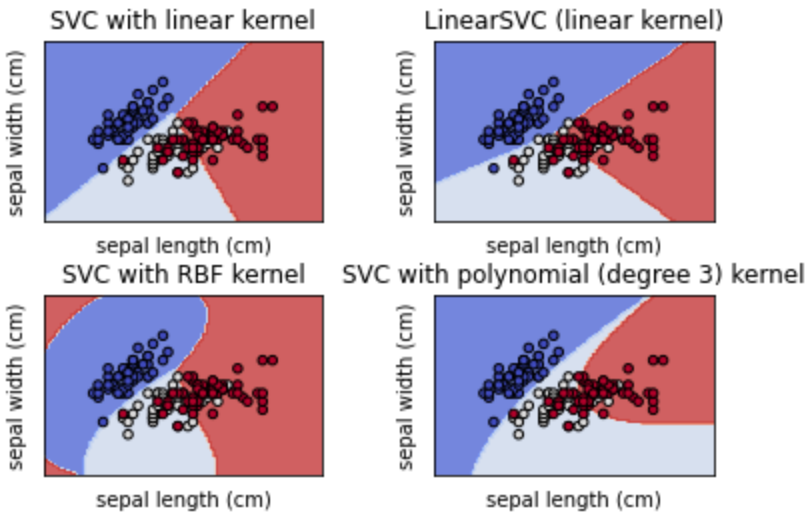
Khan Inan
Course 7743
Assignment #4
Professor Katari

```
pip install --upgrade scikit-learn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.8/dist-packages (1.0.2)
Collecting scikit-learn
  Downloading scikit_learn-1.2.1-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (9.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.8/9.8 MB 98.5 MB/s eta 0:00:00
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (3.1.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (1.22.4)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.8/dist-packages (from scikit-learn) (1.7.3)
Installing collected packages: scikit-learn
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 1.0.2
    Uninstalling scikit-learn-1.0.2:
      Successfully uninstalled scikit-learn-1.0.2
Successfully installed scikit-learn-1.2.1
```

```
import matplotlib.pyplot as plt
from sklearn import svm, datasets
from sklearn.inspection import DecisionBoundaryDisplay


# import some data to play with
iris = datasets.load_iris()
# Take the first two features. We could avoid this by using a two-dim dataset
X = iris.data[:, :2]
y = iris.target

# we create an instance of SVM and fit out data. We do not scale our
# data since we want to plot the support vectors
C = 1.0  # SVM regularization parameter
models = (
    svm.SVC(kernel="linear", C=C),
    svm.LinearSVC(C=C, max_iter=10000),
    svm.SVC(kernel="rbf", gamma=0.7, C=C),
    svm.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",
    "LinearSVC (linear kernel)",
    "SVC with RBF kernel",
    "SVC with polynomial (degree 3) kernel",
)

# Set-up 2x2 grid for plotting.
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=iris.feature_names[0],
        ylabel=iris.feature_names[1],
    )
    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()
```



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.

All of the kernels that I used seem to be relatively similar in terms of how scattered the data is. Where they differe the most is in terms of the skew of the various regions. The "SVC" with RBF kernel model seems to have the most skew where the regions are more abstract while the linearl kernels seem to have the regions dinstinctly seperated.

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.

So for the first kernel shown above, the pros are that the regions are seperated and the the clusters of data can be analyzed under seperate conditions. The cons of the first kernel is that there is no much overlap in terms of regions, and this would be useful if you are trying to see how the clusters compare. This also applies to the second kernel as well. The third kernel is interesting because the red region encompasses the other two, so this kernel would be useful for comparing the red data set with the other features. It is not as useful for analyzing the white or blue features respectively. Lastly the last kernel has a skew where the white data set intersects the other features, this results in a more useful comparison/analysis of the white data set compared to the other two colors