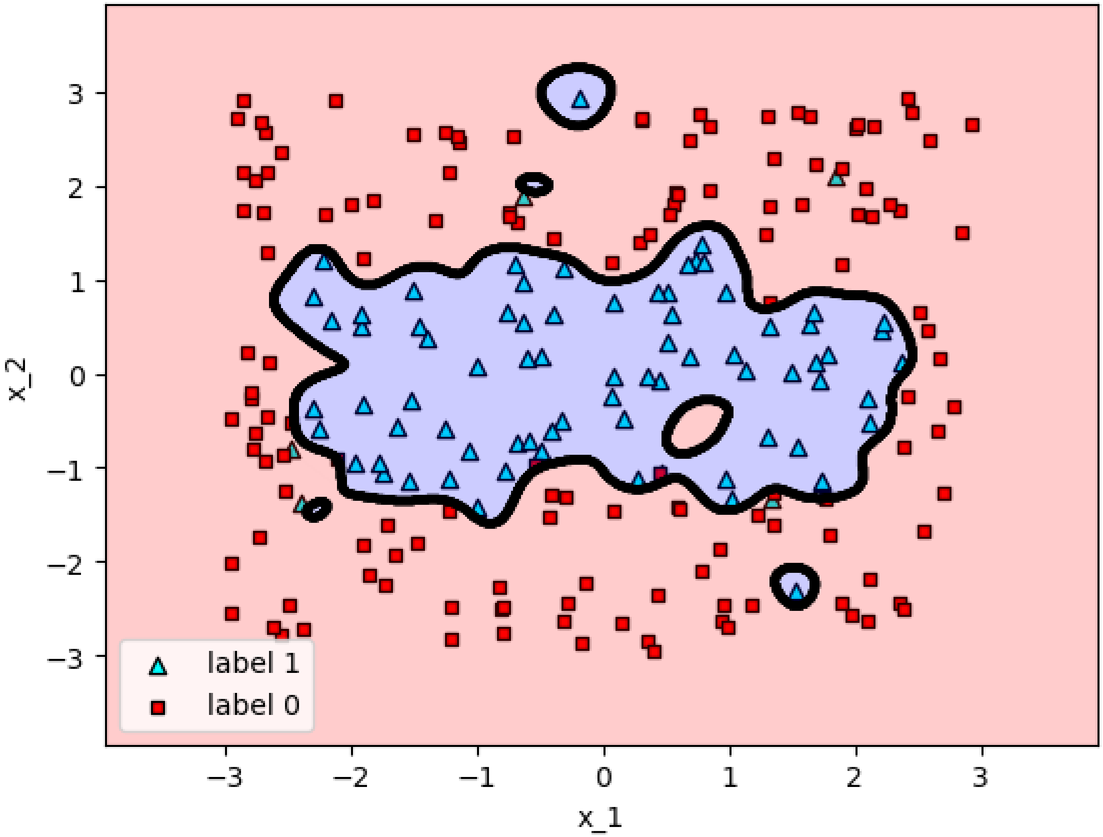
```
Gamma = 0.1
Accuracy: 0.8772727272727273
```



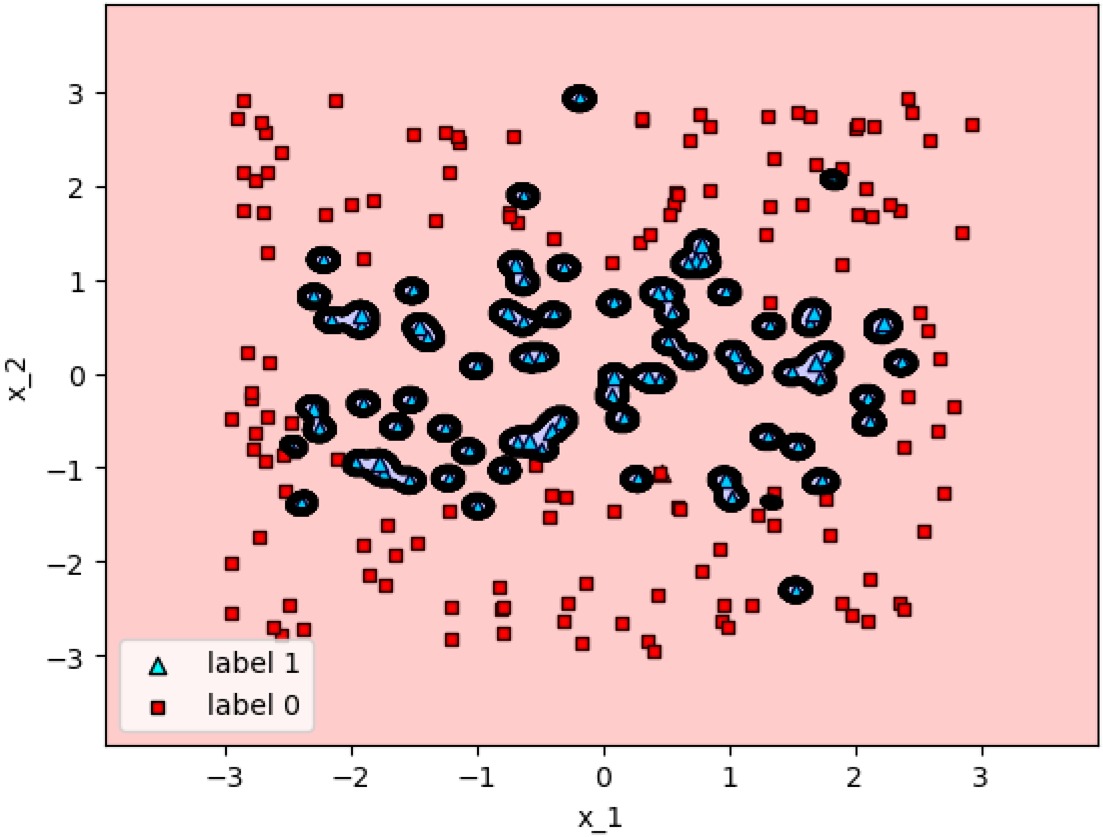
```
Gamma = 1
Accuracy: 0.9045454545454545
```



```
Gamma = 10
Accuracy: 0.9636363636363636
```



```
Gamma = 100
Accuracy: 0.990909090909091
```



In [ ]:

```
# All accuracies are training accuracies
# Compute the cross-validation accuracy for each model
```