Ishan Gupta - SVM (1 vs All) - 19BCE7467

```
[10] from sklearn.metrics import confusion_matrix
     from sklearn.model_selection import train_test_split
     from sklearn import svm, datasets
     import matplotlib.pyplot as plt
     import numpy as np
```

```
[11] iris = datasets.load_iris()
     X = iris.data[:, :2]
     y = iris.target
```

```
[12] X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, random_state = 0)
```

```
[13] linear = svm.SVC(kernel='linear', C=1, decision_function_shape='ovo').fit(X_train, y_train)
     rbf = svm.SVC(kernel='rbf', gamma=1, C=1, decision_function_shape='ovo').fit(X_train, y_train)
     poly = svm.SVC(kernel='poly', degree=3, C=1, decision_function_shape='ovo').fit(X_train, y_train)
     sig = svm.SVC(kernel='sigmoid', C=1, decision_function_shape='ovo').fit(X_train, y_train)
```

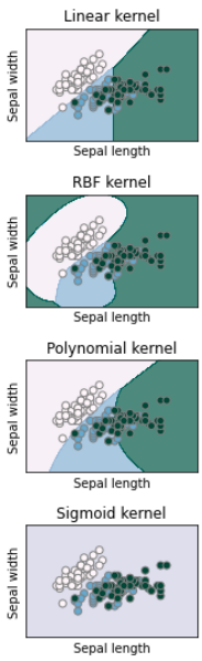
```
[14] h = .01

     x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
     y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1

     xx, yy = np.meshgrid(np.arange(x_min, x_max, h),np.arange(y_min, y_max, h))
     titles = ['Linear kernel','RBF kernel','Polynomial kernel','Sigmoid kernel']
```

```
[15] for i, clf in enumerate((linear, rbf, poly, sig)):
         plt.subplot(2, 2, i + 1)
         plt.subplots_adjust(wspace=0.4, hspace=0.4)
         Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
         Z = Z.reshape(xx.shape)
         plt.contourf(xx, yy, Z, cmap=plt.cm.PuBuGn, alpha=0.7)
         plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.PuBuGn,     edgecolors='grey')
         plt.xlabel('Sepal length')
         plt.ylabel('Sepal width')
         plt.xlim(xx.min(), xx.max())
         plt.ylim(yy.min(), yy.max())
         plt.xticks(())
         plt.yticks(())
         plt.title(titles[i])
         plt.show()
```

[15]




```
[16] linear_pred = linear.predict(X_test)
     poly_pred = poly.predict(X_test)
     rbf_pred = rbf.predict(X_test)
     sig_pred = sig.predict(X_test)
```

```
[17] accuracy_lin = linear.score(X_test, y_test)
     accuracy_poly = poly.score(X_test, y_test)
     accuracy_rbf = rbf.score(X_test, y_test)
     accuracy_sig = sig.score(X_test, y_test)
     print("Accuracy Linear Kernel:", accuracy_lin)
     print("Accuracy Polynomial Kernel:", accuracy_poly)
     print("Accuracy Radial Basis Kernel:", accuracy_rbf)
     print("Accuracy Sigmoid Kernel:", accuracy_sig)
```

```
Accuracy Linear Kernel: 0.7333333333333333
Accuracy Polynomial Kernel: 0.7333333333333333
Accuracy Radial Basis Kernel: 0.6666666666666666
Accuracy Sigmoid Kernel: 0.2
```

```
[18] # creating a confusion matrix
     cm_lin = confusion_matrix(y_test, linear_pred)
     cm_poly = confusion_matrix(y_test, poly_pred)
     cm_rbf = confusion_matrix(y_test, rbf_pred)
     cm_sig = confusion_matrix(y_test, sig_pred)
     print(cm_lin)
     print(cm_poly)
     print(cm_rbf)
     print(cm_sig)
```

```
[[11  0  0]
 [ 0  8  5]
 [ 0  3  3]]
[[11  0  0]
 [ 0  8  5]
 [ 0  3  3]]
[[11  0  0]
 [ 0  5  8]
 [ 0  2  4]]
[[ 0  0 11]
 [ 0  0 13]
 [ 0  0  6]]
```