Untitled

October 18, 2023

1. Use the IRIS dataset, implement the SVM classifier in python (make use of scikit-learn library), to do the following.

a. Apply the kernel functions such as linear, polynomial, Radial basis functions and Sigmoid.
b. Plot the scatter plot of the input features.
c. Plot the decision boundary.

[42]:
```
import matplotlib.pyplot as plt
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.svm import SVC
```

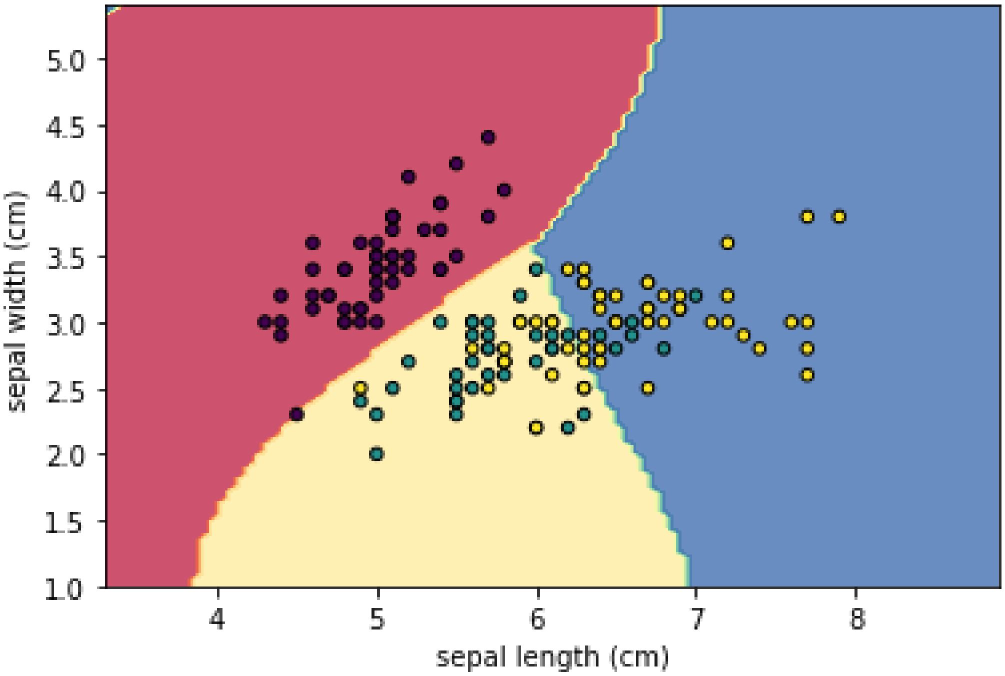
[43]:
```
from sklearn.datasets import load_iris
iris = load_iris()
# Store features matrix in X
X= iris.data[:,:2]#Store target vector in
y= iris.target
```

[44]:
```
print(iris.feature_names)
```

```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width
(cm)']
```

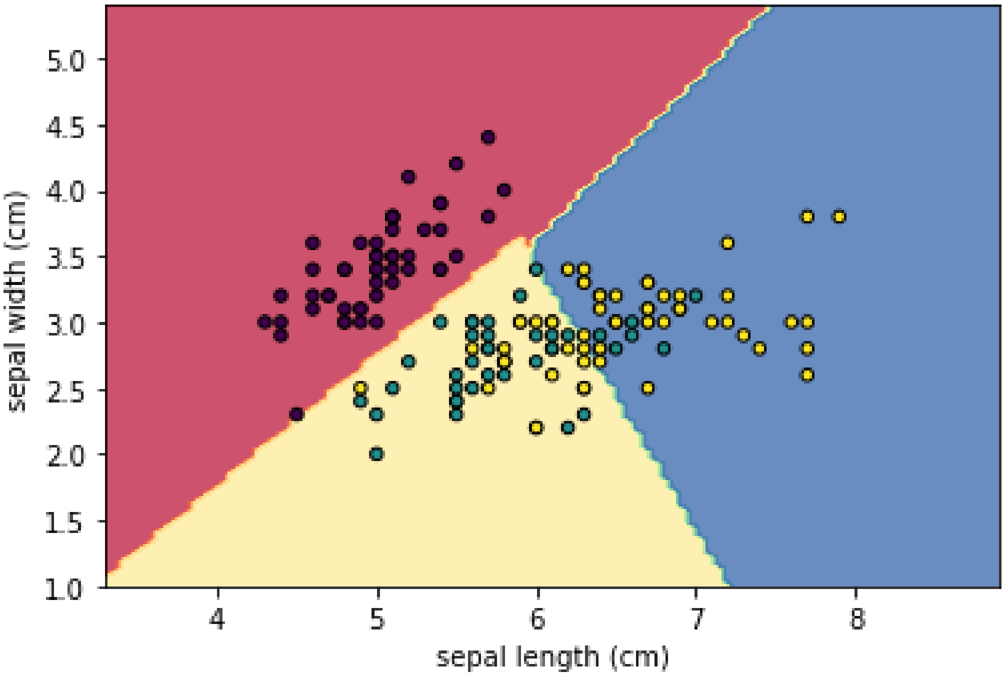
[45]:
```
svm = SVC(kernel="rbf", gamma=0.5, C=1.0)
```

