The programming assignment for the Support Vector Machine model is

Using the Scikit-Learn Library train the SVM model on the complete Iris data set using all of the features at once.

Run the SVM model (at least) four times using a different kernel each time.

Compare the results for each of the kernels.

Plot the results (including the decision boundaries) using two features at a time (since you're limited to two dimension in plotting) for three of the cases (i.e., for three of the kernels). Note that you need to fit the model using all four of the features but then plot the results using only two of the features.

Discuss the pros and cons of using each of the kernels that you've chosen.

loading plot_decision_regions() function

```
#@title loading plot_decision_regions() function
from matplotlib.colors import ListedColormap
import matplotlib.pyplot as plt
def plot_decision_regions(X, y, classifier, test_idx=None, resolution=0.02):
    # setup marker generator and color map
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])
    # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
    np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.3, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())
    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
                    alpha=0.8, c=colors[idx],
                    marker=markers[idx], label=cl,
                    edgecolor='black')
    # highlight test samples
    if test_idx:
    # plot all samples
        X_test, y_test = X[test_idx, :], y[test_idx]
        plt.scatter(X_test[:, 0], X_test[:, 1],
                    c='', edgecolor='black', alpha=1.0,
                    linewidth=1, marker='o',
                    s=100, label='test set')
```

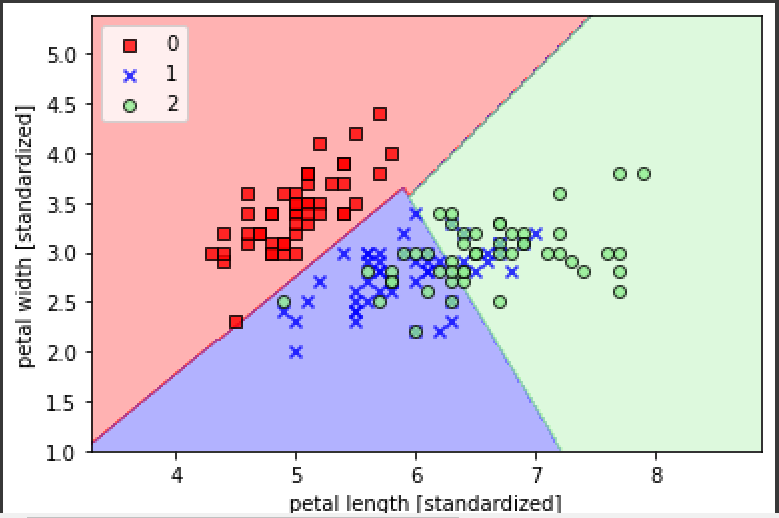
loading Iris dataset and standardizing with all four features

```
#@title loading Iris dataset and standardizing with all four features

from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [0, 1, 2, 3]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)



#Creating values to plot
plot_X = iris.data[:, [0, 1]]
plot_y = iris.target
```

Kernel One: Linear

```
#@title Kernel One: Linear
import sklearn
from sklearn.svm import SVC
import matplotlib.pyplot as plt
import numpy as np

# Run model on four features, print prediction metric
svm1 = SVC(kernel='linear', random_state=1, gamma=0.2, C=1.0)
svm1.fit(X_train_std, y_train)

X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))

y_pred1 = svm1.predict(X_test_std)
from sklearn.metrics import classification_report, confusion_matrix
print("Linear Kernal Confusion Matirx:")
print(confusion_matrix(y_test,y_pred1))
print(classification_report(y_test,y_pred1))


#ploting on 2 feature model
svm1_plot = SVC(kernel='linear', random_state=1, gamma=0.2, C=1.0)
svm1_plot.fit(plot_X, plot_y)

plot_decision_regions(plot_X, plot_y, classifier=svm1_plot)
plt.xlabel('petal length [standardized]')
plt.ylabel('petal width [standardized]')
plt.legend(loc='upper left')
plt.show()
```

```
Linear Kernal Confusion Matirx:
[[15  0  0]
 [ 0 14  1]
 [ 0  0 15]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       1.00      0.93      0.97        15
           2       0.94      1.00      0.97        15

    accuracy                           0.98        45
   macro avg       0.98      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45

<ipython-input-23-132b2f9bf9b1>:20: UserWarning: You passed a edgecolor/edgecolors
  plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
```

