In [ ]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn.datasets as datasets
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split

iris_data = pd.read_csv('Iris.csv')
iris_data
```

Out[ ]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

In [ ]:
```
iris_data.shape
```

Out[ ]:
```
(150, 6)
```

In [ ]:
```
iris_data.nunique()
```

Out[ ]:
```
Id               150
SepalLengthCm     35
SepalWidthCm      23
PetalLengthCm     43
PetalWidthCm      22
Species            3
dtype: int64
```

In [ ]:
```
iris_data.isnull().sum()
```

Out[ ]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [ ]:
```
iris_data.Species.replace(('Iris-setosa', 'Iris-versicolor', 'Iris-virginica
iris_data
```

Out[ ]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	1
1	2	4.9	3.0	1.4	0.2	1
2	3	4.7	3.2	1.3	0.2	1
3	4	4.6	3.1	1.5	0.2	1
4	5	5.0	3.6	1.4	0.2	1
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	3
146	147	6.3	2.5	5.0	1.9	3
147	148	6.5	3.0	5.2	2.0	3
148	149	6.2	3.4	5.4	2.3	3
149	150	5.9	3.0	5.1	1.8	3

150 rows × 6 columns

In [ ]:
```
x=iris_data.drop(['Species','Id'],axis='columns')
```

In [ ]:
```
y=iris_data.Species
```

In [ ]:
```
x_train,x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2, rand
```

In [ ]:
```
model = SVC(kernel='linear')
```

In [ ]:
```
model.fit(x_train,y_train)
```

Out[ ]:
```
SVC
SVC(kernel='linear')
```

In [ ]:
```
y_pred=model.predict(x_test)
print(y_pred)
```

```
[1 1 3 1 1 3 1 3 3 1 1 1 1 1 2 2 1 2 3 2 2 2 3 2 2 1 1 3 1 3]
```

In [ ]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

Out[ ]:
```
array([[14,  0,  0],
       [ 0,  8,  0],
       [ 0,  0,  8]])
```

In [ ]:
```
from sklearn.metrics import classification_report
print(classification_report(y_true=y_test,y_pred=y_pred, target_names=['Iris
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       1.00      1.00      1.00         8
 Iris-virginica       1.00      1.00      1.00         8

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```

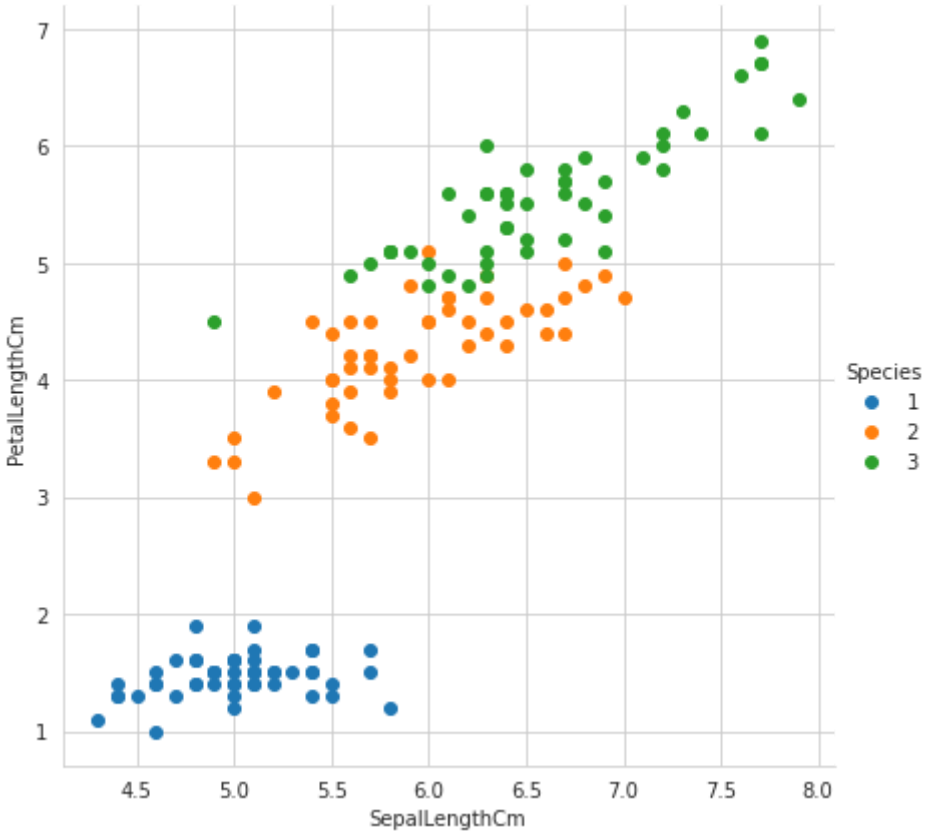
In [ ]:
```
model.score(x_test,y_test)
```

Out[ ]:
```
1.0
```

In [ ]:
```
iris = datasets.load_iris()
# Take the first two features. We could avoid this by using a two-dim datase
X = iris.data[:, :2]
y = iris.target
C=1
```

In [ ]:
```
sns.set_style("whitegrid")
sns.FacetGrid(iris_data, hue ="Species",
              height = 6).map(plt.scatter,
                              'SepalLengthCm',
                              'PetalLengthCm').add_legend()
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7fbbcb493730>
```



In [ ]:
```
models = (
    SVC(kernel="linear", C=C),
    SVC(kernel="rbf", gamma=0.7, C=C),
    SVC(kernel="poly", degree=3, gamma="auto", C=C),
)
models = (clf.fit(X, y) for clf in models)
```

```
# title for the plots
titles = (
    "SVC with linear kernel",
    "SVC with RBF kernel",
    "SVC with polynomial (degree 3) kernel",
)
```

In [ ]:

```
fig, sub = plt.subplots(2, 2)
plt.subplots_adjust(wspace=0.4, hspace=0.4)

X0, X1 = X[:, 0], X[:, 1]

for clf, title, ax in zip(models, titles, sub.flatten()):
    disp = DecisionBoundaryDisplay.from_estimator(
        clf,
        X,
        response_method="predict",
        cmap=plt.cm.coolwarm,
        alpha=0.8,
        ax=ax,
        xlabel="Sepal length(cm)",
        ylabel="Sepal width(cm)",
    )
    ax.scatter(X0, X1, c=y, cmap=plt.cm.coolwarm, s=20, edgecolors="k")
    ax.set_xticks(())
    ax.set_yticks(())
    ax.set_title(title)

plt.show()
```