[46]:
```
svm.fit(X, y)
DecisionBoundaryDisplay.from_estimator(
svm,
X,
response_method="predict",
cmap=plt.cm.Spectral,
alpha=0.8,
xlabel=iris.feature_names[0],
ylabel=iris.feature_names[1],
)
plt.scatter(X[:, 0], X[:, 1],
c=y,
s=20, edgecolors="k")
plt.show()
```



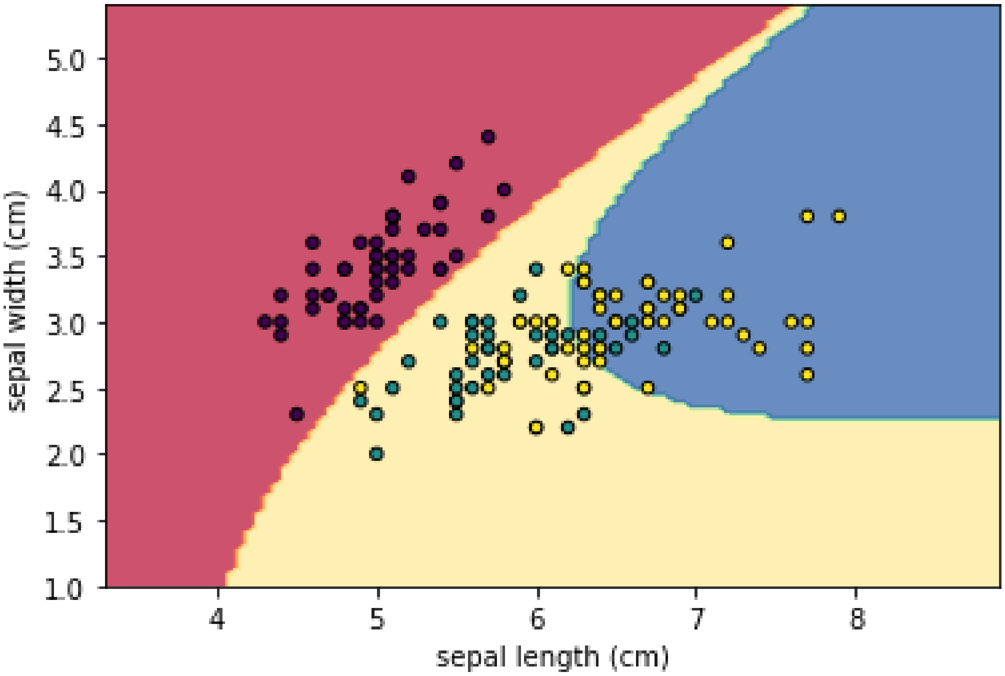
[47]:

```
svm = SVC(kernel="linear", gamma=0.5, C=1.0)
svm.fit(X, y)
DecisionBoundaryDisplay.from_estimator(
svm,
X,
response_method="predict",
cmap=plt.cm.Spectral,
alpha=0.8,
xlabel=iris.feature_names[0],
ylabel=iris.feature_names[1],
)
plt.scatter(X[:, 0], X[:, 1],
c=y,
s=20, edgecolors="k")
plt.show()
```