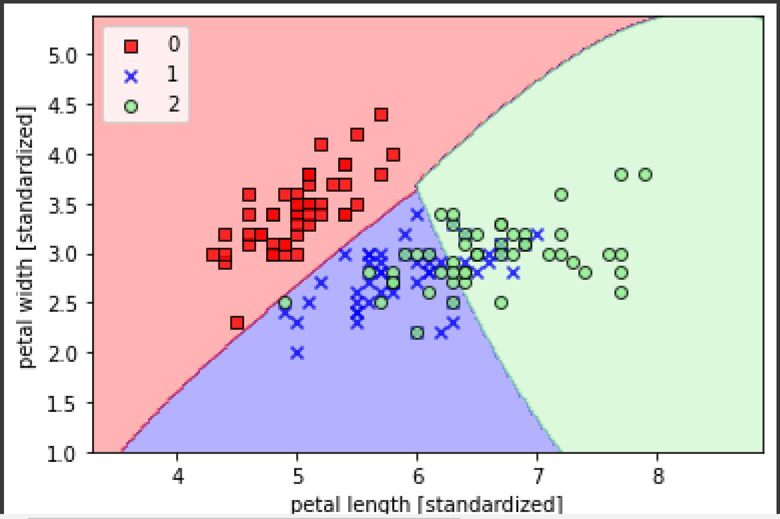


Kernel Two: RFB

```
#@title Kernel Two: RFB
import sklearn
from sklearn.svm import SVC
import matplotlib.pyplot as plt
import numpy as np

svm2 = SVC(kernel='rbf', random_state=1, gamma=0.2, C=1.0)
svm2.fit(X_train_std, y_train)

#finding confurion matrix
y_pred2 = svm2.predict(X_test_std)
from sklearn.metrics import classification_report, confusion_matrix
print("RBF Kernal Confusion Matirx:")
print(confusion_matrix(y_test,y_pred2))
print(classification_report(y_test,y_pred2))


#ploting on 2 feature model
svm2_plot = SVC(kernel='rbf', random_state=1, gamma=0.2, C=1.0)
svm2_plot.fit(plot_X, plot_y)
```

```
plot_decision_regions(plot_X, plot_y, classifier=svm2_plot)
plt.xlabel('petal length [standardized]')
plt.ylabel('petal width [standardized]')
plt.legend(loc='upper left')
plt.show()
```

```
RBF Kernal Confusion Matirx:
[[15  0  0]
 [ 0 14  1]
 [ 0  1 14]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       0.93      0.93      0.93        15
           2       0.93      0.93      0.93        15

    accuracy                           0.96        45
   macro avg       0.96      0.96      0.96        45
weighted avg       0.96      0.96      0.96        45

<ipython-input-23-132b2f9bf9b1>:20: UserWarning: You passed a edgecolor/edgecolors
  plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
```