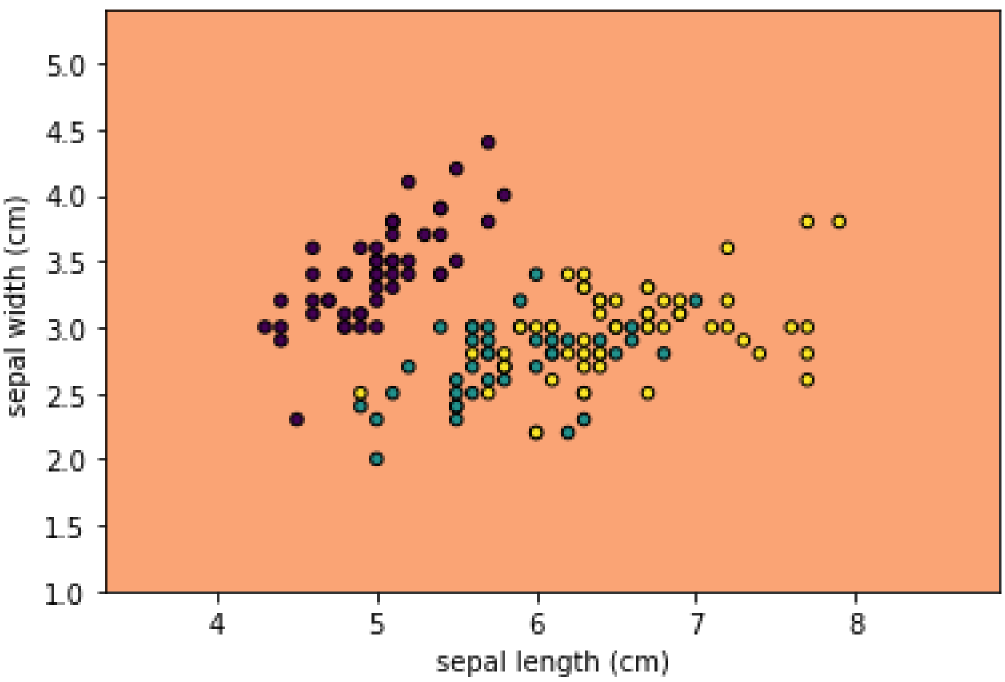
[48]:

```
svm = SVC(kernel="poly", gamma=0.5, C=1.0)
svm.fit(X, y)
DecisionBoundaryDisplay.from_estimator(
svm,
X,
response_method="predict",
cmap=plt.cm.Spectral,
alpha=0.8,
xlabel=iris.feature_names[0],
ylabel=iris.feature_names[1],
)
plt.scatter(X[:, 0], X[:, 1],
c=y,
s=20, edgecolors="k")
plt.show()
```



[49]:

```
svm = SVC(kernel="sigmoid", gamma=0.5, C=1.0)
svm.fit(X, y)
DecisionBoundaryDisplay.from_estimator(
svm,
X,
response_method="predict",
cmap=plt.cm.Spectral,
alpha=0.8,
xlabel=iris.feature_names[0],
ylabel=iris.feature_names[1],
)
plt.scatter(X[:, 0], X[:, 1],
c=y,
s=20, edgecolors="k")
plt.show()
```



[52]:
```
'''kernels = ['rbf', 'linear', 'poly', 'sigmoid']

for kernel in kernels:
    svm = SVC(kernel=kernel, gamma=0.5, C=1.0)
    svm.fit(X, y)
    DecisionBoundaryDisplay.from_estimator(
    svm,
    X,
    response_method="predict",
    cmap=plt.cm.Spectral,
    alpha=0.8,
    xlabel=iris.feature_names[0],
    ylabel=iris.feature_names[1],
    )
    plt.scatter(X[:, 0], X[:, 1],
    c=y,
    s=20, edgecolors="k")
    plt.title(kernel)
    plt.show()'''
```

[52]:
```
'kernels = [\'rbf\', \'linear\', \'poly\', \'sigmoid\']\n\nfor kernel in
kernels:\n    svm = SVC(kernel=kernel, gamma=0.5, C=1.0)\n    svm.fit(X, y)\n
DecisionBoundaryDisplay.from_estimator(\n    svm,\n    X,\n
response_method="predict",\n    cmap=plt.cm.Spectral,\n    alpha=0.8,\n
xlabel=iris.feature_names[0],\n    ylabel=iris.feature_names[1],\n    )\n
```

```
plt.scatter(X[:, 0], X[:, 1],\n    c=y,\n    s=20, edgecolors="k")\n
plt.title(kernel)\n    plt.show()'
```

[ ]:

```

```