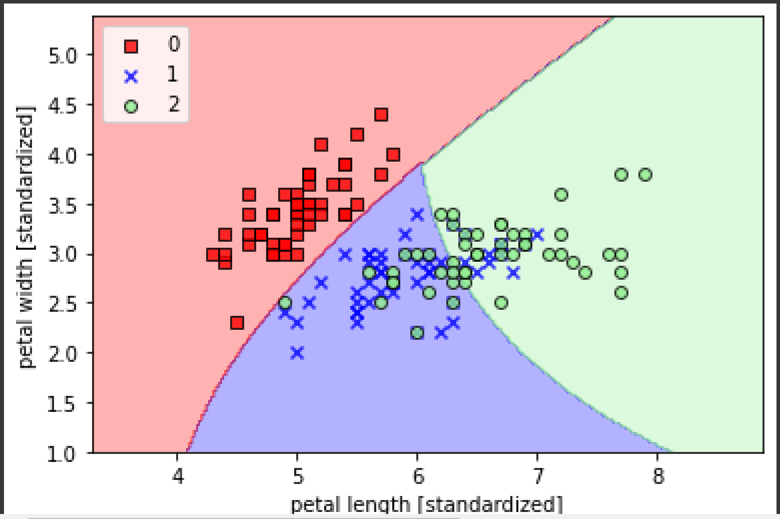


Kernel Three: Polynomial

```
#@title Kernel Three: Polynomial
import sklearn
from sklearn.svm import SVC
import matplotlib.pyplot as plt
import numpy as np
svm3 = SVC(kernel='poly', random_state=1, gamma=0.2, C=1.0)
svm3.fit(X_train_std, y_train)

#finding confusion matrix
y_pred3 = svm3.predict(X_test_std)
from sklearn.metrics import classification_report, confusion_matrix
print("Polynomial Kernal Confusion Matirx:")
print(confusion_matrix(y_test,y_pred3))
print(classification_report(y_test,y_pred3))
```

```
#ploting on 2 feature model
svm3_plot = SVC(kernel='poly', random_state=1, gamma=0.2, C=1.0)
svm3_plot.fit(plot_X, plot_y)

plot_decision_regions(plot_X, plot_y, classifier=svm3_plot)
plt.xlabel('petal length [standardized]')
plt.ylabel('petal width [standardized]')
plt.legend(loc='upper left')
plt.show()
```

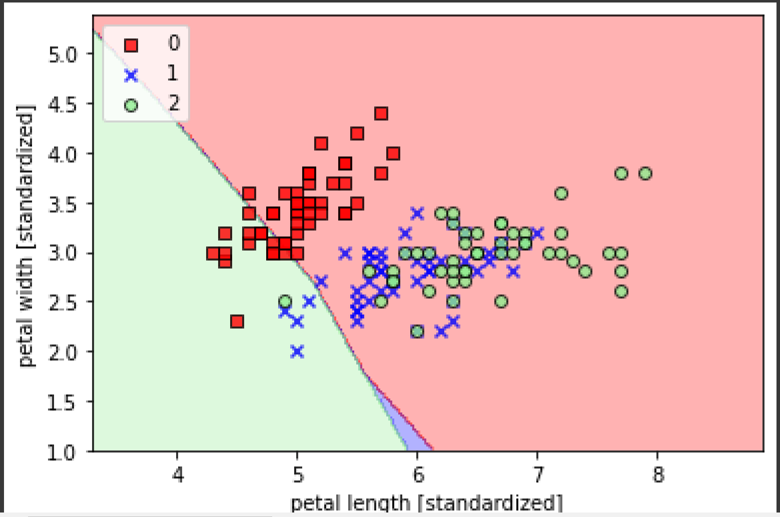
```
Polynomial Kernal Confusion Matirx:
[[15  0  0]
 [ 0 15  0]
 [ 0  4 11]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       0.79      1.00      0.88        15
           2       1.00      0.73      0.85        15

    accuracy                           0.91        45
   macro avg       0.93      0.91      0.91        45
weighted avg       0.93      0.91      0.91        45

<ipython-input-23-132b2f9bf9b1>:20: UserWarning: You passed a edgecolor/edgecolors
  plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
```



Kernel Four: Sigmoid

```
#@title Kernel Four: Sigmoid
import sklearn
from sklearn.svm import SVC
import matplotlib.pyplot as plt
import numpy as np

svm4 = SVC(kernel='sigmoid', random_state=1, gamma=0.2, C=1.0)
svm4.fit(X_train_std, y_train)

#finding confusion matrix
y_pred4 = svm4.predict(X_test_std)
from sklearn.metrics import classification_report, confusion_matrix
print("Sigmoid Kernal Confusion Matirx:")
print(confusion_matrix(y_test,y_pred4))
print(classification_report(y_test,y_pred4))


#ploting on 2 feature model
svm4_plot = SVC(kernel='sigmoid', random_state=1, gamma=0.2, C=1.0)
svm4_plot.fit(plot_X, plot_y)

plot_decision_regions(plot_X, plot_y, classifier=svm4_plot)
plt.xlabel('petal length [standardized]')
plt.ylabel('petal width [standardized]')
plt.legend(loc='upper left')
plt.show()
```

```
Sigmoid Kernal Confusion Matirx:
[[15  0  0]
 [ 0 12  3]
 [ 0  1 14]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       0.92      0.80      0.86        15
           2       0.82      0.93      0.87        15

    accuracy                           0.91        45
   macro avg       0.92      0.91      0.91        45
weighted avg       0.92      0.91      0.91        45

<ipython-input-23-132b2f9bf9b1>:20: UserWarning: You passed a
  plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
```



Discussion:

The linear kernal model fit the data best. It had an accuracy of 98% in classifing the flowers with four features. It's likely that the other models over failed because I took steps to standardize the data and split it into training and test data. however, we can see on the graphs that the decision boundaries for each classification is tighter in all the kernals that don't use the linear method.

✓ 1s completed at 6:48 PM